Моделирование и анализ информационных систем. Т. 26, № 4 (2019), с. 550–571
Modeling and Analysis of Information Systems. Vol. 26, No 4 (2019), pp. 550–571

Algorithms

DOI: 10.18255/1818-1015-2019-4-550-571

УДК 004.05

Направляемый свойством поиск реляционных инвариантов

Мордвинов Д. А.

Поступила в редакцию 1 октября 2019
После доработки 18 ноября 2019
Принята к публикации 27 ноября 2019

Аннотация. Достижимость, направляемая свойством, (Property Directed Reachability, PDR) — эффективный и масштабируемый подход к решению систем символьных ограничений, известных как дизъюнкты Хорна с ограничениями (Constrained Horn Clauses, CHC). В случае нелинейных систем дизъюнктов, которые могут возникнуть, к примеру, из задач реляционной верификации, PDR выводит индуктивные инварианты для каждого неинтерпретированного предикатного символа. Тем не менее на практике автоматический вывод таких решений не удаётся, т.к. инварианты должны выводиться для групп предикатных символов вместо индивидуальных предикатных символов. В статье описан новый алгоритм, автоматически определяющий такие группы и обобщающий существующий подход PDR. Ключевая особенность алгоритма состоит в том, что он не требует потенциально дорогой синхронизирующей трансформации системы дизъюнктов Хорна. Алгоритм был реализован над современным решателем дизъюнктов Хорна Spacer. Эксперименты показывают, что полученная реализация успешно выводит реляционные инварианты для некоторых систем дизъюнктов, на которых существующие решатели не завершаются.

Ключевые слова: реляционная верификация, дизъюнкты Хорна с ограничениями, направляемая свойством достижимость, реляционные инварианты

Для цитирования: Мордвинов Д. А., "Направляемый свойством поиск реляционных инвариантов", *Моделирование и анализ информационных систем*, **26**:4 (2019), 550–571.

Об авторах:
Мордвинов Дмитрий Александрович, orcid.org/0000-0002-6437-3020, стар. преп.,
Санкт-Петербургский государственный Университет, JetBrains Research
Университетский пр., 28, г. Петродворец, 198504 Россия, e-mail: dmitry.mordvinov@jetbrains.com

Благодарности:
Работа выполнена при финансовой поддержке JetBrains Research.

Введение

С развитием подходов к автоматической формальной верификации программ относительно функциональных спецификаций [1–8], растет потребность применения этих подходов к верификации свойств, описывающих отношения между входами-выходами различных программ [9–12]. Эта дисциплина, называемая *реляционной*

верификацией, широко применима в итеративном процессе разработки программного обеспечения, когда текущая и предыдущая версии сравниваются и проверяются на предмет отсутствия нововведённых ошибок. Другой пример применения — доказательство свойств безопасности потока информации, таких как невмешательство и балансировка времени [13], в которых исполнения одной и той же программы сравниваются для различных входов.

Многие автоматические подходы к реляционной верификации основываются на построении *произведения* сравниваемых программ [9, 14–18]. В таком случае реляционная спецификация над несколькими программами (или несколькими исполнениями одной программы) становится функциональной спецификацией над произведением этих программ. В теории такой подход позволяет верифицировать реляционные спецификации существующими подходами, однако на практике большинство существующих техник не могут справиться с потенциально сложной структурой программы-произведения. Проблему можно смягчить *слиянием* определенных циклов в произведении программ (т.е. применением т.н. *стратегий синхронизации*), но зачастую они обнаруживаются либо вручную, либо использованием неточных синтаксических эвристик. Другой недостаток состоит в том, что количество возможных стратегий экспоненциально зависит от количества сливаемых программ. Статья описывает новый полностью автоматический подход к определению стратегий синхронизации, который ведёт к эффективному нахождению *реляционных инвариантов* для произведений программ.

Предложенный подход обобщает одну из самых успешных реализаций PDR А. Гурфинкелем и др. [3] под названием Spacer. PDR инкрементально усиливает функциональную спецификацию (т.е. свойство безопасности) до тех пор, пока она либо не станет индуктивной, либо будет обнаружен контрпример. Он моделирует программы набором логических импликаций над множеством неинтерпретированных предикатных символов, называемых дизъюнктами Хорна с ограничениями. Неформально говоря, дизъюнкты задают семантику неинтерпретированных символов, и при нахождении интерпретации этих символов, выполняющие множество дизъюнктов, можно определить индуктивные инварианты верифицируемой программы. Spacer итеративно аппроксимирует семантику неинтерпретированных символов сверху и снизу; аппроксимации сверху используются для блокировки ложных контрпримеров, апроксимации снизу — для анализа веток программы без раскрутки отношения перехода.

Дизъюнкты, построенные для произведения программ, нелинейны, где неинтерпретированные символы соответствуют сравниваемым программам. В статье предлагается новый направляемый свойством подход, поддерживающий аппроксимации сверху и снизу семантик *групп* предикатов. Он работает сходным образом с обычным исследованием произведения программ, но *без явного вычисления произведения программ*. Более важно, что алгоритм автоматически определяет релевантные группы предикатов, анализируя контрпримеры к индуктивности, полученные на различных стадиях процесса верификации. Это позволяет эффективно отсекать пространство поиска различных стратегий синхронизации, что приводит к увеличению производительности. Без применения предложенного подхода, PDR пытается выводить инварианты для каждого символа в отдельности, что зачастую не полу-

чается, т.к., например, желаемые инварианты не выразимы в языке моделирования поведений программы.

Подход был реализован на базе Spacer и апробирован на тестах, полученных из различных задач реляционной верификации. Эксперименты подтверждают, что для многих систем дизъюнктов Хорна, на которых SPACER не завершается, наш подход способен быстро выводить реляционные инварианты.

Статья устроена следующим образом. В секции 1. собраны предварительные сведения, в секции 2. дано определение реляционных инвариантов систем дизъюнктов Хорна с ограничениями, ранее не представленное в литературе. Алгоритм определения стратегий синхронизации, обобщающий PDR, представлен в секции 3.. В секции 4. представлены экспериментальные данные, секции 5. и 6. завершают статью.

1. Предварительные сведения

1.1. Язык ограничений

Пусть Σ — сигнатура языка первого порядка с равенством, и пусть $\mathcal{M}$ — Σ-структура с носителем $|\mathcal{M}|$. $\mathcal{M}$ — *модель* Σ-предложения φ (т.е. Σ-формулы без свободных переменных), если φ выполняется в $\mathcal{M}$, что обозначается $\mathcal{M} \models \varphi$. Будем называть язык первого порядка, определяемый Σ, *языком ограничений*. Для Σ-формулы с n свободными переменными $\varphi(x_1, \ldots, x_n)$, обозначим $\mathcal{M}(\varphi)$ множество оценок свободных переменных, выполняющих φ в $\mathcal{M}$, т.е. $\mathcal{M}(\varphi) \stackrel{\text{def}}{=} \{\langle a_1, \ldots, a_n\rangle \mid \mathcal{M} \models \varphi(a_1, \ldots, a_n)\} \subseteq |\mathcal{M}|^n$.

1.2. Дизъюнкты Хорна с ограничениями

Пусть $\mathcal{R} = \{P_0, P_1, \ldots, P_n\}$ — конечное множество предикатных символов, которые будем называть *реляционными* (или *неинтерпретированными*) символами. *Дизъюнкт Хорна с ограничением* — это $\Sigma \cup \mathcal{R}$-формула вида

$$\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m) \Rightarrow R(\overline{v}_R),$$

где φ — бесквантворная Σ-формула, $R, R_i \in \mathcal{R}$, $\overline{v}_R$ и $\overline{x}_i$ — векторы переменных. Посылка импликации называется *телом* дизъюнкта C и обозначается $body(C)$, заключение $R(\overline{v}_R)$ называется *головой* дизъюнкта. *Система дизъюнктов* — это конечное множество дизъюнктов. Символ P_0 — специальный символ "запросов", корень всех деревьев вывода системы дизъюнктов. Если в тело дизъюнкта входит не более одного применения неинтерпретированного символа, дизъюнкт называется *линейным*, в противном случае — *нелинейным*. Система дизъюнктов линейна, если каждый дизъюнкт в ней линеен.

1.3. Проблема безопасности

Пара $\langle \mathcal{P}, \varphi_{safe}\rangle$ называется *проблемой безопасности* системы дизъюнктов $\mathcal{P} = \{C_1, \ldots, C_n\}$ относительно *свойства безопасности* φ_{safe} — Σ-формулы над пере-

менными $\overline{v}_{P_0}$. Множество *правил* реляционного символа R, обозначаемое $rules(R)$, — множество дизъюнктов $\mathcal{P}$ с применением R в голове. Без потери общности будем считать, что в голове каждого правила одного и символа R находятся применения R к одному и тому же набору переменных $\overline{v}_R$, т.е. если дизъюнкты $C_i, C_j \in \mathcal{P}$ имеют головы $R(\overline{v}_R)$ и $R(\overline{v}'_R)$, то $\overline{v}_R = \overline{v}'_R$.

Расширим нотацию $body$, обозначив за $body(R)$ дизъюнкцию тел правил для R:

$$body(R) \stackrel{\text{def}}{=} \bigvee_{C \in rules(R)} body(C).$$

Тело слияния реляционных символов $R_1, \ldots, R_m \in \mathcal{R}$ — это конъюнкция их тел

$$body(R_1, \ldots, R_m) \stackrel{\text{def}}{=} body(R_1) \stackrel{+}{\wedge} \ldots \stackrel{+}{\wedge} body(R_m),$$

где $\varphi \stackrel{+}{\wedge} \psi$ — конъюнкция φ и ψ, гарантирующая, что свободные переменные операндов различны $\varphi \wedge \psi$:

$$\varphi(\overline{x}) \stackrel{+}{\wedge} \psi(\overline{y}) \stackrel{\text{def}}{=} (\varphi \wedge \psi)\,(\overline{x} \uplus \overline{y})\,.$$

$\overline{\ell}_R$ обозначает вектор *экзистенциальных* (или *локальных*) переменных R, т.е. свободных переменных $body(R)$, не входящих в переменные головы $\overline{v}_R$.

Пример 1. Пусть $\mathcal{M}$ — комбинация стандартной модели линейной целочисленной арифметики и начальной модели теории алгебраических типов данных с сортом $tree$, заданным конструкторами $leaf : tree$ и $node : \mathbb{N} \times tree \times tree \to tree$. Рассмотрим следующую проблему безопасности, которая 1) считает количество узлов дерева (реляционный символ $size$), 2) суммирует значения в узлах дерева (реляционный символ sum) и 3) строит новое дерево, полученное из старого увеличением всех значений в узлах на 2 (реляционный символ inc):

$$\begin{aligned}
T = leaf \wedge n = 0 &\Rightarrow size(T,n)\\
T = node(v,L,R) \wedge n = 1 + n^L + n^R \wedge size(L,n^L) \wedge size(R,n^R) &\Rightarrow size(T,n)\\
T = leaf \wedge s = 0 &\Rightarrow sum(T,s)\\
T = node(v,L,R) \wedge s = v + s^L + s^R \wedge\\
sum(L,s^L) \wedge sum(R,s^R) &\Rightarrow sum(T,s)\\
T = leaf \wedge U = leaf &\Rightarrow inc(T,U)\\
T = node(v,L,R) \wedge U = node(v+2,L',R') \wedge\\
inc(L,L') \wedge inc(R,R') &\Rightarrow inc(T,U)\\
size(T,n) \wedge sum(T,s) \wedge inc(T,T') \wedge sum(T',s') &\Rightarrow P_0(T,n,s,s')\\
\varphi_{safe} \stackrel{\text{def}}{=} s' = s + 2n
\end{aligned}$$

Требуется доказать, что сумма значений в узлах второго дерева равняется сумме в узлах оригинального плюс удвоенное количество узлов в дереве. Здесь $\mathcal{R} = \{P_0, size, sum, inc\}$, $\overline{v}_{P_0} = \{T, n, s, s'\}$, $\overline{\ell}_{P_0} = \{T'\}$, $\overline{v}_{size} = \{T, n\}$, $\overline{\ell}_{size} = \{v, L, R, n^L, n^R\}$, $body(size) = (T = leaf \wedge n = 0) \vee \big(T = node(v,L,R) \wedge n = 1 + n^L + n^R \wedge size(L,n^L) \wedge size(R,n^R)\big)$.

1.4. Семантика неподвижной точки

Пусть $k_0, k_1, \ldots, k_n$ — арности символов $P_0, P_1, \ldots, P_n$ соответственно. Пусть $\overline{X} = \langle X_0, X_1, \ldots, X_n \rangle$ — кортеж отношений на носителе $|\mathcal{M}|$, $X_i \subseteq |\mathcal{M}|^{k_i}$. За $\left(\mathcal{M}, \overline{X}\right)$ обозначим обогащение $\mathcal{M}\{P_0 \mapsto X_0, P_1 \mapsto X_1, \ldots, P_n \mapsto X_n\}$.

Семантика системы дизъюнктов $\mathcal{P}$ в структуре $\mathcal{M}$ — (поточечно) наименьшая $(n+1)$-ка отношений $\overline{X}$, такая, что для всех $P \in \mathcal{R}$,

$$\left(\mathcal{M}, \overline{X}\right) \models \forall \overline{v}_P \cup \overline{\ell}_P. \left(body(P) \Rightarrow P(\overline{v}_P)\right).$$

Семантика $\mathcal{P}$ — наименьшая неподвижная точка оператора немедленных последствий $\mathcal{P}$ [20]; она всегда существует по теореме Кнастера-Тарского [19, 20]. Будем называть элементы кортежа семантики всей системы семантиками соответствующего символа и обозначать их $[\![P_i]\!]_{\mathcal{M}}$; таким образом, семантикой системы является кортеж $\langle [\![P_0]\!]_{\mathcal{M}}, [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}} \rangle$.

Система дизъюнктов *безопасна* по отношению к φ_{safe}, если $[\![P_0]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\varphi_{safe})$. Например, система дизъюнктов в примере 1 безопасна по отношению к свойству безопасности $s' = s + 2n$.

1.5. Доказательства безопасности

Символьная интерпретация (далее просто интерпретация) Π отображает $P \in \mathcal{R}$ в Σ-формулу над $\Sigma \cup \overline{v}_P$. Для $\Sigma \cup \mathcal{R}$-формулы ψ, $[\![\psi]\!]_\Pi$ — это формула, полученная одновременной подстановкой всех применений реляционных символов в ψ формулами их Π-образов с соответствующим переименованием переменных.

Интерпретация Π — *доказательство безопасности* проблемы $\langle \mathcal{P}, \varphi_{safe} \rangle$, если Π безопасна и индуктивна:

$$\mathcal{M} \models \forall \overline{v}_{P_0}. \Pi(P_0) \Rightarrow \varphi_{safe} \qquad \text{(безопасность)}$$

$$\text{для всех } P \in \mathcal{R},\ \mathcal{M} \models \forall \overline{v}_P \cup \overline{\ell}_P. ([\![body(P)]\!]_\Pi \Rightarrow \Pi(P)) \qquad \text{(индуктивность)}$$

Предложение 1. Если существует доказательство безопасности проблемы $\langle \mathcal{P}, \varphi_{safe} \rangle$, то $\mathcal{P}$ безопасна по отношению к φ_{safe}.

Доказательство. Пусть Π — доказательство безопасности. По индуктивности, для кортежа $\overline{Y} = \langle \mathcal{M}(\Pi(P_0)), \mathcal{M}(\Pi(P_1)), \ldots, \mathcal{M}(\Pi(P_n)) \rangle$ имеем

$$\left(\mathcal{M}, \overline{Y}\right) \models \forall \overline{v}_P \cup \overline{\ell}_P. \left(body(P) \Rightarrow P(\overline{v}_P)\right)$$

для всех $P \in \mathcal{R}$. Т.к. $\langle [\![P_0]\!]_{\mathcal{M}}, [\![P_1]\!]_{\mathcal{M}}, \ldots, [\![P_n]\!]_{\mathcal{M}} \rangle$ — поточечно наименьший такой кортеж, имеем $[\![P_0]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\Pi(P_0))$. По безопасности, $\mathcal{M}(\Pi(M)) \subseteq \mathcal{M}(\varphi_{safe})$. Таким образом, имеем $[\![P_0]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\varphi_{safe})$. □

2. Реляционные инварианты

Системы дизъюнктов Хорна с ограничениями широко используются в автоматической верификации для доказательства корректности программ по отношению к спецификациям корректности. Однако, когда речь заходит о верификации реляционных свойств нескольких программ, моделируемых нелинейными дизъюнктами Хорна, рассуждать о таких системах становится сложнее. Зачастую доказательства безопасности не выразимы в языке ограничений. Например, несмотря на то, что система в примере 1 безопасна, *не существует доказательства безопасности, представимого в языке ограничений примера* (т.е. формулой, построенной из арифметических символов, равенства и символов *leaf* и *node*).

Пример 2. Рассмотрим более простой пример:

$$x=0 \wedge z=0 \Rightarrow mul(x,y,z)$$
$$x>0 \wedge x'=x-1 \wedge z=z'+y \wedge mul(x',y,z') \Rightarrow mul(x,y,z)$$
$$x=x' \wedge y=y' \wedge mul(x,y,z) \wedge mul(x',y',z') \Rightarrow P_0(x,y,z,x',y',z')$$
$$\varphi_{safe} \stackrel{\text{def}}{=} z=z'$$

Инвариант $mul(x,y,z) = (z = x \cdot y)$ не представим в линейной целочисленной арифметике, как и любая теоретико-множественная модель этой системы.

В этой секции обобщается классическое понятие доказательства безопасности таким образом, что в обоих этих случаях существует представимое доказательство. Ключевая идея состоит в том, чтобы сопоставлять формулы *группам* реляционных символов, а не каждому символу в отдельности. Это позволяет определять в языке ограничений *отношения* между переменными различных вычислений вместо резюмирования каждого вычисления в отдельности.

2.1. Определение реляционных интерпретаций

Обозначим за $\mathbb{N}^X$ множество мультимножеств на X, т.е. множество всех отображений X на множество натуральных чисел. Если $\overline{x} = x_1, \ldots, x_n$ — вектор элементов X (возможно повторяющихся), мы будем отождествлять его с мультимножеством $\{x_k \mapsto \#x_k\}$, где $\#x_i$ — число вхождений x_i в $\overline{x}$. Мультимножества частично упорядочены включением: для $m_1, m_2 \in \mathbb{N}^X$, $m_1 \subseteq m_2$, если $\forall x \in X, m_1(x) \leq m_2(x)$.

Определение 1. Пусть $\mathcal{P}$ — система дизъюнктов над множеством реляционных символов $\mathcal{R}$. *Реляционная интерпретация* — частичное отображение $\mathbb{N}^{\mathcal{R}}$ на множество формул языка ограничений, отображающее $\overline{R} = \{R_1 \mapsto n_1, \ldots, R_k \mapsto n_k\}$ в формулу над переменными $\overline{v}_{\overline{R}} \stackrel{\text{def}}{=} \underbrace{\overline{v}_{R_1} \uplus \ldots \uplus \overline{v}_{R_1}}_{n_1 \text{ times}} \uplus \ldots \uplus \underbrace{\overline{v}_{R_k} \uplus \ldots \uplus \overline{v}_{R_k}}_{n_k \text{ times}}$.

Пусть E — реляционная интерпретация. Обозначим за $dom(E)$ её область определения. Без потери общности будем считать, что $\mathcal{R} \subseteq dom(E)$: если $R \in \mathcal{R} \backslash dom(E)$, то отобразим R в $\top$. Пусть $R_1, \ldots, R_m \in \mathcal{R}$, φ — формула. Определим реляционную

подстановку индуктивно:

$$[\![\varphi \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m)]\!]_E \stackrel{\text{def}}{=} \varphi \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k} \in \{R_1,\ldots,R_m\} \\ \langle R_{i_1},\ldots,R_{i_k}\rangle \in dom(E)}} E(R_{i_1},\ldots,R_{i_k})(\overline{x}_{i_1},\ldots,\overline{x}_{i_k}) \tag{1}$$

и

$$\left[\!\!\left[\bigwedge_{i=1}^{k}\bigvee_{j=1}^{m_i} F_{i,j}\right]\!\!\right]_E \stackrel{\text{def}}{=} \bigvee_{\substack{1\le j_1\le m_1 \\ \ldots \\ 1\le j_k\le m_k}} [\![F_{1,j_1} \wedge \ldots \wedge F_{k,j_k}]\!]_E \tag{2}$$

Неформально, (1) перебирает все возможные варианты "групповых" подстановок в тело одного дизъюнкта $R_1(\overline{x}_1)\wedge\ldots\wedge R_m(\overline{x}_m)$. В (2) определяется реляционная подстановка в тело слияния нескольких символов, которое, по определению, есть конъюнкция дизъюнкций тел дизъюнктов. В (2) объединяются одним из $m_1\cdot\ldots\cdot m_k$ возможных способов, с дальнейшей групповой подстановкой в каждое из объединённых тел дизъюнктов.

Пример 3. Рассмотрим следующую систему дизъюнктов:

$$\begin{aligned} \varphi_1 &\Rightarrow f(x_1,x_2) \\ \varphi_2 \wedge f(x_1',x_2') \wedge f(x_1'',x_2'') &\Rightarrow f(x_1,x_2) \\ \psi_1 &\Rightarrow g(y) \\ \psi_2 \wedge g(y') &\Rightarrow g(y) \end{aligned}$$

и следующую реляционную интерпретацию:

$$E = \{f \mapsto \top, g \mapsto \eta_1(y), \langle f,g\rangle \mapsto \eta_2(x_1,x_2,y)\}.$$

Результат подстановки E в $body(f,g)$ таков:

$$\begin{aligned} [\![body(f,g)]\!]_E = \Big[\!\Big[\Big(\varphi_1 \vee \big(\varphi_2 \wedge f(x_1',x_2') \wedge f(x_1'',x_2'')\big)\Big) \wedge \\ \Big(\psi_1 \vee \big(\psi_2 \wedge g(y')\big)\Big)\Big]\!\Big]_E = \\ [\![\varphi_1 \wedge \psi_1]\!]_E \vee [\![\varphi_1 \wedge \psi_2 \wedge g(y')]\!]_E \vee \\ \quad [\![\varphi_2 \wedge \psi_1 \wedge f(x_1',x_2') \wedge f(x_1'',x_2'')]\!]_E \vee \\ \quad [\![\varphi_2 \wedge \psi_2 \wedge f(x_1',x_2') \wedge f(x_1'',x_2'') \wedge g(y')]\!]_E = \\ (\varphi_1 \wedge \psi_1) \vee (\varphi_1 \wedge \psi_2 \wedge \eta_1(y')) \vee (\varphi_2 \wedge \psi_1) \vee \\ \quad (\varphi_2 \wedge \psi_2 \wedge \eta_1(y') \wedge \eta_2(x_1',x_2',y') \wedge \eta_2(x_1'',x_2'',y')) \end{aligned}$$

Реляционные интерпретации обобщают "классические" интерпретации: если E — реляционная интерпретация, определённая только на *единичных* мультимножествах (т.е. мультимножествах, состоящих из одного элемента, входящего один раз), а Π — "классическая" интерпретация, отображающая те же символы в те же формулы, то для всех $\Sigma\cup\mathcal{R}$-формул φ, $[\![\varphi]\!]_E$ логически равносильно $[\![\varphi]\!]_\Pi$.

2.2. Реляционные доказательства безопасности

Реляционная интерпретация E — *доказательство безопасности* проблемы $\langle \mathcal{P}, \varphi_{safe} \rangle$, если она безопасна и индуктивна:

$$\mathcal{M} \models \forall \overline{v}_{P_0}.E(P_0) \Rightarrow \varphi_{safe} \qquad \text{(безопасность)}$$
$$\text{для всех } \overline{P} \in dom(E),\ \mathcal{M} \models \forall \overline{v}_{\overline{P}} \cup \overline{\ell}_{\overline{P}}.\big([\![body(\overline{P})]\!]_E \Rightarrow E(\overline{P}) \big). \qquad \text{(индуктивность)}$$

Кроме реляционных подстановок, главная разница между "классическими" и реляционными доказательствами безопасности состоит в том, что последние должны быть индуктивны относительно тел *слияний* символов. Т.е. если $\overline{P} \in dom(E)$ для неединичного мультимножества $\overline{P}$, то E должно быть индуктивно относительно $body(\overline{P})$.

Пример 4. Хотя для системы в примере 2 не существует "классического" доказательства безопасности, представимое в языке линейной целочисленной арифметики, существует реляционное доказательство безопасности:

$$E = \{P_0 \mapsto z = z',\ mul \mapsto \top,$$
$$\langle mul, mul \rangle \mapsto (x^{mul_1} = x^{mul_2} \wedge y^{mul_1} = y^{mul_2} \Rightarrow z^{mul_1} = z^{mul_2})\}$$

Пример 1 также имеет бескванторное реляционное доказательство безопасности:

$$E = \{P_0 \mapsto s' = s + 2n, \quad sum \mapsto \top, \quad inc \mapsto \top,$$
$$\langle size, sum, sum, inc \rangle \mapsto T^{size} = T^{sum_1} = T^{inc} \wedge$$
$$T^{sum_2} = U^{inc_2} \Rightarrow s^{sum_2} = s^{sum_1} + 2n^{size}\}$$

2.3. Корректность

Теорема 1. *Если существует реляционное доказательство безопасности проблемы* $\langle \mathcal{P}, \varphi_{safe} \rangle$*, то* $\mathcal{P}$ *безопасна по отношению к* φ_{safe}*.*

Доказательство. По безопасности E, достаточно показать, что $[\![P_0]\!]_{\mathcal{M}} \subseteq \mathcal{M}(E(P_0))$. Докажем это, построив другую систему дизъюнктов $\mathcal{P}'$ и "классическое" доказательство её безопасности П, используя E.

Для каждого $\overline{P} \in dom(E)$ введём новый реляционный символ $R_{\overline{P}}$. Определим реляционную интерпретацию E', которая отображает $\overline{P}$ на $R_{\overline{P}}(\overline{v}_{\overline{P}})$. Теперь определим $\mathcal{P}'$ над реляционными символами $\{R_{\overline{P}} \mid \overline{P} \in dom(E)\}$. Для каждого $R_{\overline{P}}$, положим $body(R_{\overline{P}}) \stackrel{\text{def}}{=} [\![body(\overline{P})]\!]_{E'}$; голова каждого правила для $R_{\overline{P}}$ — это $R_{\overline{P}}(\overline{v}_{\overline{P}})$.

Например, для системы $\mathcal{P}$ и реляционной интерпретации E в примере 3, $\mathcal{P}'$ следующая:

$$\begin{aligned}
\varphi_1 &\Rightarrow R_f(x_1, x_2)\\
\varphi_2 \wedge R_f(x_1', x_2') \wedge R_f(x_1'', x_2'') &\Rightarrow R_f(x_1, x_2)\\
\psi_1 &\Rightarrow R_g(y)\\
\psi_2 \wedge R_g(y') &\Rightarrow R_g(y)\\
\varphi_1 \wedge \psi_1 &\Rightarrow R_{f,g}(x_1, x_2, y)\\
\varphi_1 \wedge \psi_2 \wedge R_g(y') &\Rightarrow R_{f,g}(x_1, x_2, y)\\
\varphi_2 \wedge \psi_1 \wedge R_f(x_1', x_2') \wedge R_f(x_1'', x_2'') &\Rightarrow R_{f,g}(x_1, x_2, y)\\
\varphi_2 \wedge \psi_2 \wedge R_f({x_1}', {x_2}') \wedge R_f({x_1}'', {x_2}'') \wedge R_g(y') \wedge&\\
R_{f,g}({x_1}', {x_2}', y) \wedge R_{f,g}({x_1}'', {x_2}'', y) &\Rightarrow R_{f,g}(x_1, x_2, y)
\end{aligned}$$

Семантика $R_{\overline{P}}$ в $\mathcal{P}'$ есть декартово произведение $\bigtimes_{p\in\overline{P}} [\![p]\!]_{\mathcal{M}}$, т.к. $R_{\overline{P}}$ применяется к переменным различных отношений и "достигает" каждого состояния, достигаемого каждым $p \in \overline{P}$. В частности,

$$[\![P_0]\!]_{\mathcal{M}} = [\![R_{P_0}]\!]_{\mathcal{M}}\,. \tag{3}$$

Наконец, определим "классическую" интерпретацию Π, отображающую $R_{\overline{P}}$ на $E(\overline{P})$. Π — доказательство безопасности $\mathcal{P}'$: она безопасна и индуктивна по построению. Обратим внимание, что

$$\Pi(R_{P_0}) = E(P_0). \tag{4}$$

Т.к. семантика $\mathcal{P}'$ — поточечно наименьший индуктивный кортеж отношений, имеем

$$[\![R_{P_0}]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\Pi(R_{P_0})). \tag{5}$$

По (3), (4) и (5) получаем:

$$[\![P_0]\!]_{\mathcal{M}} = [\![R_{P_0}]\!]_{\mathcal{M}} \subseteq \mathcal{M}(\Pi(R_{P_0})) = \mathcal{M}(E(P_0)) \subseteq \mathcal{M}(\varphi_{safe}).$$

□

2.4. Быстрая подстановка

По (2), наивная реализация $[\![\varphi]\!]_E$ требует вычисления $m_1 \cdot \ldots \cdot m_k$ правил, что имеет экспоненциальную сложность с ростом k. В этом разделе демонстрируется альтернативный метод, позволяющий избежать явного перечисления всех комбинаций сливаемых правил, что позволяет преодолеть наиболее слабую сторону подходов к синтаксическому преобразованию системы [16, 17].

Ключевая идея состоит в построении *эквивыполнимой* формулы вместо точного вычисления, используя правила (1) и (2).

Пусть $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ — все применения реляционных символом в φ. Для каждого применения $R_i(\overline{x}_i)$ заведём пропозициональный атом (т.е. нульместный предикатный символ) a_i. Пусть φ' — формула, полученная заменой всех вхождений $R_i(\overline{x}_i)$ на a_i для всех i в φ. Заметим, что $[\![\varphi]\!]_E$ эквивыполнимо с

$$\varphi' \wedge \bigwedge_{\substack{R_{i_1},\ldots,R_{i_k}\in\{R_1,\ldots,R_m\}\\ \langle R_{i_1},\ldots,R_{i_k}\rangle\in dom(E)}} \left(a_{i_1}\wedge\ldots\wedge a_{i_k} \Rightarrow E(R_{i_1},\ldots,R_{i_k})(\overline{x}_{i_1},\ldots,\overline{x}_{i_k})\right).$$

Пример 5. Для примера 3, следующая формула эквивыполнима с $[\![body(f,g)]\!]_E$:

$$\big(\varphi_1 \vee (\varphi_2 \wedge a_1 \wedge a_2)\big) \wedge \big(\psi_1 \vee (\psi_2 \wedge a_3)\big) \wedge \big(a_3 \Rightarrow \eta_1(y')\big) \wedge \\ \wedge \big(a_1 \wedge a_3 \Rightarrow \eta_2(x_1', x_2', y')\big) \wedge \big(a_2 \wedge a_3 \Rightarrow \eta_2(x_1'', x_2'', y')\big)$$

Этот метод используется в реализации RelRecMc (см. секцию 3.); он позволяет сохранить композициональность в выводе реляционных лемм и значительно повышает скорость вычисления реляционной подстановки по сравнению с наивной реализацией.

3. Алгоритм

В данной секции описывается направляемый свойством алгоритм автоматического вывода реляционных инвариантов. Алгоритм расширяет RecMC из [3]; мы заимствуем оттуда нотацию и общую структуру алгоритма и отсылаем читателя для более подробных объяснений и доказательств корректности.

3.1. Леммы и ветви

Алгоритм поддерживает две структуры данных ρ и σ, в которых хранит *достижимые ветви* системы и *леммы* о системе соответственно. Первая отображает $P \in \mathcal{R}$ и натуральное число b на множество формул над $\overline{v}_P$, вторая отображает мультимножество $\overline{P} \in \mathbb{N}^{\mathcal{R}}$ и натуральное число b на множество формул над $\overline{v}_{\overline{P}}$.

Алгоритм использует ρ для нахождения контрпримеров к безопасности и σ для построения реляционных доказательств. В частности, в ρ размещаются *факты о достижимости*, т.е. достижимые ветви системы; $\rho(P,b)$ — множество формул, аппроксимирующих снизу семантику P глубины b, т.е. объединение всех выводов системы сверху вниз высоты b. Двойственно, в σ размещаются факты, *резюмирующие* систему, которые называются *леммами*. Формулы в $\sigma(P,b)$ аппроксимируют сверху семантику P глубины b, они используются для построения реляционного доказательства безопасности.

ρ и σ неявно определяют "классическую" интерпретацию U^b_ρ и реляционную интерпретацию O^b_σ соответственно, аппроксимирующие семантику системы глубины b сверху и снизу[1]:

$$U^b_\rho(P) \stackrel{\text{def}}{=} \bigvee \{\delta \in \rho(P,c) \mid c \le b\}\,,$$
$$O^b_\sigma(\overline{P}) \stackrel{\text{def}}{=} \bigwedge \{\delta \in \sigma(\overline{R},c) \mid \overline{R} \in dom(\sigma), \overline{R} \subseteq \overline{P}, c \ge b\}\,.$$

Леммы для мультимножества $\overline{P}$ включают в себя леммы подмножеств $\overline{P}$.

Мы сокращаем $[\![\pi]\!]_{U^b_\rho}$ и $[\![\pi]\!]_{O^b_\sigma}$ до $[\![\pi]\!]^b_\rho$ и $[\![\pi]\!]^b_\sigma$ соответственно. Для простоты положим значения интерпретаций на уровне -1 $U^{-1}_\rho(P) \stackrel{\text{def}}{=} \bot$ и $O^{-1}_\sigma(\overline{P}) \stackrel{\text{def}}{=} \bot$ для всех $P \in \mathcal{R}$ и $\overline{P} \in \mathbb{N}^{\mathcal{R}}$.

[1] Конъюнкция пустого множества — $\top$, дизъюнкция — $\bot$.

Алгоритм 1: Псевдокод RelRecMc

Вход : Проблема безопасности $\langle \mathcal{P}, \varphi_{safe} \rangle$
Выход: $out \in \langle$`БЕЗОПАСНО/НЕБЕЗОПАСНО`, реляционное доказательство/ контрпример$\rangle$

```
1  b ← 0; ρ ← ∅; σ ← ∅;
2  пока true
3      ⟨res, ρ, σ⟩ ← RelBndSafety(⟨P, φ_safe⟩, b, ρ, σ);
4      если res = ДОСТИЖИМО вернуть ⟨НЕБЕЗОПАСНО, _, ρ⟩;
5      иначе
6          inductive ← true;
7          для всех P̄ such that ⟨P̄, b⟩ ∈ dom(σ)
8              для всех δ ∈ σ(P̄, b)
9                  если ⊨ [[body(P̄)]]^b_σ ⇒ δ
10                     σ ← σ ∪ {⟨P̄, b + 1⟩ ↦ δ};
11                 иначе inductive ← false;
12         если inductive вернуть ⟨БЕЗОПАСНО, O^b_σ, _⟩;
13         b ← b + 1;
```

3.2. Внешний цикл

В алгоритме 1 представлен псевдокод процедуры RelRecMc, которая итеративно добавляет новые ветви в ρ и усиливает леммы σ до тех пор, пока либо ρ не засвидетельствует контрпример (строка 4), или σ не станет индуктивным (строка 12). Итерация номер b RelRecMc проверяет достижимость нарушения свойства за b шагов. Если нарушения не достижимы (строки 6 - 11), σ содержит доказательство безопасности всех выводов системы сверху-вниз высоты b. Затем RelRecMc распространяет все индуктивные леммы из σ на уровень $b+1$ и итерируется, если их не достаточно для доказательства безопасности системы в целом.

3.3. Внутренний цикл

Процедура RelBndSafety, представленная в алгоритме 2, проверяет безопасность всех выводов системы сверху-вниз высоты, ограниченной сверху B. Она формулирует и решает *запросы* $\langle \overline{P}, \pi, b \rangle$, где $\overline{P} \in \mathbb{N}^{\mathcal{R}}$, π — отрицание свойства безопасности для $\overline{P}$, and $b \in \mathbb{N}$. Неформально, ответ на $\langle \overline{P}, \pi, b \rangle$ состоит в определении того, достигает ли $\overline{P}$ π за b шагов.

Запросы помещаются в очередь Q, которая в начале содержит только $\langle P_0, \neg\varphi_{safe}, B \rangle$. Каждая итерация начинается с выбора запроса с наименьшим b (строка 3), на который может быть дан положительный (строка 12) или отрицательный (строка 8) ответ, либо который может породить дочерние запросы, на которые нужно ответить для ответа на данный запрос (строка 31). Когда на все запросы дан ответ, алгоритм возвращает результат `(НЕ-)ДОСТИЖИМО` (строка 34 или 32 соответственно).

Вывод ветвей Если π достижимо за один шаг от предшественников, ограниченных высотой $b-1$ (строка 4), алгоритм выводит по новой ветви для всех $\overline{P}[i]$ (строка 7). Неформально, вместо исследования всех ветвей $\overline{P}[i]$, алгоритм исследу-

Алгоритм 2: Псевдокод RelBndSafety

Вход : Проблема безопасности $\langle \Phi, \varphi_{safe} \rangle$, уровень B, хранилища ρ, σ
Выход : $out \in \langle$ДОСТИЖИМО/НЕДОСТИЖИМО$, \rho, \sigma \rangle$
Данные: Очередь запросов Q

$Q \leftarrow \{\langle P_0, \neg\varphi_{safe}, B\rangle\}$;
пока $(Q \neq \emptyset)$
  выбрать $\langle \overline{P}, \pi, b\rangle$ из Q;
  если $\exists m \,.\, m \models [\![body(\overline{P})]\!]_{\rho}^{b-1} \wedge \pi$
    для $i \leftarrow 1$ **до** $|\overline{P}|$
      $\psi_{\overline{P}[i]} \leftarrow \text{MBP}\left([\![body(\overline{P}[i])]\!]_{\rho}^{b-1}, \overline{\ell}_{\overline{P}[i]}, m \right)$;
      $\rho \leftarrow \rho \cup \{\langle \overline{P}[i], b\rangle \mapsto \psi_{\overline{P}[i]} \mid 1 \le i \le n\}$;
    $Q \leftarrow Q \setminus \{\langle \overline{R}, \eta, c\rangle \mid \overline{R} \subseteq \overline{P}, c \geq b, \bigwedge_{i=1}^{+|\overline{R}|} \psi_{\overline{R}[i]} \wedge \eta \not\Rightarrow \bot\}$;
  иначе если $[\![body(\overline{P})]\!]_{\sigma}^{b-1} \wedge \pi \Rightarrow \bot$
    пусть ψ такая, что $[\![body(\overline{P})]\!]_{\sigma}^{b-1} \Rightarrow \psi$ и $\psi \wedge \pi \Rightarrow \bot$;
    $\sigma \leftarrow \sigma \cup \{\langle \overline{P}, b\rangle \mapsto \psi\}$;
    $Q \leftarrow Q \setminus \left\{ \langle \overline{R}, \eta, c\rangle \mid \overline{P} \subseteq \overline{R}, c \leq b, \left[\!\!\left[\bigwedge_i \overline{R}[i](\overline{v}_{\overline{R}[i]}) \right]\!\!\right]_{\sigma}^{c} \wedge \eta \Rightarrow \bot \right\}$;
  иначе
    пусть $cti \models [\![body(\overline{P})]\!]_{\sigma}^{b-1} \wedge \pi$;
    $\psi \leftarrow \pi$;
    $apps \leftarrow \emptyset$;
    для $i \leftarrow 1$ **до** $|\overline{P}|$
      let $C \in rules(\overline{P}[i])$ be s.t. $cti \models [\![body(C)]\!]_{\sigma}^{b-1}$;
      $\eta \wedge R_1(\overline{x}_1) \wedge \ldots \wedge R_m(\overline{x}_m) \leftarrow body(C)$;
      $\psi \leftarrow \psi \wedge \eta$;
      для $j \leftarrow 1$ **до** m
        если $cti \models [\![R_j(\overline{x}_j)]\!]_{\rho}^{b-1}$
          $\psi \leftarrow \psi \wedge [\![R_j(\overline{x}_j)]\!]_{\rho}^{b-1}$;
        иначе $apps \leftarrow apps \cup \{R_j(\overline{x}_j)\}$;
    $Groups \leftarrow \text{Partition}(apps, \pi, \psi)$;
    для всех $\{i_1, \ldots, i_k\} \in Groups$
      $rels \leftarrow \langle R_{i_1}, \ldots, R_{i_k}\rangle$;
      $vars \leftarrow \overline{x}_{i_1} \ldots \overline{x}_{i_k}$;
      $\psi' \leftarrow \psi \wedge \left[\!\!\left[\bigwedge_{\substack{R_j(\overline{x}_j) \in apps \\ j \notin \{i_1, \ldots, i_k\}}} R_j(\overline{x}_j) \right]\!\!\right]_{\sigma}^{b-1}$;
      $\psi' \leftarrow \text{MBP}\left(\psi', \overline{v}_{\overline{P}} \cup \overline{\ell}_{\overline{P}} \setminus vars, cti\right)$;
      $Q \leftarrow Q \cup \{\langle rels, \psi'[vars \leftarrow \overline{v}_{rels}], b-1\rangle\}$;
если $[\![P_0]\!]_{\rho}^{n} \wedge \neg\varphi_{safe} \not\Rightarrow \bot$ **вернуть** $\langle$*ДОСТИЖИМО*$, \rho, \sigma\rangle$;
утвердить $[\![P_0]\!]_{\sigma}^{n} \wedge \neg\varphi_{safe} \Rightarrow \bot$;
вернуть $\langle$*НЕДОСТИЖИМО*$, \rho, \sigma\rangle$;

ет за раз только одну ветвь $\psi_{\overline{P}[i]}$, выбираемую в направлении к свойству. Каждый запрос $\langle \overline{R}, \eta, c\rangle \in Q$, где η достижимо в обновлённой интерпретации U_{ρ}^{c}, считается отвеченным и удаляется из Q (в частности, $\langle \overline{P}, \pi, b\rangle$).

Для получения символьного представления для ветви $\psi_{\overline{P}[i]}$, алгоритм исполь-

зует *проекцию на основе модели* (model-based projection, MBP) [3, 21]. По данной формуле $\exists \overline{x}.\tau$, где τ — бескванторная формула и модели m, $\text{MBP}(\tau, \overline{x}, m)$ строит бескванторную конъюнкцию литералов τ' такую, что $m \models \tau'$, $\tau' \Rightarrow \exists \overline{x}.\tau$, и если $\mathcal{M}$ допускает элиминацию кванторов, то для каждой формулы τ существует конечное множество моделей $m_1, \ldots, m_k$, что $\exists \overline{x}.\tau \Leftrightarrow \bigvee_{i=1}^{k} \text{MBP}(\tau, \overline{x}, m_i)$. Неформально, последовательность проекций на основе моделей *лениво* устраняют квантор из $\exists \overline{x}.\tau$. При данной m, $\text{MBP}\left(\left[\!\left[body(\overline{P}[i])\right]\!\right]_{\rho}^{b-1}, \overline{\ell}_{\overline{P}[i]}, m\right)$ может рассматриваться как выбор ветви $\overline{P}[i]$, выполняющей m, и устранение из неё всех локальных переменных $\overline{\ell}_{\overline{P}[i]}$. Ленивость в элиминации кванторов позволяет сохранять размер символьных представлений маленьким и позволяет рассматривать только релевантные поведения системы. В частности, хотя $\exists \overline{x}.\tau$ эквивалентно дизъюнкции ветвей, алгоритм рассматривает лишь одну ветвь за запрос; другие ветви будут рассмотрены по требованию в следующих итерациях.

Вывод лемм Если σ достаточно сильна для доказательства недостижимости π (строка 9), RelBndSafety выводит новую лемму вычислением Крейговского интерполянта [2] (далее обозначается ITP) формул $\left[\!\left[body(\overline{P})\right]\!\right]_{\sigma}^{b-1}$ и $\neg\pi$. В результате новая лемма аппроксимирует сверху семантику P глубины b, всё ещё доказывая её безопасность относительно π. Важно, что новая лемма сформулирована только над переменными $\overline{P}$, выражая *отношения* между элементами $\overline{P}$. Каждый запрос $\langle \overline{R}, \eta, c \rangle \in Q$ такой, что обновлённая σ доказывает невыполнимость η, считается отвеченным и удаляется из очереди (в частности, $\langle \overline{P}, \pi, b \rangle$).

Генерация запросов Если ни $\left[\!\left[body(\overline{P})\right]\!\right]_{\rho}^{b-1} \wedge \pi$ выполнимо, ни $\left[\!\left[body(\overline{P})\right]\!\right]_{\sigma}^{b-1} \wedge \pi$ невыполнимо, формулы в ρ на уровне $b-1$ слишком сильны для того, чтобы засвидетельствовать контрпример, а леммы на уровне $b-1$ слишком слабы для доказательства безопасности. В этом случае существует потенциальный контрпример (*контрпример к индуктивности* в терминах IC3 [22]) *cti*, получаемый алгоритмом на строке 14, который должен быть засвидетельствован ρ или заблокирован σ. Алгоритм порождает дочерние запросы, ответы на которые помогут ответить на $\langle \overline{P}, \pi, b \rangle$ либо доказательством недостижимости *cti*, либо свидетельством его достижимости.

Для каждого отношения в $\overline{P}$ алгоритм выбирает дизъюнкт C, выполняющийся в *cti* (*свидетельствующий cti*) (строка 18). Такой дизъюнкт гарантированно существует, т.к. $cti \models \left[\!\left[body(\overline{P})\right]\!\right]_{\sigma}^{b-1}$. Пусть $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ — все применения реляционных символов в C (строка 19). На следующих шагах алгоритм пытается усилить леммы, чтобы заблокировать *cti*. Для этого алгоритм определяет, какие леммы уровня $b-1$ слишком грубы, разделяя $R_1(\overline{x}_1), \ldots, R_m(\overline{x}_m)$ на две группы: применения, свидетельствующие *cti* (строка 22) и другие применения *apps* (строка 24). Т.к. ветви применений в первой группе уже свидетельствуют контрпример,

[2] Крейговский интерполянт двух формул A и B таких, что $A \wedge B$ невыполнимо, — это формула I, удовлетворяющая трём условиям: 1) $A \Rightarrow I$, 2) $I \wedge B$ невыполнимо, 3) I состоит только из общих для A и B символов.

не имеет смысла усиливать их леммы, поэтому алгоритм пытается усилить только леммы для *apps*.

На этом шаге алгоритм ведёт себя отличным от [3] образом. Вместо усиления лемм для каждого отношения в отдельности, он пытается вывести лемму для *группы* предикатов в *apps*.

Алгоритм параметризован оракулом Partition (строка 25), который разделяет входное множество атомов на список мультимножеств применений. Например, множество атомов $\{f(\overline{x_1}), f(\overline{x_2}), g(\overline{y})\}$ может быть поделено на группы $[\{f \mapsto 2\}, \{g \mapsto 1\}]$ отношений и список векторов $[\langle \overline{x}_1, \overline{x}_2 \rangle, \overline{y}]$ переменных. В результате список *rels* содержит все мультимножества символов, которые будут исследованы на в дочерних запросах.

В нашей реализации Partition разделяет применения на рекурсивную и нерекурсивную группы[3]. Если в рекурсивной группе более $\left|\overline{P}\right|$ элементов, она в свою очередь разбивается на подгруппы размера не более $\left|\overline{P}\right|$. Это останавливает неограниченный рост размеров мультимножеств запросов. Partition угадывает подходящее разбиение рекурсивных применений определений синхронизаций, сохраняющих индуктивность свойства в духе [17] (секция 3.4. демонстрирует его работу на примере 1).

Для каждого мультимножества в *rels* генерирует дочерний запрос, аппроксимирующий снизу множество ошибочных состояний группы. Свойство безопасности j-ой группы — конъюнкция родительского свойства безопасности π (строка 15) ограничений всех дизъюнктов, свидетельствующих *cti* (строка 20), достижимых дочерних состояний, свидетельствующих *cti* (строка 23) и лемм оставшихся мультимножеств в разбиении (строка 29). Неформально, RelBndSafety усиливает свойство безопасности π информацией о достижимых состояниях и дочерних леммах, связанных с *cti*. Далее, алгоритм проектирует свойство безопасности, устраняя все переменные, кроме $vars[j]$, используя проекцию на основе модели (строка 30), переименовывает переменные и помещает новый запрос в очередь (строка 31). Дочерние свойства формулируются над переменными дочерних отношений, для этого переменные переименовываются (строка 31). Аналогично выводу ветвей, использование проекции на основе моделей не портит корректность, поддерживая размер формул в запросах небольшим.

3.4. Пример

В этом разделе показаны несколько итераций алгоритма для задачи в примере 1. После синтаксической предобработки алгоритм работает со следующей системой, в которой переменные различных отношений различны:

[3] Два символа (взаимно) рекурсивны, если они принадлежат одной сильно связной компоненте в орграфе (V, E) с $V = \mathcal{R}$ и $(P, R) \in E$, если и только если R входит в тело некоторого правила для P.

$$
\begin{aligned}
T_1 = \mathit{leaf} \wedge n = 0 &\Rightarrow \mathit{size}(T_1, n)\\
T_1 = \mathit{node}(v_1, L_1, R_1) \wedge n = 1 + n^L + n^R \wedge & \\
\mathit{size}(L_1, n^L) \wedge \mathit{size}(R_1, n^R) &\Rightarrow \mathit{size}(T_1, n)\\
T_2 = \mathit{leaf} \wedge s_2 = 0 &\Rightarrow \mathit{sum}(T_2, s_2)\\
T_2 = \mathit{node}(v_2, L_2, R_2) \wedge s_2 = v_2 + s_2^L + s_2^R \wedge & \\
\mathit{sum}(L_2, s_2^L) \wedge \mathit{sum}(R_2, s_2^R) &\Rightarrow \mathit{sum}(T_2, s_2)\\
T_4 = \mathit{leaf} \wedge U = \mathit{leaf} &\Rightarrow \mathit{inc}(T_4, U)\\
T_4 = \mathit{node}(v_4, L_4, R_4) \wedge U = \mathit{node}(v_4 + 2, L', R') \wedge & \\
\mathit{inc}(L_4, L') \wedge \mathit{inc}(R_4, R') &\Rightarrow \mathit{inc}(T_4, U)\\
A = T_0 \wedge B = T_0 \wedge C = T_0 \wedge D = E \wedge \mathit{size}(A, n_0) \wedge & \\
\mathit{sum}(B, s_0) \wedge \mathit{inc}(C, D) \wedge \mathit{sum}(E, s_0') &\Rightarrow P_0(T_0, n_0, s_0, s_0')\\
\varphi_{\mathit{safe}} &\stackrel{\text{def}}{=} s_0' = s_0 + 2n_0
\end{aligned}
$$

Уровень *0* Алгоритм начинает с вызова RelBndSafety для уровня 0, который кладёт запрос $\langle P_0, s_0' \neq s_0 + 2n_0, 0\rangle$ в Q. И $[\![\mathit{body}(P_0)]\!]_\rho^{-1}$, и $[\![\mathit{body}(P_0)]\!]_\sigma^{-1}$ ложны, поэтому алгоритм попадает на строку 11, где добавляется $\mathrm{ITP}(\bot, \neg\varphi_{\mathit{safe}}) = \bot$ в $\sigma(P_0, 0)$. RelBndSafety завершается с результатом *UNREACHABLE*, но т.к. добавленная лемма не индуктивна, RelRecMc продолжает свою работу на уровне 1.

Уровень *1* RelBndSafety начинает с $Q = \{\langle P_0, s_0' \neq s_0 + 2n_0, 1\rangle\}$. Здесь $[\![\mathit{body}(P_0)]\!]_\rho^0 \equiv \bot$, $[\![\mathit{body}(P_0)]\!]_\sigma^0 \equiv \varphi_0 \stackrel{\text{def}}{=} A = T_0 \wedge B = T_0 \wedge C = T_0 \wedge D = E$. Т.к. $\varphi_0 \wedge \neg\varphi_{\mathit{safe}}$ выполнима, алгоритм получает $\mathit{cti} = \{A, B, C, D, E, T_0 \mapsto \mathit{leaf}; n_0 \mapsto 1; s_0 \mapsto 0; s_0' \mapsto 1\}$ и переходит на строку 14. Затем выбирается единственное возможное правило для P_0 с телом $\varphi_0 \wedge \mathit{size}(A, n_0) \wedge \mathit{sum}(B, s_0) \wedge \mathit{inc}(C, D) \wedge \mathit{sum}(E, s_0')$. Теперь $\mathit{cti} \not\models [\![\mathit{size}(A, n_0)]\!]_\rho^0 \equiv A = \mathit{leaf} \wedge n_0 = 0$, $\mathit{cti} \models [\![\mathit{sum}(B, s_0)]\!]_\rho^0$, $\mathit{cti} \models [\![\mathit{inc}(C, D)]\!]_\rho^0$, $\mathit{cti} \not\models [\![\mathit{sum}(E, s_0')]\!]_\rho^0$, откуда $\mathit{apps} = \{\mathit{size}(A, n_0), \mathit{sum}(E, s_0')\}$ и $\psi \equiv \neg\varphi_{\mathit{safe}} \wedge \varphi_0 \wedge B = \mathit{leaf} \wedge s_0 = 0 \wedge C = \mathit{leaf} \wedge D = \mathit{leaf}$. Т.к. и *inc*, и *sum* не рекурсивны с P_0, Partition ничего не делает: $\textsc{Partition}(\mathit{apps}) = \{\{1, 4\}\}$.

Обозначим $\alpha_1 = \{\mathit{size} \mapsto 1, \mathit{sum} \mapsto 1\}$. Для получения запроса для α_1, алгоритм строит проекцию ψ, устраняя все переменные, кроме A, n_0, E, и s_0', получая $\psi' \equiv \mathrm{MBP}(\psi, \{T_0, B, C, D, s_0\}, \mathit{cti}) \equiv A = \mathit{leaf} \wedge E = \mathit{leaf} \wedge s_0' \neq 2n_0$, которая после переименования переменных становится $\psi_1 \stackrel{\text{def}}{=} T_1 = \mathit{leaf} \wedge T_2 = \mathit{leaf} \wedge s_2 \neq 2n$. Таким образом, Q становится $\{\langle P_0, \neg\varphi_{\mathit{safe}}, 1\rangle, \langle \alpha_1, \psi_1, 0\rangle\}$, и алгоритм уходит на следующую итерацию.

На второй итерации алгоритм выбирает из очереди $\langle \alpha_1, \psi_1, 0\rangle$ как запрос с наименьшим уровнем. Пусть $\beta_1 \stackrel{\text{def}}{=} [\![\mathit{body}(\alpha_1)]\!]_\rho^{-1} \equiv [\![\mathit{body}(\alpha_1)]\!]_\sigma^{-1} \equiv T_1 = \mathit{leaf} \wedge n = 0 \wedge T_2 = \mathit{leaf} \wedge s_2 = 0$. Т.к. $\beta_1 \wedge \psi_1$ не выполнима, алгоритм выводит новую лемму $\delta_1 \stackrel{\text{def}}{=} \mathrm{ITP}(\beta_1, \psi_1) \equiv T_1 = T_2 = \mathit{leaf} \wedge n = s_2 = 0$. Таким образом, $\sigma(\alpha_1, 0) = \{\delta_1\}$, запрос на уровне 0 отвечен и убирается из Q.

На третьей итерации алгоритм снова выбирает $\langle P_0, \neg\varphi_{\mathit{safe}}, 1\rangle$. На этот раз $[\![\mathit{body}(P_0)]\!]_\sigma^0 \equiv \varphi_0 \wedge \delta_1(A, n, B, s_0) \wedge \delta_1(A, n, E, s_0')$. Т.к. теперь $[\![\mathit{body}(P_0)]\!]_\sigma^0 \wedge \neg\varphi_{\mathit{safe}}$ не выполнима, в $\sigma(P_0, 1)$ добавляется $\mathrm{ITP}([\![\mathit{body}(P_0)]\!]_\sigma^0, \neg\varphi_{\mathit{safe}}) \equiv \varphi_{\mathit{safe}}$. Новое окружение не индуктивно, поэтому RelRecMc переходит на уровень 2.

Уровень 2 Запрос $\langle P_0, \neg\varphi_{safe}, 2\rangle$ выбирается из Q. На этот раз $cti = \{A, B, C, T_0 \mapsto node(0,leaf,leaf); D, E \mapsto leaf; n_0, s_0, s'_0 \mapsto 1\}$. Т.к. $cti \not\models [\![size(A,n_0)]\!]^1_\rho$, $cti \not\models [\![sum(B,s_0)]\!]^1_\rho$, $cti \not\models [\![inc(C,D)]\!]^1_\rho$, $cti \not\models [\![sum(E,s'_0)]\!]^1_\rho$, получаем $apps = \{size(A,n_0), sum(B,s_0), inc(C,D), sum(E,s'_0)\}$. Т.к. ни один из символов не рекурсивен с P_0, получаем $Groups = \{\{1,2,3,4\}\}$. Для получения дочернего свойства безопасности алгоритм устраняет T_0 проекцией и уходит на следующую итерацию с $\{\langle P_0, \neg\varphi_{safe}, 2\rangle, \langle\alpha_2, \psi_2, 1\rangle\}$, где $\alpha_2 \stackrel{\text{def}}{=} \{size \mapsto 1, sum \mapsto 2, inc \mapsto 1\}$ и $\psi_2 \stackrel{\text{def}}{=} T_1 = T_2 = T_4 \wedge U = T_3 \wedge s_3 \neq s_2 + 2n$.

На следующей итерации $\langle\alpha_2, \psi_2, 1\rangle$ выбирается из Q. RelRecMc применяет подход, описанный в Секции 2.4. для эффективного вычисления подстановки σ. Положим

$$\begin{aligned}
\beta_{size} &\stackrel{\text{def}}{=} (T_1 = leaf \wedge n = 0) \vee \\
&\quad (T_1 = node(v_1,L_1,R_1) \wedge n = 1 + n^L + n^R \wedge a_L \wedge a_R) \\
\beta_{sum_1} &\stackrel{\text{def}}{=} (T_2 = leaf \wedge s_2 = 0) \vee \\
&\quad (T_2 = node(v_2,L_2,R_2) \wedge s_2 = v_2 + s_2^L + s_2^R \wedge b_L \wedge b_R) \\
\beta_{sum_2} &\stackrel{\text{def}}{=} (T_3 = leaf \wedge s_3 = 0) \vee \\
&\quad (T_3 = node(v_3,L_3,R_3) \wedge s_3 = v_3 + s_3^L + s_2^R \wedge c_L \wedge c_R) \\
\beta_{inc} &\stackrel{\text{def}}{=} (T_4 = leaf \wedge U = leaf) \vee \\
&\quad (T_4 = node(v_4,L_4,R_4) \wedge U = node(v_4 + 2, L', R') \wedge d_L \wedge d_R)
\end{aligned}$$

Здесь a_L, a_R, b_L, b_R, c_L, c_R, d_L и d_R — пропозициональные абстракции применений реляционных символов. Тогда:

$$\begin{aligned}
[\![body(\alpha_2)]\!]^0_\sigma \equiv\ & \beta_{size} \wedge \beta_{sum_1} \wedge \beta_{sum_2} \wedge \beta_{inc} \wedge \\
& (a_L \wedge b_L \Rightarrow \delta_1(L_1, n_L, L_2, s_2^L)) \wedge \\
& (a_L \wedge b_R \Rightarrow \delta_1(L_1, n_L, R_2, s_2^R)) \wedge \\
& (a_L \wedge c_L \Rightarrow \delta_1(L_1, n_L, L_3, s_3^L)) \ldots
\end{aligned}$$

Если вместо этого прямолинейно перевести формулу $[\![body(\alpha_2)]\!]^0_\sigma$ в ДНФ и заменить всевозможные комбинации применений реляционных символов формулой δ_1, мы получим формулу в 2^4 раз большего размера.

$[\![body(\alpha_2)]\!]^0_\sigma \wedge \psi_2$ выполнима с

$$cti = \{T_k \mapsto node(1,leaf,leaf); T_3, U \mapsto node(3,leaf,leaf; \ldots\}$$

для $k \in \{1,2,4\}$. Таким образом, на строке 18 алгоритм выбирает по второму (рекурсивному) правилу для каждого отношения в $\mathcal{R}$.

Предположим теперь, что cti не выполняет ни одну из ветвей в ρ ни одного применения в телах правил. Тогда получаем $apps = \{size(L_1,n^L), size(R_1,n^R), sum(L_2,s_2^L), sum(R_2,s_2^R), sum(L_3,s_3^L), sum(R_3,s_3^R), inc(L_4,L'), inc(R_4,R')\}$ где все применения рекурсивны, и $\psi \equiv \psi_2 \wedge T_1 = node(v_1,L_1,R_1) \wedge n = 1 + n^L + n^R \wedge \ldots$.

Если Partition вернет единственную группу из всех применений, мы получим запрос для мультимножества размера 8, что в результате может породить запрос

для 16 отношений и т.д.; в результате, RelBndSafety расходится. Чтобы контролировать размер мультимножеств, Partition разбивает *apps* на группы размера, не превосходящего $|\alpha_2| = 4$. Однако существует $C_8^4 = 70$ вариантов уже для разбиения 8 отношений на две группы размера 4. Для выбора наилучшего разбиения, Partition применяет следующую эвристику.

Так как формула свойства безопасности в каждом запросе в Q (кроме корневого) есть результат проекции, она представляет собой конъюнкцию литералов. Для каждого подмножества *apps*, PARTITION определяет *максимальное индуктивное подмножество* литералов.

В данном случае множество литералов в ψ_2 является $\{T_1 = T_2, T_1 = T_4, U = T_3, s_3 \neq s_2 + 2n\}$. Например, литерал $T_1 \wedge T_2$ индуктивен относительно $size(R_1,n^R)$ и $sum(R_2,s_2^R)$. Чтобы убедиться в этом, переименуем $T_1 = T_2$ в $R_1 = R_2$ (т.к. T_1, R_1 и T_2, R_2 — первые аргументы соответствующих применений *size* и *sum*) и заметим, что имеет место $\psi \Rightarrow R_1 = R_2$. В нашем случае, вся формула ψ_2 индуктивна относительно групп $\{size(L_1,n^L), sum(L_2,s_2^L), sum(L_3,s_3^L), inc(L_4,L')\}$ и $\{size(R_1,n^R), sum(R_2,s_2^R), sum(R_3,s_3^R), inc(R_4,R')\}$, поэтому Partition возвращает $Groups = \{\{1,3,5,7\},\{2,4,6,8\}\}$. Для обеих групп в Q добавляются запросы α_2 уровня 0.

На следующих итерациях, алгоритм выводит лемму $T_1 = T_2 = T_4 \wedge U = T_4 \Rightarrow s_3 = s_2 + 2n$ и завершает построение реляционного доказательства безопасности.

3.5. Общие свойства

В этой секции утверждаются важные свойства RelRecMc и RelBndSafety.

Теорема 2. *Алгоритм* RelRecMc *корректен, т.е. если он останавливается для проблемы* $\langle \mathcal{P}, \varphi_{safe} \rangle$*, то она безопасна, тогда и только тогда, когда алгоритм вернул* `БЕЗОПАСНО`*.*

Теорема 3. *Алгоритм* RelBndSafety *корректен, т.е. если он возвращает для проблемы* $\langle \mathcal{P}, \varphi_{safe} \rangle$ *на уровне* B `НЕДОСТИЖИМО`*, тогда и только тогда, когда все выводы системы сверху-вниз высоты, ограниченной сверху* B*, безопасны.*

Оба алгоритма проверяют на выполнимость формулы языка ограничений (например, в строке 9 RelRecMc или в строке 4 RelBndSafety). Назовём оракулом выполнимости в $\mathcal{M}$ процедуру, которая для любой формулы языка ограничений φ за один шаг гарантированно проверяет $\mathcal{M} \models \varphi$. Также подразумевается, что такой оракул способен выдавать модели формул, Крейговские интерполянты и проектировать формулы на основе моделей; на практике таким оракулом является SMT-решатель.

Теорема 4. RelBndSafety *полна относительно оракула выполнимости в* $\mathcal{M}$*, т.е. при наличии оракула выполнимости в* $\mathcal{M}$ *для любой проблемы безопасности* $\langle \mathcal{P}, \varphi_{safe} \rangle$*, уровня* $B \in \mathbb{N}$ *и пустых* σ *и* ρ*,* RelBndSafety *останавливается и возвращает* `ДОСТИЖИМО`*, либо* `НЕДОСТИЖИМО`*.*

Теоремы, аналогичные 2, 3 и 4 доказаны в Теоремах 1 и 2 работы [3]. Для того, чтобы превратить доказательство, представленное в [3], достаточно заменить

все интерпретации и семантики символов на их реляционные аналоги. Например, инвариант традиционного алгоритма

$$\text{для всех } P \in \mathcal{R}, [\![P]\!]^b_{\mathcal{M}} \subseteq \mathcal{M}(O^b_\sigma(P))$$

становится инвариантом RelRecMc

$$\text{для всех } \langle P_1, \ldots, P_n\rangle \in dom(\sigma), [\![P_1]\!]^b_{\mathcal{M}} \times \cdots \times [\![P_n]\!]^b_{\mathcal{M}} \subseteq \mathcal{M}(O^b_\sigma(\langle P_1, \ldots, P_n\rangle)).$$

Теорема 5. *При наличии оракула выполнимости в $\mathcal{M}$* RelRecMc *является коразрешающей процедурой проблем безопасности дизъюнктов, т.е. если $\mathcal{P}$ небезопасно, процедура гарантированно остановится и вернёт* НЕБЕЗОПАСНО.

Доказательство. Любая небезопасная система дизъюнктов имеет резолютивное опровержение с деревом вывода некоторой конечной высоты. Пусть H — наименьшая высота среди всех таких деревьев. Т.к. алгоритм RelRecMc итеративно увеличивает глубину b, то по корректности и относительной полноте RelBndSafety все вызовы RelBndSafety в строке 3 остановятся и вернут `НЕДОСТИЖИМО` (в противном случае, существует резолютивное опровержение меньшей глубины). На шаге $b = H$ остановится RelBndSafety и вернёт ответ НЕБЕЗОПАСНО. □

Для систем с конечным пространством состояний RelRecMc — полная разрешающая процедура; для них алгоритм полиномиален от количества состояний.

Если при выполнении строки 25 алгоритма 2 *Groups* содержит только единичные мультимножества, поведение RelBndSafety совпадает с поведением BndSafety из [3]. Другими словами, представленный алгоритм ведёт себя аналогично BndSafety на *линейных* системах дизъюнктов и *обобщает* его поведение на нелинейных системах: если Partition разбивает входные применения на группы размера 1, то алгоритм выводит "классические" доказательства безопасности аналогично оригинальному алгоритму.

В некоторых случаях, когда RecMC [3] не может вывести доказательство безопасности нелинейных систем, т.к. каждая теоретико-множественная модель не представима в языке ограничений, наш алгоритм справляется с поиском реляционного доказательства безопасности. Напротив, если RecMC успешно доказывает или опровергает безопасность системы, наш алгоритм также справляется.

4. Эксперименты

Алгоритм был реализован над Spacer, современным решателем дизъюнктов Хорна в SMT-решателе Z3 [23][4]. Мы сравнили реализацию со Spacer и HoIce [8] на двух наборах тестов[5]. Эксперименты были проведены на компьютере под управлением ОС Arch Linux с процессором Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz.

Первый набор тестов содержит 840 проблем из [8], не являющихся проблемами реляционной верификации. Мы сравнили наш алгоритм со Spacer и HoIce и показали его жизнеспособность. Таймаут для этих тестов был 30 секунд. Spacer решил

[4]Реализация доступна по ссылке https://github.com/dvvrd/z3.
[5]Тесты доступны по ссылке https://github.com/dvvrd/spacer-benchmarks.

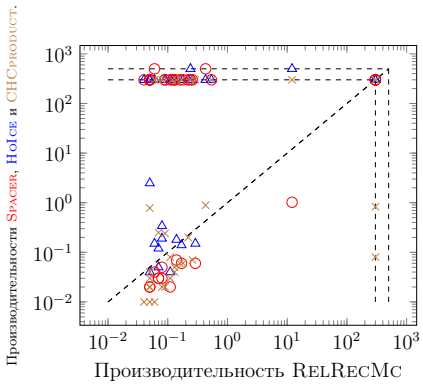

Рис. 1. Сравнение RelRecMc с другими решателями. Каждая точка графика представляет пару времён исполнения теста (сек. × сек.) алгоритмом RelRecMc (ось x) и другим алгоритмом (ось y). Таймауты указаны внутренней пунктирной линией; на внешних пунктирных линиях лежат тесты, на которых решатели завершились с ошибкой времени исполнения.
Fig. 1. RelRecMc vs competitors. Each point in a plot represents a pair of the run times (sec × sec) of RelRecMc (x-axis) and a competitor (y-axis). Timeouts are placed on the inner dashed lines; and crashes are on the outer dashed lines.

788 из 840 проблем с 50 таймаутами и 2 ошибками времени исполнения. Наша реализация решила 807 проблем с 34 таймаутами. Накладные расходы по времени в сравнении со Spacer незначительны (менее 0.1 секунды на 87% проблем). Наша реализация решила большинство проблем, решённых Spacer. Тем не менее, 10 были решены Spacer, но не нашей реализацией; мы объясняем это несовершенством текущей реализации. HoIce решил 808 проблем с 26 таймаутами и 6 ошибками времени исполнения, но и Spacer, и наша реализация RelRecMc затратила меньше времени, чем HoIce на решённых проблемах.

Второй набор тестов содержит 37 проблем реляционной верификации, взятых из [24]. Мы сравнили нашу реализацию со Spacer, HoIce и алгоритмом CHCproduct из [17], реализованным как синтаксическое преобразование входной системы дизъюнктов с последующим решением преобразованной системы решателем Spacer. Схематичное сравнение показано на рис. 1.

Spacer и HoIce решили только 11 из 37 задач с 5-минутным таймаутом. CHCproduct решил 24 задачи, а RelRecMc решил 32 задачи из 37. Важно, что RelRecMc решил задачи, с которыми не справился Spacer после явного синтаксического слияния дизъюнктов. Для больших небезопасных проблем, например, `point-location*`, CHCproduct порождает экспоненциальное количество правил, расходуя всё отведённое время, в то время как другие решатели находят контрпример за секунды.

5. Обзор литературы

Большинство подходов к реляционной верификации основаны на автоматическом или интерактивном анализе программ-произведений [9–12,14–18,25]. Все такие под-

ходы трактуют верификатор функциональных спецификаций как чёрный ящик. Поэтому они вынуждены предопределять стратегию синхронизации. Напротив, наш подход не строит произведение программ явно, а использует SMT-решатель для построения и стратегий синхронизации, и реляционных инвариантов одновременно.

Декартова логика Хоара [26, 27] для доказательств k-свойств безопасности состоит из множества правил и эвристик для выравнивания циклов сравниваемых программ. Они анализируют условия циклов, ветвления, реляционные пред- и постусловия. Для выбора стратегии синхронизации в работе [27] определяется максимальное множество циклов, завершение каждого из которых влечёт завершение других (иначе реляционные инварианты не обнаруживаются, даже если они существуют). Напротив, наш подход не учитывает завершения циклов и может строить реляционные инварианты для циклов с различными количествами итераций.

Существуют подходы к трансформации нелинейных систем дизъюнктов, которые используют существующие решатели для автоматического построения реляционных инвариантов [16, 17]. Преобразование CHCprodcut [17] строит декартово произведение множества правил системы дизъюнктов. Когда правила перемножаемых реляционных символов имеют более одного рекурсивного применения в теле, преобразование CHCprodcut определено не единственным образом. Для этого применяется расширенная техника *синхронного произведения*, которая пытается выбрать произведение, соединяющее структурно схожие рекурсивные применения. Альтернативно, [16] предлагает трансформацию, основанную на известном подходе fold/unfold. Хотя результирующая система дизъюнктов решается проще, сложность трансформации растёт экспоненциально от числа перемноженных предикатных символов. Для сравнения, наш подход обрабатывает рекурсивные группы по требованию, используя модели, полученные от SMT-запросов, и поэтому не ведёт к экспоненциальному взрыву в сложности.

Недавний подход [18] наиболее близок к данной работе. Он анализирует контрпримеры для определения нетривиальных стратегий синхронизации, но использует заданное пользователем множество предикатов для построения реляционных инвариантов. Наш подход не требует предикатов и строит инварианты интерполяцией, эффективно используя преимущества, унаследованные от [3]. В будущем мы планируем поддержать определение других стратегий синхронизации.

6. Заключение и дальнейшие планы

В статье был представлен новый подход к решению нелинейных систем дизъюнктов Хорна, основанный на направляемой свойством достижимости. Его ключевая особенность состоит в автоматическом выводе реляционных инвариантов, аппроксимирующих сверху семантику групп неинтерпретированных символов. Подход способен автоматически определять, какие предикаты должны быть сгруппированы, а для каких будет достаточно вывести индивидуальные инварианты. Подход был реализован поверх решателя Spacer, была показана его практическая полезность на тестовых наборах различных задач реляционной верификации.

В будущем планируется разработка подхода к автоматическому определению

нетривиальных стратегий синхронизации, ищущего различные раскрутки частей программ-произведений; это даст возможность более гибкого поиска реляционных инвариантов.

Список литературы / References

[1] Krystof H., Bjørner N, "Generalized Property Directed Reachability", *LNCS, Springer*, **7317** (2012), 157–171.

[2] Cimatti A., Griggio A., Mover S., Tonetta S., "IC3 Modulo Theories via Implicit Predicate Abstraction", *LNCS, Springer*, **8413** (2014), 46–61.

[3] Komuravelli A., Gurfinkel A., Chaki S., "SMT-Based Model Checking for Recursive Programs", *Formal Methods in System Design*, **48**:3 (2016), 175–205.

[4] Jovanović D., Dutertre B., "Property-Directed k-induction", *FMCAD,IEEE*, 2016, 85–92.

[5] Fedyukovich G., Bodík R., "Accelerating Syntax-Guided Invariant Synthesis", *LNCS, Springer*, **10805** (2018), 251–269.

[6] Непомнящий В.А. и др., "Ориентированный на верификацию язык C-light", *Системная информатика: Сб. науч. тр. РАН. Сиб. отд-ние. Ин-т систем информатики*, **9** (2004), 51–134; [Nepomniaschy V.A. et al., "Verification oriented language C-light", *Sistemnaya Informatika: Sb. Nauch. Tr. RAN. Sib. Otd-nie. In-t Sistem Informatiki*, **9** (2004), 51–134, (in Russian).]

[7] Мандрыкин М. У., Мутилин В. С., Хорошилов А. В., "Введение в метод CEGAR — уточнение абстракции по контрпримерам", *Труды Института системного программирования РАН*, **24** (2013); [Mandrykin M. U., Mutilin V. S., Khoroshilov A. V., "Introduction to CEGAR — Counter-Example Guided Abstraction Refinement", *Proceedings of ISP RAS*, **24** 2013.]

[8] Champion A., Kobayashi N., Sato R., "HoIce: An ICE-Based Non-linear Horn Clause Solver", *Asian Symposium on Programming Languages and Systems, Springer*, 2018, 146–156.

[9] Lahiri S. K., McMillan K. L., Sharma R., Hawblitzel C., "Differential Assertion Checking", *FSE, ACM*, 2013, 345–355.

[10] Almeida J. B., Barbosa M., Barthe G., Dupressoir F., Emmi M., "Verifying Constant-Time Implementations", *USENIX*, 2016, 53–70.

[11] Kiefer M., Klebanov V., Ulbrich M., "Relational Program Reasoning Using Compiler IR", *LNCS, Springer*, **9971** (2016), 149–165.

[12] Beckert B., Bingmann T., Kiefer M., Sanders P., Ulbrich M., Weigl A., "Relational Equivalence Proofs Between Imperative and MapReduce Algorithms", *LNCS, Springer*, **11294** (2018), 248–266.

[13] Athanasiou K. et al., "SideTrail: Verifying Time-Balancing of Cryptosystems", *LNCS, Springer*, **11294** (2018), 215–228.

[14] Barthe G., Crespo J. M., Kunz C., "Relational Verification Using Product Programs", *LNCS, Springer*, **6664** (2011), 200–214.

[15] Felsing D. et al., "Automating Regression Verification", *ASE, ACM*, 2014, 349–360.

[16] De Angelis E. et al., "Relational Verification Through Horn Clause Transformation", *LNCS, Springer*, **9837** (2016), 147–169.

[17] Mordvinov D., Fedyukovich G., "Synchronizing Constrained Horn Clauses", *EPiC Series in Computing, EasyChair*, **46** (2017), 338–355.

[18] Shemer R., Gurfinkel A., Shoham S., Vizel Y., "Property Directed Self Composition", *LNCS, Springer*, **11561** (2019), 161–179.

[19] Clarke E. M., "Program Invariants as Fixedpoints", **21**:4 (1979), 273–294.

[20] Apt K. R., *From Logic Programming to Prolog*, Prentice Hall London, 1997.

[21] Bjørner N., Janota M., "Playing with Quantified Satisfaction", *LPAR (short papers)*, **35** (2015), 15–27.

[22] Bradley A. R., "SAT-Based Model Checking without Unrolling", *LNCS, Springer*, **6538** (2011), 70–87.

[23] De Moura L., Bjørner N., "Z3: An Efficient SMT Solver", *LNCS, Springer*, **4963** (2008), 337–340.

[24] Mordvinov D., Fedyukovich G., "Verifying Safety of Functional Programs with Rosette/Unbound", *CoRR.*, **abs/1704.04558.** (2017).

[25] Strichman O., Veitsman M., "Regression Verification for Unbalanced Recursive Functions", *LNCS*, **9995** (2016), 645–658.

[26] Sousa M., Dillig I., "Cartesian Hoare Logic for Verifying k-safety Properties", *PLDI, ACM*, 2016, 57–69.

[27] Pick L., Fedyukovich G., Gupta A., "Exploiting Synchrony and Symmetry in Relational Verification", *LNCS, Springer*, **10981** (2018), 164–182.

Mordvinov D. A., "Property-Directed Inference of Relational Invariants", *Modeling and Analysis of Information Systems*, **26**:4 (2019), 550–571.

DOI: 10.18255/1818-1015-2019-4-550-571

Abstract. Property Directed Reachability (PDR) is an efficient and scalable approach to solving systems of symbolic constraints also known as Constrained Horn Clauses (CHC). In the case of non-linear CHCs, which may arise, e.g., from relational verification tasks, PDR aims to infer an inductive invariant for each uninterpreted predicate. However, in many practical cases this reasoning is not successful, as invariants should be derived for groups of predicates instead of individual predicates. The article describes a novel algorithm that identifies these groups automatically and complements the existing PDR technique. The key feature of the algorithm is that it does not require a possibly expensive synchronization transformation over the system of CHCs. We have implemented the algorithm on top of a up-to-date CHC solver Spacer. Our experimental evaluation shows that for some CHC systems, on which existing solvers diverge, our tool is able to discover relational invariants.

Keywords: relational verification, constrained horn clauses, property-directed reachability, relational invariants

On the authors:
Dmitry A. Mordvinov, orcid.org/0000-0002-6437-3020, senior reseacher,
St Petersburg University and JetBrains Research,
28 Universitetskiy pr., Petrodvorets 198504, Russia, e-mail: dmitry.mordvinov@jetbrains.com

Acknowledgments:
This work was supported by JetBrains Research.