Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРИМЕНЕНИЕ ТЕОРИИ АЛГЕБР КЛИНИ И ЕЕ РАСШИРЕНИЙ ДЛЯ АВТОМАТИЗАЦИИ ДОКАЗАТЕЛЬСТВ В СИСТЕМЕ COQ

Автор: Головин Павел Андреевич _____

Направление подготовки: 01.03.02 Прикладная математика и информатика

Квалификация: Бакалавр

Руководитель ВКР: Чивилихин Д. С., к.т.н. _____

Санкт-Петербург, 2020 г.

Обучающийся Головин Павел Андреевич
Группа М3439 Факультет ИТиП

Направленность (профиль), специализация
Математические модели и алгоритмы в разработке программного обеспечения

Консультанты:

а) Моисеенко Е. А., без степени, без звания _____

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ

Руководитель ОП

проф., д.т.н. Парфенов В.Г. _____

«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Головин Павел Андреевич

Группа М3439 **Факультет** ИТиП

Квалификация: Бакалавр

Направление подготовки: 01.03.02 Прикладная математика и информатика

Направленность (профиль) образовательной программы: Математические модели и алгоритмы в разработке программного обеспечения

Тема ВКР: Применение теории Алгебр Клини и ее расширений для автоматизации доказательств в системе Coq

Руководитель Чивилихин Д. С., к.т.н., научный сотрудник Университета ИТМО

2 Срок сдачи студентом законченной работы до: «____» _____ 20___ г.

3 Техническое задание и исходные данные к работе

Требуется исследовать применимость теории алгебр Клини и ее расширений для автоматизации доказательств в системе Coq. В частности для доказательств, связанных с моделями памяти. Требуется изучить существующие реализации алгебр Клини в Coq и проанализировать их применение на реальных доказательствах, связанных с моделями памяти.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

а) Изучение алгебр Клини на предмет использования в доказательствах, связанных с моделями памяти.

б) Анализ существующих реализаций алгебр Клини в Coq.

в) Применение алгебры Клини к реальным доказательствам, связанным с моделями памяти.

г) Оценка результатов.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

а) Brunet P. Algebras of Relations : from algorithms to formal proofs : Theses : 2016LYSE1198 / Brunet Paul. — Université de Lyon, 10/2016.

б) Pous D. Kleene Algebra with Tests and Coq Tools for while Programs // Interactive Theorem Proving / ed. by S. Blazy, C. Paulin-Mohring, D. Pichardie. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. — P. 180–196. — ISBN 978-3-642-39634-2.

в) Repairing Sequential Consistency in C/C++11 / O. Lahav [et al.] // Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. — Barcelona, Spain : Association for Computing Machinery, 2017. — P. 618–632. — (PLDI 2017). — ISBN 9781450349888. — DOI: 10.1145/3062341.3062352.

7 Дата выдачи задания «____» _____ 20___ г.

Руководитель ВКР _____

Задание принял к исполнению _____ «____» _____ 20___ г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся: Головин Павел Андреевич

Наименование темы ВКР: Применение теории Алгебр Клини и ее расширений для автоматизации доказательств в системе Coq

Наименование организации, в которой выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Исследовать перспективность использования теории алгебры Клини и ее расширений для упрощения формальных доказательств в системе Coq, связанных с моделями памяти.

2 Задачи, решаемые в ВКР:

а) Применение одной из существующих реализаций алгебры Клини к реальным доказательствам, связанным с моделями памяти.

б) Оценить количество доказательств, которые удалось автоматизировать.

3 Число источников, использованных при составлении обзора: 23

4 Полное число источников, использованных в работе: 25

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	7	7	11

6 Использование информационных ресурсов Internet: да, число ресурсов: 1

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Раздел работы**
Система интерактивных доказательств Coq	1.2, 2,3
Пакет с формализацией алгебр Клини relation-algebra	2, 3
Коллекция доказательств свойств бинарных отношений и списков hahn	2, 3
LaTeX	1, 2, 3

8 Краткая характеристика полученных результатов

Были исследованы различные расширения теории алгебр Клини, а также их реализации в Coq. Самая полная из существующих на сегодняшний день была испытана на реальных доказательствах, связанных с моделями памяти. Эта реализация формализует алгебры Клини с тестами, благодаря которым удалось автоматизировать существенное количество доказательств, что говорит о перспективность данного метода.

9 Гранты, полученные при выполнении работы

Нет

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы

Нет

Обучающийся Головин П. А. _____

Руководитель ВКР Чивилихин Д. С. _____

«____» _____ 20___ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня компьютерные технологии все чаще используются в медицине, энергетике, финансах и машиностроении, где цена ошибки в коде очень высока и даже может стоить человеческих жизней. Это заставляет обратить пристальное внимание на корректность программ.

За последние годы был достигнут большой прогресс в области построения систем интерактивного доказательства теорем, которые позволяют формально верифицировать программы. Пользователь такой системы должен сначала формализовать программу и ее спецификацию, а потом с помощью специального языка написать доказательство, которое система проверит на корректность. Примерами таких систем являются Coq [21], Agda [20], Isabelle/HOL [19], Idris [3].

Coq среди этих инструментов имеет наиболее развитую экосистему и большое сообщество. Он часто используется для создания верифицированных систем, связанных с компиляцией языков программирования [12, 17] и, в частности, для формализации слабых моделей памяти [22, 24].

Модель памяти языка программирования призвана специфицировать поведения параллельных программ в присутствии гонок по данным. Простые модели памяти позволяют себе просто запрещать гонки или гарантируют последовательную согласованность всех операций (событий) в программе.

Но такой подход сильно ограничивает параллелизм и мешает компилятору оптимизировать код, от чего сильно страдает производительность программ. Поэтому на сегодняшний день все языки используют более слабые ограничения (слабые модели памяти).

Такие модели часто используют бинарные отношения на элементарных событиях чтения и записи, например, отношение синхронизации на атомарных переменных «synchronized with» [24], чтобы выделить только конфликтующие операции и наложить ограничения на возможные исполнения программы.

Проблема заключается в том, что создание формальной слабой модели памяти, которая корректно применима при компиляции программы под разные архитектуры - очень сложная задача, требующая объемных доказательств с аккуратным рассмотрением большого количества случаев. Поэтому при разработке современных моделей памяти стараются использовать системы интер-

активного доказательства теорем [24], чтобы упростить написание и проверку доказательств с помощью компьютера.

Так как, работая со слабыми моделям памяти, мы часто имеем дело с бинарными отношениями, то можно попробовать обобщить и вынести (автоматизировать) доказательства утверждений, связанных с их свойствами. Это должно облегчить разработку и исследование слабых моделей памяти в системе Coq.

Алгебра Клини - это алгебраическое обобщение бинарных отношений, регулярных выражений и некоторых других конструкций со схожей сигнатурой. Эта теория хорошо изучена и имеет ряд интересных расширений.

Цель данной работы заключается в исследовании перспективности использования теории алгебр Клини и ее расширений для упрощения формальных доказательств в системе Coq, связанных с моделями памяти.

В первой главе рассматривается алгебра Клини и ее расширения, анализируется их применимость к доказательствам в моделях памяти. Также рассматриваются существующие реализации этой теории в Coq. Формулируются цели и задачи работы.

Во второй главе описывается предлагаемый способ применения одной из реализаций теории алгебр Клини для упрощения связанных с моделями памяти доказательства. Проводится сравнение с другими способами автоматизации.

В третьей главе анализируется эффективность использования нового метода на практике, рассматриваются возникшие проблемы.

ГЛАВА 1. ОБЗОР ТЕОРИИ АЛГЕБР КЛИНИ И ЕЕ РЕАЛИЗАЦИЙ В COQ

Как было сказано выше, модели памяти могут строиться с помощью бинарных отношений над множеством элементарных событий. Такие модели называются аксиоматические [2].

В качестве примера рассмотрим Рисунок 1 с параллельной программой, которая выполняется в двух потоках. Код этой программы содержит гонку по данным: переменные *x* и *y* одновременно читаются и пишутся из разных потоков, поэтому могут иметь место несколько вариантов исполнения.

$$\begin{array}{l} a := [x] \\ [y] := 1 \end{array} \quad \Big\| \quad \begin{array}{l} b := [y] \\ [x] := 1 \end{array} \tag{1}$$

Рисунок 1 – Пример параллельной программы, содержащей гонку

Для каждого возможного исполнения строятся аксиоматически заданные базовые отношения, как например на Рисунке 2, это отношение порядка команд в коде программы (*po*) и отношения «прочитано из» (*rf*), которое показывает, где было прочитано записанное значение.

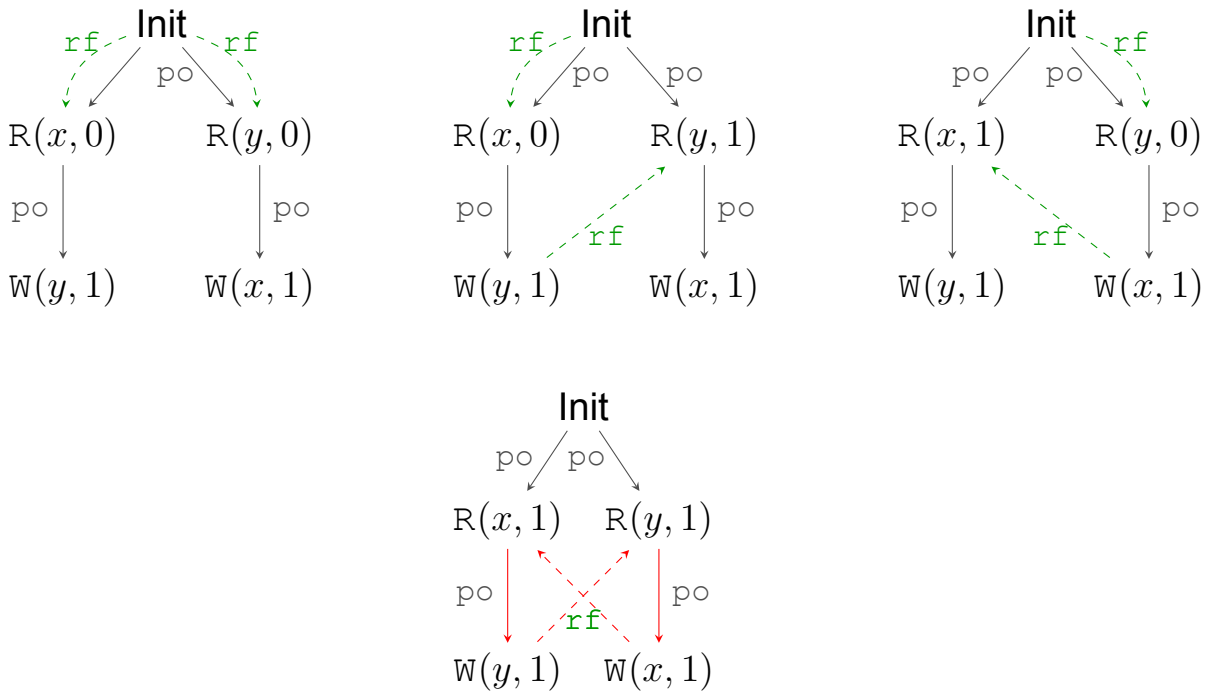

Рисунок 2 – Различные исполнения кода на Рисунке 1

Теперь, абстрагируясь от кода программы, мы можем анализировать полученные из отношений графы. Например, так как последнее поведение нельзя воспроизвести поочередным исполнением команд из разных потоков, то мы можем захотеть запретить его. Это легко сделать, потребовав иррефлексивность транзитивного замыкания объединения отношений: $(po \cup rf)^+$. Хотя в более сложных случаях этого ограничения не достаточно [24].

Такой анализ обычно преследует следующие цели:

— сформулировать ограничения на исполнения программы: например, модель памяти RC11 запрещает поведения, в которых отношение «произошло до» содержит циклы, которое в свою очередь состоит из комбинации более примитивных отношений [24];

— доказать вложенность одной модели в другую: например, компиляция нашего языка под архитектуру POWER корректна, если после нее в модели архитектуры [2] не появилось новых поведений относительно модели памяти языка;

— оценить свойства исполнения или модели в целом: например, монотонность модели, которая говорит о том, что если части исполнения программы допустимы, то и все исполнение допустимо.

Анализу также сопутствуют формальные доказательства всех утверждений и свойств, например, в системе Coq. И так как базовый формализм аксиоматических моделей памяти основан на бинарных отношениях, нам часто приходится доказывать свойства бинарных отношений, их равенства или неравенства, которые не специфичны для моделей памяти (универсальны) и могут быть автоматизированы.

Поэтому цель данной работы состоит в том, чтобы упростить доказательства в моделях памяти путем автоматизации доказательств свойств бинарных отношений.

1.1. Анализ Алгебры Клини и ее расширения

Сначала рассмотрим, как можно представить бинарные отношения в виде алгебры с предикатами равенства ($=$) и неравенства ($\leqslant$) и определим свойство универсальности (общезначимости) этих соотношений.

Далее определим формальную алгебру Клини, относительно которой бинарные отношения полны и корректны как модель. Это позволит с помощью

проверки выводимости в алгебре Клини проверять равенства и неравенства бинарных отношений.

Затем рассмотрим различные расширения алгебры Клини и проанализируем их полезность в контексте моделей памяти. В конце раздела можно найти краткие итоги этого анализа в виде Таблицы 1.

1.1.1. Алгебра отношений

Для некоторого множества O, зададим *бинарные отношения* $a, b \subseteq \mathcal{P}(O \times O)$ и операции над ними:

а) композиция:

$$a \cdot b \triangleq \{(x, y) : \exists z : (x, z) \in a, (z, y) \in b\}$$

с нейтральным элементом (множеством всех петель) $1 \triangleq \{(x, x) : x \in O\}$ и нулем (пустым отношением) $0 \triangleq \emptyset$;

б) рефлексивное транзитивное замыкание:

$$a^* \triangleq \bigcup_{n \in \overline{0..n}} a^n, \text{ где } a^n \triangleq \begin{cases} a^{n-1} \cdot a & n \geqslant 1 \\ 1 & n = 0 \end{cases}$$

в) объединение:

$$a \cup b = \{(x, y) : (x, y) \in a \text{ или } (x, y) \in b\}$$

Также будем пользоваться нотацией для транзитивного замыкания: $x^+ \triangleq x \cdot x^*$.

Алгеброй отношений Rel$\langle O \rangle$ назовем кортеж $\langle O, \cup, \cdot, \_^*, 0, 1 \rangle$.

Для любых двух выражений $f, e \in Rel\langle O \rangle$ определим выражение *равенства* и *неравенства*, как теоретико-множественные операции равенства и включения множеств соответственно:

$$f = e \triangleq f \subseteq e \wedge e \subseteq f \qquad f \leqslant e \triangleq f \subseteq e$$

Обозначим за *Rel* - класс алгебр *Rel*$\langle O \rangle$ для всех O.

Для определения общезначимости оценки выражения введем еще несколько обозначений:

— конечный алфавит Σ;

— выражения f, e состоящие из конечного количества связок $(\cup, \cdot, \_^*, 0, 1)$ и атомарных переменных из алфавита Σ;

— элементарная функция оценки S, которая сопоставляет атомарным переменным произвольные бинарные отношения из алгебры $Rel\langle O\rangle$.

Тогда (не)равенство выражений f и g является *общезначимым в классе алгебр отношений* ($Rel \models f \leqslant g, Rel \models f = g$), если после оценки переменных любой функцией S для любой алгебры $Rel\langle O\rangle$ получается верное (не)равенство бинарных отношений.

Неформально говоря, общезначимое соотношение является универсальным свойством бинарных отношений, не зависящим от значений переменных. Например: $r \cup (r \cdot r) \leqslant r^*$

1.1.2. Алгебра Клини

Назовем *Алгеброй Клини* кортеж $\langle A, \cup, \cdot, \_^*, 0, 1\rangle$ такой, что $\langle A, \cup, \cdot, 0, 1\rangle$ - идемпотентное полукольцо над множеством A, а операция $\_^*$ удовлетворяет ряду аксиом [15]:

а) $1 \cup a \cdot a^* \leqslant a^*$;

б) $1 \cup a^* \cdot a \leqslant a^*$;

в) $b \cup a \cdot x \leqslant x \Rightarrow a^* \cdot b \leqslant x$;

г) $b \cup x \cdot a \leqslant x \Rightarrow b \cdot a^* \leqslant x$;

где $a \leqslant b \triangleq a \cup b = b$.

Тогда за *KA* обозначим аксиоматизированную теорию над алгеброй Клини, состоящую из аксиом идемпотентного полу-кольца и аксиом (а-г).

Записью $KA \vdash f = g$ будем обозначать факт, что равенство $f = g$, является логическим следствием аксиом *KA*. Аналогично определим $KA \vdash f \leqslant g$.

Известно, что теория алгебры Клини разрешима и является PSpace-полной [25]. Также *KA* полна и корректна относительно алгебры отношений [5]: $Rel \models f = g \Leftrightarrow KA \vdash f = g$.

Поэтому если мы формализуем в Coq процедуру разрешения *KA*-(не)равенств, мы сможем автоматически доказывать общезначимые факты для бинарных отношений, которые удовлетворяют сигнатуре *Rel*. Существующие реализации будут рассмотрены ниже.

Уже такой небольшой набор операций позволит нам сформулировать с помощью (не)равенств *KA*-выражений, например, условие рефлексивности от-

ношения r:

$$r \cup 1 \leqslant r$$

Пояснение: рефлексивное отношение содержит в себе все петли, то есть 1, поэтому при объединении не изменится.

Также это позволит автоматически доказывать сложные неравенства, необходимость в которых иногда возникает на практике:

$$(r_1 \cup r_2)^+ \leqslant r_1^+ \cup (r_2^+ \cdot r_1^*) \cup (r_2^* \cdot (r_1^+ \cdot r_2^+)^+ \cdot r_1^*)$$

1.1.3. Алгебра Клини с тестами

Теперь рассмотрим очень полезное расширение KA: алгебра Клини с тестами.

алгебра Клини с тестами, это кортеж $\langle K, \mathbb{B}, [\cdot] \rangle$ такой, что:

— K - алгебра Клини $\langle A, \cup, \cdot, \_^*, 0, 1 \rangle$;

— $\mathbb{B}$ - булева алгебра $\langle B, \wedge, \vee, \neg, \top, \bot \rangle$ (тесты);

— $[\,\_\,]$ - гомоморфизм из $\langle B, \vee, \wedge, \top, \bot \rangle$ в $\langle A, \cup, \cdot, 1, 0 \rangle$.

Аналогично за *KAT* обозначим аксиоматическую теорию, состоящую из аксиом *KA*, аксиом булевой алгебры $\mathbb{B}$ и свойств гомоморфизма. И аналогично обозначим за $KAT \vdash f = g, f \leqslant g$ выводимость в этой теории.

Также расширим определение бинарных отношений $Rel\langle O \rangle$, добавив булеву алгебру $\langle \mathcal{P}\langle O \rangle, \cap, \cup, O, \emptyset \rangle$ и гомоморфизм $[a] \triangleq \{(x, x) : x \in a\}$. То есть тесты в модели бинарных отношений - это множества элементов из O, которые можно воспринимать как логические элементы. А гомоморфизм $[\,\_\,]$ сопоставляет им множество петель. Последовательная композиция отношений с тестами позволяет фильтровать их по истинностным значениям этих тестов.

Такая расширенная алгебра отношений полна и корректна относительно алгебры Клини с тестами [16]: $KAT \vdash f = g \Leftrightarrow Rel \models f = g$.

Задача проверки выводимости (не)равенств в *KAT* разрешима и является PSpace-полной [10]. Также существует формализация алгоритма разрешения в Coq [23], которая будет рассмотрена в следующих разделах.

В моделях памяти доменами отношений являются элементарные события, потому данная теория позволяет нам накладывать любые вычислимые предикаты на события в программе. Например, мы можем определить множество операций чтения/записи как W/R соответственно. Это позволяет нам,

сказать, что отношение «прочитано из» (rf) задано только между операциями чтения и записи [24]:

$$\mathrm{rf} \leqslant [W] \cdot \mathrm{rf} \cdot [R]$$

Пояснение: $[W]$ - это множество петель над событиями записи, а $[R]$ - над операциями чтения. Если ребра отношения *rf* существуют только между ними, то после фильтрации отношение не уменьшится.

1.1.4. Алгебры Клини с инверсией

Теперь добавим к алгебре Клини операцию инверсии $\_^{\smile}$, удовлетворяющую следующим аксиомам:

а) $(a \cup b)^{\smile} = a^{\smile} \cup b^{\smile}$;

б) $(a \cdot b)^{\smile} = b^{\smile} \cdot a^{\smile}$;

в) $(a^*)^{\smile} = (a^{\smile})^*$;

г) $a^{\smile\smile} = a$;

д) $a \leqslant a \cdot a^{\smile} \cdot a$.

Тогда мы получим теорию *алгебр Клини с инверсией* (*KAC*). В ней аналогично с *KA* определяются (не)равенства, оценки и выводимость.

В алгебре отношений операции инверсии соответствует операция переворачивания всех ребер:

$$a^{\smile} \triangleq \{(y, x) : (x, y) \in a\}$$

KAC корректна и полна относительно бинарных отношений с операцией инверсии, но не корректна относительно регулярных языков, в частности, аксиома (д) в них не выполнена. Теорию без этой аксиомы обычно обозначают как KAC^- [5].

Для *KAC* задача разрешимости остается PSpace-полной, но, в сравнении с KAC^-, алгоритмы разрешения пока что сложны для формализации в Coq [5], поэтому полных реализаций пока нет.

В моделях памяти иногда возникают функциональные отношения, для которых верно, что каждому элементу слева отношение сопоставляет не более одного справа. Свойство, что отношение r является функциональным, можно выразить в *KAC* с помощью операции инверсии ($\_^{\smile}$):

$$r^{\smile} \cdot r \leqslant 1$$

Пояснение: если отношение функциональное, то любой переход по ребру от «значения» к «аргументу» и по любому другому ребру опять к «значению» приведет нас в тоже значение. То есть композиция таких переходов - это всегда петля.

1.1.5. Решетки Клини

Теперь рассмотрим одно из самых значительных расширений алгебр Клини. Если мы добавим к *KA* операцию пересечения($\cap$), то получим *решетку Клини* (*KL*).

Алгебра отношение *Rel* в этом случае дополняется следующей операцией:

$$f \cap e \triangleq \{(x, y) \colon (x, y) \in f \wedge (x, y) \in e\}$$

Данная аксиоматическая теория является разрешимой [8], но алгоритм разрешения чрезвычайно сложен и требует экспоненциального количества памяти, поэтому автоматизация (не)равенств из этой теории в Coq пока не представляется возможной. Также существует доказательство того, что эта задача является EXPSpace-полной задач [13].

Тем не менее операция пересечения была бы очень полезна в моделях памяти. Часто ограничение на исполнение программы формулируется как ацикличность отношения «произошло до», которое в свою очередь является комбинацией других базовых отношений [24]. С помощью новой операции ацикличность легко выразить неравенством:

$$r^{+} \cap 1 \leqslant 0$$

Пояснение: любой цикл при транзитивном замыкании приводит к появлению петель и наоборот.

1.1.6. Аллегории Клини

Если же мы одновременно добавим в алгебру Клини операции пересечения $\cap$ и инверсии ($\_^{\smile}$), то получим богатую теорию *аллегорий Клини (KAl)*.

Недавние результаты показывают, что для такой теории существует система аксиом, которая полна и корректна, но пока только относительно регулярных языков [7]. Она также разрешима и EXPSpace-полная [6].

Кроме создания системы аксиом для бинарных отношений, остается также открытым вопрос о добавлении в теорию универсального отношения ($\top \triangleq O \times O$) [7]. Это позволило бы, например, выразить тотальность отношения r, то есть факт того, что между любой парой событий есть ребро хотя бы в одну сторону: $\top \leqslant r \cup r^{\smile}$.

Таблица 1 – Сравнение расширений алгебр Клини

Название	Пример доказываемого свойства бинарных отношений	Сложность	Реализация в Coq
KA	Рефлексивность, Транзитивность	PSpace	relation-algebra, atbr
KAT	Предикаты на доменах	PSpace	relation-algebra
KAC	Функциональные отношения	PSpace	—
KL	Ацикличность, Разность	EXPSpace	—
KAl_{Lang}	—	EXPSpace	—

1.2. Анализ существующих реализаций в Coq

Теперь рассмотрим существующие реализации алгебры Клини и ее расширений в Coq: библиотеки atbr [4] и relation-algebra [23]. Остановимся на каждой подробно и опишем их минусы и плюсы.

1.2.1. atbr

Эта библиотека предоставляет инструменты для автоматического доказательства общезначимых (не)равенств в *KA* и KAC^{-}.

То есть в случае, когда выражение содержит операцию инверсии $\_^{\smile}$, у нас нет гарантии, что общезначимое (не)равенство будет доказано автоматически. Алгоритм разрешения для KAC^{-} значительно проще, чем для *KAC* и сводится лишь к проталкиванию операции инверсии до атомарных переменных, что вряд ли может быть полезно для бинарных отношений.

KA реализована в полной мере с доказательством теоремы о полноте, что позволяет быть уверенным в корректности кода библиотеки.

1.2.2. relation-algebra

Эта библиотека является развитием atbr, в которой допущена ошибка при дизайне, и поэтому авторам перед расширением функционала пришлось переписать ее с нуля. В ней реализована полная поддержка *KAT* и *KA*.

Также важно отметить, что relation-algebra реализует алгоритм исключения гипотез [9, 14], что позволяет автоматически доказывать не только общезначимые, но и те (не)равенства которые следуют из гипотез определенного вида, а конкретнее, сводящиеся к равенствам Хоара ($r = 0$).

Это чрезвычайно полезно на практике. Например, мы можем доказывать монотонность некоторых свойств отношений, то есть факт того, что если для нескольких отношений свойство выполняется, то и для их комбинации оно тоже будет выполнено. К примеру монотонность пустоты отношений:

$$r_1 \leqslant 0, r_2 \leqslant 0 \vdash r_1 \cup r_2 \leqslant 0 \qquad\qquad r \leqslant 0 \vdash r^+ \leqslant 0$$

В своем интерфейсе библиотека использует канонические структуры и классы типов [18] в качестве средств полиморфизма. Это позволяет ей обобщить алгоритм разрешения (не)равенств для любой структуры, которая отвечает сигнатуре и аксиомам *KAT*.

Чтобы генерация доказательств работала для конкретно заданной структуры необходимо предъявить операции, соответствующие сигнатуре *KAT* и доказать, что для них выполнены все аксиомы.

1.3. Постановка задачи

Для того, чтобы исследовать применение теории алгебры Клини на практике, была взята библиотека hahn [1]. Она содержит базовую формализацию модели исполнения и доказательства огромного числа лемм, которые непосредственно используются для построения различных моделей памяти [22].

Так как цель этой работы заключается в исследовании применимости теории Клини к моделям памяти, то задачи этой работы следующие:

— применить библиотеку relation-algebra, которая реализует автоматическое разрешение *KAT*-(не)равенств, к доказательствам в hahn с целью упростить их;

— оценить объем доказательств, которые получилось упростить; проанализировать перспективность нового метода.

Выводы по главе 1

Расширения теории алгебр Клини могут сильно облегчить разработку и исследование аксиоматических моделей памяти, так как через них можно выразить многие важные в этой области свойства. К сожалению, немногие расширения возможно реализовать в Coq эффективно и даже те, что реализованы, потребовали огромного труда разработчиков.

Но возможно, что даже существующие разработки в этой области помогут существенно упростить работу с моделями памяти.

ГЛАВА 2. ПРИМЕНЕНИЕ *KAT* К МОДЕЛЯМ ПАМЯТИ

Теперь опишем предлагаемый способ применения библиотеки для автоматического разрешения (не)равенств бинарных отношений с тестами relation-algebra для упрощения доказательств в библиотеке hahn.

Далее сравним новый подход к автоматизации доказательств с теми, которые уже применяются в hahn для автоматизации доказательств.

2.1. Описание решения проблемы

В Coq доказательства состоят из последовательности команд (*тактик*), каждая из которых упрощает или полностью доказывает текущее утверждение.

Библиотека relation-algebra предоставляет тактики `kat` и `hkat`, которые позволяют автоматически доказать общезначимые или следующие из специальных гипотез (не)равенства соответственно.

Как упоминалось в первой главе, для их использования необходимо определить сигнатуру *KAT* для используемой в hahn реализации бинарных отношений и доказать выполнимость аксиом.

Чтобы автоматически доказать с помощью новых тактик какое-то утверждение необходимо, чтобы оно было сформулировано как проверка общезначимости *KAT*-(не)равенств (или как проверка следствия из гипотез, с которыми способна работать `hkat`). Поэтому также необходимо переформулировать некоторые утверждения.

2.1.1. Определение экземпляра *KAT*

В библиотеке hahn уже определены многие базовые операции с бинарными отношениями, такие как объединение ($r_1 \cup r_2$) или композиция ($r_1 \cdot r_2$). Поэтому предлагается лишь объявить экземпляр канонической структуры `kat.ops`, которую предоставляет relation-algebra. Он сопоставит сигнатуру *KAT* с уже определенными в hahn операциями. Также этот экземпляр будет содержать булеву алгебру доменов отношений, которую для этого требуется определить и доказать необходимые аксиомы.

Далее интерфейс библиотеки relation-algebra требует определить экземпляр класса типов `kat.laws`, содержащий доказательства выполнения всех аксиом *KAT* для операций из hahn. Все они были доказаны с помощью встроенных в Coq тактик, а также частично были переиспользованы леммы из relation-algebra и тактики из hahn.

После этого механизмы вывода типов в Coq могут восстановить информацию о том, что выражение принадлежит *KAT*-сигнатуре, если оно состоит из соответствующих операций. Поэтому тактики `kat(hkat)` смогут автоматически сгенерировать доказательства для утверждений, если те являются общезначимыми (или следуют из специальных гипотез).

2.1.2. Переформулирование определений

Некоторые леммы в hahn сформулированы в терминах не принадлежащих сигнатуре *KAT*, но тем не менее могут быть в ней переформулированы.

Для примера рассмотрим утверждение `max_elt a r`. Оно означает, что `a` является максимальным событием в отношении `r`, то есть из `a` нет исходящих ребер отношения. В hahn оно определяется так:

```
1 Definition max_elt {A: Type} (r: relation A) (a: A) :=
2   forall (b: A), ¬ (r a b).
```

Оно не отвечает сигнатуре алгебры Клини с тестами, и его нельзя доказать автоматически. Но его можно переформулировать как $[eq\ a] \cdot r \leqslant \emptyset$, где $eq\ a$ - предикат проверяющих событие на равенство с a. То есть a - максимальный элемент тогда и только тогда, когда множество путей, которые сначала проходят по петле в элементах равным a, а потом по отношению r, пусто (иначе из a есть исходящие ребро).

Чтобы формально обосновать предположение о том, что оба этих определения эквиваленты, докажем лемму:

```
1 Lemma max_elt_iff_kat:
2   forall (a: A) (r: relation A), max_elt r a ↔ [eq a] ⋅ r ⩽ ∅ .
```

Также, используя эту лемму, с помощью последовательного применения встроенной тактики `rewrite` мы можем в процессе доказательства заменять старые определения на новые:

```
1 Lemma max_elt_iter: max_elt r a → max_elt r⁺ a.
2 Proof.
3   repeat rewrite → max_elt_iff_kat.
4   (* current goal: [eq b] ⋅ r ⩽ ∅ → [eq b] ⋅ r⁺ ⩽ ∅ *)
5   hkat.
6 Qed.
```

Если утверждение примет необходимый вид, то `(h)kat` автоматически докажет его.

Благодаря такому подходу, мы не будем переписывать определения всех теорем и лемм, а только изменим их доказательства. Это сохранит их интерфейс и не сломает нетронутых доказательств, которые используют переформулируемые утверждения.

В этом примере у нас получилось привести исходное определение `max_elt` к виду $r \leqslant \emptyset$, что дает возможность не только автоматически доказывать такие утверждения, но и использовать их в качестве гипотез. Благодаря этому, после переписывания всех вхождений `max_elt` в утверждении леммы `max_elt_iter`, тактика `hkat` автоматически сгенерировала остальное доказательство.

Кроме гипотез вида $r \leqslant \emptyset$ библиотека relation-algebra поддерживает гипотезы нескольких других видов [23], которые к нему сводятся или могут быть исключены [9, 14]:

— $r = 0$;
— $[a] \cdot x = x \cdot [b]$, $[a] \cdot x \leqslant x \cdot [b]$;
— $x \cdot [a] = [b] \cdot x$, $x \cdot [a] \leqslant [b] \cdot x$;
— $r \leqslant [a] \cdot r$, $r \leqslant r \cdot [a]$;
— $a = b$, $a \leqslant b$;
— $[a] \cdot r = [a]$, $r \cdot [a] = [a]$, где r - атомарная переменная.

Исходя из выше описанного, все определения в hahn, которые получилось выразить в сигнатуре *KAT* можно разделить на 3 группы:

а) Утверждения, которые можно использовать как гипотезы и доказывать их автоматически.
б) Утверждения, которые нельзя использовать как гипотезы, но можно их доказывать.
в) Определения отношений, которые могут входить в совершенно любые утверждения.

Те утверждения, которые не получилось переформулировать или которые относятся ко второй группе будут просто игнорироваться тактикой `hkat`. А отношения, определение которых не выражается в *KAT* будут восприниматься как атомарные.

В Таблице 2 указаны все переопределения, которые удалось выполнить в hahn, разбитые по этим группам. Также в Таблице 3 можно найти несколь-

ко важных определений, которые не удалось выразить через *KAT*, но удалось через другие расширения алгебры Клини.

Таблица 2 – Переформулирование определений в сигнатуру *KAT*

Название	Оригинальное определение (в логике первого порядка)	*KAT*
Переформулирование утверждений, которые можно использовать в качестве гипотез		
`upward_closed r p`	$\forall xy.r\ x\ y \Rightarrow p\ y \Rightarrow p\ x$	$r \cdot [p] \leqslant [p] \cdot r$
`doma r p`	$\forall xy.r\ x\ y \Rightarrow p\ x$	$r \leqslant [p] \cdot r$
`domb r p`	$\forall xy.r\ x\ y \Rightarrow p\ y$	$r \leqslant r \cdot [p]$
`max_elt a r`	$\forall y.\neg(r\ a\ y)$	$[eq\ a] \cdot r \leqslant 0$
`min_elt a r`	$\forall x.\neg(r\ x\ a)$	$r \cdot [eq\ a] \leqslant 0$
`wmax_elt a r`	$\forall y.r\ a\ y \Rightarrow a = y$	$[eq\ a] \cdot r \leqslant [eq\ a] \cdot r \cdot [eq\ a]$
`wmin_elt a r`	$\forall x.r\ x\ a \Rightarrow a = x$	$r \cdot [eq\ a] \leqslant [eq\ a] \cdot r \cdot [eq\ a]$
`DOM a r`	$\forall xy.r\ x\ y \Rightarrow x = a$	$r \leqslant [eq\ a] \cdot r$
`COD a r`	$\forall xy.r\ x\ y \Rightarrow y = a$	$r \leqslant r \cdot [eq\ a]$
Переформулированные утверждений, которые нельзя использовать в качестве гипотез		
`transitive r`	$\forall xyz.r\ x\ z \Rightarrow r\ z\ y \Rightarrow r\ x\ y$	$r \cdot r \leqslant r$
`reflexive r`	$\forall x.r\ x\ x$	$1 \leqslant r$
Переформулирование определений отношений		
`restr_rel p r`	$\forall xy.r\ x\ y \wedge p\ x \wedge p\ y$	$[p] \cdot r \cdot [p]$
`clos_refl r`	$\forall xy.x = y \vee r\ x\ y$	$1 \cup r$

где (`eq a : A →` **`Prop`**) - предикат равенства с `a`, (`r : relation A`) - отношение, (`a, x, y : A`) - события, (`p: A →` **`Prop`**) - предикаты на событиях, `A` - тип доменов отношений, событий

Написав предложенные переопределения и доказав соответствующие леммы об их корректности, мы сможем автоматически доказывать утверждения, которые удалось выразить с помощью алгебры Клини с тестами, если они общезначимы или следуют из утверждений, которые удалось переформулировать в специальной форме.

Важно отметить, что общезначимость означает, что для алгоритма разрешения (не)равенств предикаты и атомарные отношения непрозрачны. То есть он не может использовать те свойства, которые следуют из их определения.

Например, лемму `seq_eq_wmax` на Листинге 1, `hkat` доказать не сможет, несмотря на то, что она принадлежит сигнатуре *KAT*, так как рассматривает `eq b` как атомарный символ.

Таблица 3 – Переформулирование определений, которые не удалось выразить в KAT, но можно выразить в других расширениях алгебры Клини

Название	Оригинальное определение (в логике первого порядка)	Новое определение
`irreflexive r`	$\forall x.\neg(r\ x\ x)$	$1 \cap r \leqslant 0$
`acyclic r`	$irreflexive\ r^+$	$1 \cap r^+ \leqslant 0$
`is_total r`	$\forall xy.r\ x\ y \vee r\ y\ x$	$\top \leqslant r \cup r^\smile$
`cross_rel p1 p2`	$\forall xy.p_1\ x \wedge p_2\ y$	$[p_1] \cdot \top \cdot [p_2]$
`singl_rel a b`	$\forall xy.x = a \wedge y = b$	$[eq\ a] \cdot \top \cdot [eq\ b]$

Пояснение: красным выделены не входящие в сигнатуру KAT связки, где $\top$ - универсальное отношение, ($p_1, p_2 : A \to$ **Prop**) - предикаты на событиях

Но зная, что этот предикат `eq` выражает равенство, мы можем легко доказать недостающее свойство (`[eq b] · r · [eq b] ⩽ [eq b]`) и закончить доказательство леммы. К сожалению, это свойство не сводится к равенству Хоара, и его нельзя использовать как гипотезу.

```
1  Lemma wmax_elt_iff_kat:
2    wmax r b ↔ [eq b] ·r ⩽ [eq b] ·r ·[eq b].
3
4  Lemma seq_eq_wmax: wmax r b → [eq b] ·r ⩽ [eq b].
5  Proof.
6    rewrite → wmax_elt_iff_kat.
7    (* current goal:
8       [eq b] ·r ⩽ [eq b] ·r ·[eq b] → [eq b] ·r ⩽ [eq b] *)
9    Fail hkat. (* can't solve *)
10   assert ([eq b] ·r ·[eq b] ⩽ [eq b]) as H.
11   { unfold_all; firstorder; congruence. }
12   rewrite → H; trivial
13 Qed.
```

Листинг 1 – Пример невозможного использования hkat из-за нехватки гипотезы о внутреннем устройстве предиката

В заключении, чтобы удобно переписывать утверждения состоящие из разных определений, воспользуемся тактикой `autorewrite` [11], что позволит использовать динамическую базу утверждений для переписывания, которую можно пополнять по мере добавления новых определений.

Так же для удобства объявим тактики `hahn_(h)kat`, которые будут применять `autorewrite`, а потом вызывать `(h)kat`, что также скроет все переформулировки от пользователя тактик (см. Листинг 2)

```
1   Hint Rewrite restr_rel_iff_kat: redefDb.
2   Hint Rewrite wmax_elt_iff_kat: redefDb.
3   ...
4   Ltac lift_to_kat := repeat autorewrite with redefDb in *.
5   Ltac hahn_kat := lift_to_kat; kat.
6   Ltac hahn_hkat := lift_to_kat; hkat.
```

Листинг 2 – Переопределение тактик `h(kat)`, для автоматической замены старых определений на новые

2.2. Сравнение с существующими решениями

На данный момент в hahn большинство доказательств состоит из ручного использования других, уже доказанных утверждений, их автоматического перебора и стандартных тактик Coq.

Проблема ручного использования очевидна. И по крайней мере от части такого труда (`h`)`kat` может помочь избавится.

Автоматический же перебор тоже хорошо справляется с этой задачей, но он предназначен в первую очередь для простых утверждений. Проблема этого подхода заключается в том, что не всегда можно по утверждению понять «простое» ли оно. Иногда перебор занимает значительное время и приходится параметризовать его глубиной перебора и вручную стараться выбирать минимальное возможное ее значение. Это делает использование этого подхода не удобным.

Главным преимуществом использования автоматических средств доказательств утверждений из некоторой разрешимой теории заключается в том, что существует четкий критерий, когда метод работает, а когда нет: если утверждение имеет сигнатуру этой теории и оно общезначимо в ней. Это не только делает его более удобным, но дает возможность формально доказывать полноту алгоритма генерации доказательств.

Кроме предложенного метода к таким средствам относятся, например, встроенная тактика `tauto` [11] для интуиционистского исчисления высказываний.

Это помогает не только при выборе того или иного метода доказательства, но также дает целую область знаний о базовом формализме, на которую мы можем опереться в своих рассуждениях. Например, зная, что легко можно доказать любое свойство бинарных отношении с тестами, разработчик может

выписать через (не)равенство необходимое преобразование текущего утверждения, доказать его одной тактикой (h)kat и выполнить преобразование:

```
1 Proof.
2 ...
3 (* current goal: irreflexive (dd* · (dd* · de · ee* · ed)+) *)
4 assert (dd* · (dd* · de · ee* · ed)+ ⩽ (dd* · de · ee* · ed)+) as H.
5 { kat. }
6 rewrite → H.
7 (* current goal: irreflexive ((dd* · de · ee* · ed) +) *)
8 ...
9 Qed.
```

Листинг 3 – Пример доказательства вспомогательного утверждения, необходимого для преобразования, с помощью *KAT*

Все же автоматический перебор покрывает большее число простых случаев, потому что слабо ограничен сигнатурой. А например, даже самое простое утверждение, которое содержит пересечение отношений, (h)kat решить не в силах. Но по мере усложнения утверждений количество бесполезных действий, совершаемых перебором, резко увеличивается, чего не происходит с более специализированными методами.

Поэтому эти два подхода не исключают друг друга, а дополняют.

Выводы по главе 2

Применение *KAT* для автоматизации доказательств - это не универсальный, но удобный инструмент, который служит дополнением к уже имеющимся, что может сделать разработку моделей памяти проще и быстрее.

ГЛАВА 3. АНАЛИЗ РЕЗУЛЬТАТОВ ПРИМЕНЕНИЯ *KAT* К МОДЕЛЯМ ПАМЯТИ

Теперь, когда мы получили возможность использовать новый метод автогенерации доказательств, проанализируем его применение к леммам в библиотеке hahn.

Сначала в этой главе будут рассмотрены проблемы, с которыми пришлось столкнуться при этом, а также будет описано как удалось полностью или частично их решить.

После этого подробно будет описано, как с помощью нового метода изменились доказательства в hahn.

3.1. Проблемы в использовании `(h)kat`

После реализации всех экземпляров классов типов и канонических структур оказалось, что Coq все еще не может восстановить информацию о том, что некоторые операции принадлежат сигнатуре *KAT*.

Так же оказалось, что тактика `hkat` тратит очень много времени на анализ гипотез, которые она может использовать для генерации доказательства.

3.1.1. Проблемы с выводом типов

Изначально библиотека была спроектирована с использованием наиболее общих интерфейсов. Поэтому сигнатура моноида, которую включает в себя *KAT*, сформулирована с учетом поддержки гетерогенных структур. Применительно к бинарным отношениями это значит, что типы левых и правых концов могут отличаться.

В hahn бинарные отношения гомогенные, то есть соединяются элементы одного и того же типа: типа событий в программе. Поэтому типы левых и правых концов, которыми параметризован моноид, не используются при определении экземпляра сигнатуры. В результате чего, механизм унификации Coq не может вывести эти типы, и тактики `kat` и `hkat` не работают.

Решением этой проблемы стало явное указание типов-параметров моноида. Для этого определение оргинальных тактик `hkat/kat` было скопированно в `hahn_kat` и `hahn_hkat` было скопировано и явно прописаны типы параметров. Заметим, что скопирована была не вся реализация этих тактик, а лишь верхний уровень, который вызывал алгоритм непосредственно разрешения (не)равенств. Этого оказалось достаточно.

3.1.2. Скорость агрегации гипотез

Тактика `hkat` включает в себя алгоритм исключения гипотез [9, 14], который сводит задачу доказательства из гипотез к доказательству общезначимого утверждения, и потом фактически применяет тактику `kat`. Этот алгоритм должен сначала агрегировать гипотезы, то есть привести их к одной гипотезе вида $r \leqslant 0$ и именно этот процесс занимает наибольшее время работы `hkat`, иногда больше минуты.

Дело в том, что агрегация работает путем перебора всех гипотез текущего доказательства и последовательных попыток применения всех лемм агрегации, которые разбивают равенства на два неравенства, объединяют неравенства вида $r \leqslant 0$ и приводят гипотезы из специального вида к $r \leqslant 0$.

Главным образом проблема производительности была решена тем, что все переформулирования были изменены так, чтобы новые определения сразу имели предагрегированный вид ($r \leqslant 0$).

Также было обнаружено, что перебор в агрегации был не эффективным, так как часто пытался трансформировать гипотезы, которые уже были приведены к виду $r \leqslant 0$, что всегда оканчивалось неудачей. Поэтому разделением его на фазы удалось дополнительно ускорить `hahn_hkat` в среднем на 100 миллисекунд на леммах из hahn при среднем времени выполнения тактики равным одной секунде. В некоторых случаях ускорение составило до 20 процентов. Оценка проводилась с помощью встроенной команды `Time` и замеров времени комплияции всей библиотеки.

3.2. Описание результатов

Некоторые доказательства в hahn получилось полностью автоматизировать, а некоторые значительно упростить. Рассмотрим отдельно каждую из этих групп доказательств, и в конце приведем полную статистику.

3.2.1. Примеры полностью доказанных утверждений

Леммы, которые получилось полностью доказать с помощью новых тактик особенно интересны потому, что в них может пропасть необходимость. Вспомогательные утверждения выносятся как отдельные леммы, чтобы ускорить проверку доказательств или для того чтобы не перегружать текущее доказательство. Второй случай перестает быть актуальным, если утверждение полностью доказывается одной тактикой.

Для примера рассмотрим фрагмент доказательства одного из свойств ацикличности, фундаментального понятия в контексте моделей памяти:

```
1  Lemma acyclic_union: acyclic r → acyclic (r' ⋅ r^*) →
2    acyclic (r ∪ r').
3  Proof.
4    unfold acyclic; ins.
5    (* current goal: irreflexive (r ∪ r')^+ *)
6    rewrite path_union.
7    (* current goal: irreflexive (r^+ ∪ (r^* ⋅ r')^+ ⋅ r^+) *)
8  ...
9  Qed.
10
11 Lemma path_union (r r': relation A) :
12   (r ∪ r')^+ ⩽ r^+ ∪ (r^* ⋅ r')^+ ⋅ r^+.
13 Proof.
14   apply inclusion_t_ind_right.
15   unionL; [vauto|].
16   by rewrite rtE; rewrite ← !ct_step; basic_solver 12.
17   relsf; unionL.
18   - by unionR left; rewrite ct_unit.
19   - by rewrite !seqA; rewrite ← ct_end; basic_solver 12.
20   - rewrite (ct_step (r^+ ⋅ r')).
21   rewrite ← inclusion_t_rt at 1; basic_solver 22.
22   - rewrite !seqA, inclusion_t_rt at 1.
23   rewrite ← (ct_end (r^* ⋅ r')); basic_solver 12.
24 Qed.
```

Лемму `acyclic_union` мы не можем полностью автоматизировать, так как ацикличность требует операции пересечения отношений ($\cap$), но ее доказательство использует в качестве преобразования неравенство `path_union`, которое новый метод может доказать автоматически.

Раньше лемма `path_union` требовала десяток нетривиалных строчек доказательства, где в нескольких местах использовались тактики с разной глубиной перебора `basic_solver N`.

Теперь мы можем доказать эту вспомогательную лемму одной тактикой или вообще не выносить ее и сгенерировать доказательство «на месте»:

В некоторых подобных случаях тактики `hahn_(h)kat` заменили собой доказательства, занимающие до 70 строк.

Кроме преобразований бинарных отношений с тестами полностью автоматизировались монотонность свойств, которые тактика `hkat` может использовать как гипотезы (см. Таблицу 2).

```
Lemma acyclic_union: acyclic r → acyclic (r' ⋅ r^*) →
  acyclic (r ∪ r').
Proof.
  unfold acyclic; ins.
  assert(path_union:(r ∪ r')^+ ⩽ r^+ ∪ (r^* ⋅ r')^+ ⋅ r^+) by hahn_kat.
  rewrite path_union.
...
Qed.
```

Например монотонность свойства элемента быть максимальным в отношении (в комментариях приведено старое доказательство):

```
Lemma max_elt_t : max_elt r a → max_elt (r^+) a.
Proof. hahn_hkat. Qed.
(* Proof.
    red; ins; apply clos_trans_t1n in REL; induction REL; eauto.
  Qed. *)
```

3.2.2. Примеры частичного применения `(h)kat`

Некоторые из тех доказательств, которые не удалось автоматизировать полностью, получилось значительно упростить.

Во многих случаях когда полной автоматизации мешает всего одна «сложная» гипотеза, после ее использования оставшаяся часть доказательства решается автоматически.

Также тактиками `hahn_(h)kat` получилось заменить в некоторых местах тактики основанные переборе. Это имеет смысл, потому что при написании новых доказательств разработчик, смотря на текущее утверждение, скорее всего будет использовать инструменты, которые гарантированно его доказывают.

```
Lemma seq_ct_absorb_r: r^+ ⋅ r' ⊆ r' ∪ r^* ⋅ ((r ⋅ r') \ r').
Proof.
  rewrite ← seq_rt_absorb_r; hahn_kat.
  (* rewrite ← seq_rt_absorb_r; eauto with hahn. *)
Qed.
```

Листинг 4 – Пример замены тактики основанной на переборе на `kat` (в комментариях указано старое доказательство)

В предыдущем разделе уже было сказано, что упростилась работа с общезначимыми преобразованиями отношений. Благодаря этому удалось частично упростить леммы, в которых доказательство этих преобразований не было вынесено.

Также в нескольких леммах было достаточно добавить в контекст уже доказанное утверждение, чтобы `hkat` смогла использовать его как гипотезу и автоматически сгенерировать доказательство:

```
1 Lemma seq_eqv_max: [max_elt r] ⋅r ⩽ 0.
2 Proof. basic_solver. Qed.
3
4 Lemma seq_eqv_max_t: [max_elt r] ⋅r⁺ = 0.
5 Proof.
6   pose (@seq_eqv_max r); hahn_hkat.
7   (* rewrite ct_begin; seq_rewrite seq_eqv_max; basic_solver. *)
8 Qed.
```

Листинг 5 – Пример автоматизации с использованием уже доказанной гипотезы (в комментариях указано старое доказательство)

3.2.3. Общая статистика

Всего hahn содержит 973 леммы. И часть из них связана не только с отношениями, но и, например, со списками. Поэтому при подсчете статистики учитывались только те модули, которые содержат свойства бинарных отношений или предикатов на их домены. Они суммарно содержат 671 лемму и 1489 строк доказательств.

С помощью нового метода упростились 229 лемм и удалось автоматизировать около 500 строчек доказательств. При этом время компиляции hahn не деградировало.

187 лемм не изменили свой размер. В них 3-4 тактики или одна, основанная на переборе, заменились на одну тактику `hahn_(h)kat`, что никак не повлияло на размер доказательств в строчках, но упростило их.

Чистая же разница составила 261 строку доказательств. Более подробную статистику по каждой тактике можно посмотреть в Таблице 4.

Статистику по сокращению и автоматизации доказательств в зависимости от определений, которые они содержатся в леммах, можно увидеть в Таблице 5. По ней видно, что далеко не все леммы, содержащие утверждение `wmax(min)_elt`, могут быть автоматизированы, что связано с непрозрачностью предиката `eq`. Большое количество лемм использует в качестве предпосылки транзитивность или рефлексивность отношений, которые невозможно выразить в виде равенств Хоара ($r \leqslant 0$) или исключить. Проблема остальных лемм

Таблица 4 – Статистика применения *KAT* в библиотеке hahn, разбитая по тактикам

Тактика	Полностью доказанные леммы	Частично доказанные леммы	Количество сокращенных срок доказательств
`hahn_kat`	103	18	111
`hahn_hkat`	84	22	150
Всего	187	40	261

заключается в том, что они или гипотезы, из которых они следуют, не укладываются в сигнатуру *KAT*.

Таблица 5 – Статистика применения *KAT* в библиотеке hahn, разбитая по определениям

Группа определений содержащиеся в леммах*	Полностью доказанные леммы	Частично доказанные леммы	Не упрощенные леммы	Сокращено / Автоматизировано	Количество строк до упрощения
KAT-(не)равенства	74	5**	0	211/312	339
`max(min)_elt`	36	16	19	0/54	68
`doma(b)`	36	14	23	44/96	254
`wmax(min)_elt`	14	0	36	0/14	52
`restr_rel`	12	0	21	0/12	168
`COD/DOM`	8	0	6	2/10	17
`reflexive` / `transitive`	4	10	87	4/31	453
`acyclic`	0	5	33	27/32	433
`irreflexive`	0	4	28	4/8	267
Всего	187	40	250	261/491	1489

** - группы, кроме первой, могут пересекаться

* - в этой группе также находятся утверждения со «сложными» гипотезами.

Оценим также размер добавленного служебного кода: объявление экземпляров сигнатуры *KAT*, доказательство аксиом, переопределение тактик `(h)kat`, а также леммы об эквивалентности старых и новый определений. Все это занимает около 350 строк кода, из которых половина - это код специфика-

ций: объявление тактик, объявление сигнатур утверждений, импортирование модулей и т. д. Более подробные цифры можно найти в Таблице 6.

Таблица 6 – Статистика по добавленному в hahn коду

	Количество строк спецификаций	Количество строк доказательств	Количество строк комментариев
Объявление экземпляра сигнатуры *KAT*	65	41	4
Переформулирование определений	65	93	7
Переопределение тактик `(h)kat`	50	1	8
Всего	180	135	19

Если сопоставить количество сокращенных строк и количество добавленных, то можно заметить, что кодовая база библиотеки hahn почти не изменилась и даже немного выросла. Но стоит отметить, что объем доказательств строго уменьшился, а служебный код в основном состоит из небольших лемм об эквивалентности.

Этот служебный код легко расширять. При добавлении новых лемм в hahn вообще не нужно ничего менять, а при добавлении нового определения, необходимо будет лишь добавить новую лемму об эквивалентности или сразу сформулировать его в сигнатуре *KAT*.

Выводы по главе 3

Из описанных выше результатов видно, что разрешение общезначимых *KAT*-(не)равенств способно автоматизировать треть доказательств в библиотеке hahn, связанных с бинарными отношениями. А также удалось сократить размеры этих доказательств на 17 процентов.

При этом новый подход имеет четкие границы применения и не потребует в дальнейшем больших затрат на его поддержку, что делает его практичным.

Из этого можно сделать вывод, что теория алгебр Клини действительно способна ощутимо упростить разработку и исследования моделей памяти.

ЗАКЛЮЧЕНИЕ

В данной работе были проанализированы различные расширения алгебры Клини и их реализации в Coq с целью исследования их применимости к автоматизации доказательств, связанных с моделями памяти.

Была выбрана реализация алгебры Клини с тестами (relation-algebra [23]) как самая полная из существующих сегодня, и применена к библиотеке hahn, которая является базовым фундаментом для построение аксиоматических моделей памяти, с целью выяснить, как много доказательств возможно с ее помощью упростить.

Был предложен и реализован метод автоматизации доказательств утверждений, которые изначально не были сформулированы в сигнатуре *KAT*, но могут быть в ней переформулированы. Частично были преодолены проблемы с производительностью нового подхода.

Результаты показывают, что новый подход позволяет сократить существенное количество доказательств, что свидетельствует о перспективности применения теории Алгебр Клини для упрощения разработки и исследования моделей памяти.

При этом большую часть доказательств упростить не удалось из-за невозможности выразить их утверждения с помощью сигнатуры *KAT*. Но непрекращающиеся исследования оставляют надежду на то, что в скором времени найдутся более простые и практичные в формализации доказательства разрешимости других расширений алгебры Клини, которые позволят реализовать их в Coq и значительно расширить возможности предложенного подхода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 A Coq library for lists and relations [Электронный ресурс] / V. Vafeiadis [et al.]. — 2020. — URL: https://github.com/vafeiadis/hahn.

2 An Axiomatic Memory Model for POWER Multiprocessors / S. Mador-Haim [et al.] // Proceedings of the 24th International Conference on Computer Aided Verification. — Berkeley, CA : Springer-Verlag, 2012. — P. 495–512. — (CAV'12). — ISBN 9783642314230. — DOI: 10.1007/978-3-642-31424-7_36.

3 *Brady E. C.* Idris — Systems Programming Meets Full Dependent Types // Proceedings of the 5th ACM Workshop on Programming Languages Meets Program Verification. — Austin, Texas, USA : Association for Computing Machinery, 2011. — P. 43–54. — (PLPV '11). — ISBN 9781450304870. — DOI: 10.1145/1929529.1929536.

4 *Braibant T., Pous D.* Deciding Kleene Algebras in Coq // Logical Methods in Computer Science. — 2012. — Vol. 8, no. 1. — DOI: 10.2168/LMCS-8(1:16)2012.

5 *Brunet P.* Algebras of Relations : from algorithms to formal proofs : Theses : 2016LYSE1198 / Brunet Paul. — Université de Lyon, 10/2016. — URL: https://tel.archives-ouvertes.fr/tel-01455083.

6 *Brunet P.* Reversible Kleene lattices // 42nd International Symposium on Mathematical Foundations of Computer Science. Vol. 83 / ed. by K. G. Larsen, H. L. Bodlaender, J.-F. Raskin. — Kim G. Larsen. Aalborg, Denmark : Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 08/2017. — 66:1–66:14. — (Leibniz International Proceedings in Informatics (LIPIcs)). — DOI: 10.4230/LIPIcs.MFCS.2017.66. — URL: https://hal.archives-ouvertes.fr/hal-01474911.

7 *Brunet P.* A Complete Axiomatisation of a Fragment of Language Algebra // 28th EACSL Annual Conference on Computer Science Logic (CSL 2020). Vol. 152 / ed. by M. Fernández, A. Muscholl. — Dagstuhl, Germany : Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. — 11:1–11:15. — (Leibniz International Proceedings in Informatics (LIPIcs)). — ISBN 978-3-95977-132-0. — DOI: 10.4230/LIPIcs.CSL.2020.11. — URL: https://drops.dagstuhl.de/opus/volltexte/2020/11654.

8 *Brunet P.*, *Pous D.* Petri automata for Kleene allegories // Logic in Computer Science. — Kyoto, Japan : IEEE, 07/2015. — P. 68–79. — DOI: 10.1109/LICS.2015.17. — URL: https://hal.archives-ouvertes.fr/hal-01073936.

9 *Cohen E.* Hypotheses in kleene algebra //. — 1993.

10 *Cohen E.*, *Kozen D.*, *Smith F.* The complexity of Kleene algebra with tests : tech. rep. / Computer Science Department, Cornell University. — 07/1996. — TR96–1598.

11 *Coq T.* The Coq Proof Assistant : Reference Manual : Version 7.2 : tech. rep. / INRIA. — 02/2002. — P. 290. — RT–0255. — URL: https://hal.inria.fr/inria-00069919.

12 Formalizing the LLVM Intermediate Representation for Verified Program Transformations / A. Bb [et al.] //. Vol. 47. — 01/2012. — P. 427–440. — DOI: 10.1145/2103621.2103709.

13 *Fürer M.* The complexity of the inequivalence problem for regular expressions with intersection // Automata, Languages and Programming / ed. by J. de Bakker, J. van Leeuwen. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1980. — P. 234–245. — ISBN 978-3-540-39346-7.

14 *Hardin C.*, *Kozen D.* On the elimination of hypotheses in Kleene algebra with tests. Rap. tech. 2002-1879 // Computer Science Department, Cornell University. — 2002.

15 *Kozen D.* A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events // Information and Computation. — 1994. — Vol. 110, no. 2. — P. 366–390. — ISSN 0890-5401. — DOI: https://doi.org/10.1006/inco.1994.1037. — URL: http://www.sciencedirect.com/science/article/pii/S0890540184710376.

16 *Kozen D.*, *Smith F.* Kleene algebra with tests: Completeness and decidability // Proc. 10th Int. Workshop Computer Science Logic (CSL'96). Vol. 1258 / ed. by D. van Dalen, M. Bezem. — Utrecht, The Netherlands : Springer-Verlag, 09/1996. — P. 244–259. — (Lecture Notes in Computer Science).

17 *Leroy X.* Formal Verification of a Realistic Compiler // Commun. ACM. — New York, NY, USA, 2009. — July. — Vol. 52, no. 7. — P. 107–115. — ISSN 0001-0782. — DOI: 10.1145/1538788.1538814.

18 *Mahboubi A.*, *Tassi E.* Canonical Structures for the working Coq user // ITP 2013, 4th Conference on Interactive Theorem Proving. Vol. 7998 / ed. by S. Blazy, C. Paulin, D. Pichardie. — Rennes, France : Springer, 07/2013. — P. 19–34. — (LNCS). — DOI: 10.1007/978-3-642-39634-2_5. — URL: https://hal.inria.fr/hal-00816703.

19 *Nipkow T.*, *Paulson L. C.*, *Wenzel M.* Isabelle/HOL: a proof assistant for higher-order logic. Vol. 2283. — Springer Science & Business Media, 2002.

20 *Norell U.* Towards a practical programming language based on dependent type theory. Vol. 32. — Citeseer, 2007.

21 *Paulin-Mohring C.* Introduction to the coq proof-assistant for practical software verification // LASER Summer School on Software Engineering. — Springer. 2011. — P. 45–95.

22 *Podkopaev A.*, *Lahav O.*, *Vafeiadis V.* Bridging the Gap Between Programming Languages and Hardware Weak Memory Models // CoRR. — 2018. — Vol. abs/1807.07892. — arXiv: 1807.07892. — URL: http://arxiv.org/abs/1807.07892.

23 *Pous D.* Kleene Algebra with Tests and Coq Tools for while Programs // Interactive Theorem Proving / ed. by S. Blazy, C. Paulin-Mohring, D. Pichardie. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. — P. 180–196. — ISBN 978-3-642-39634-2.

24 Repairing Sequential Consistency in C/C++11 / O. Lahav [et al.] // Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. — Barcelona, Spain : Association for Computing Machinery, 2017. — P. 618–632. — (PLDI 2017). — ISBN 9781450349888. — DOI: 10.1145/3062341.3062352.

25 *Stockmeyer L. J.*, *Meyer A. R.* Word Problems Requiring Exponential Time(Preliminary Report) // Proceedings of the Fifth Annual ACM Symposium on Theory of Computing. — Austin, Texas, USA : Association for Comput-

ing Machinery, 1973. — P. 1–9. — (STOC '73). — ISBN 9781450374309. — DOI: 10.1145/800125.804029.