**УДК 004.89**

# СИНТАКСИКО-СЕМАНТИЧЕСКИЙ АНАЛИЗ ДОВЕРЕННЫХ БИБЛИОТЕК С ИСПОЛЬЗОВАНИЕМ ЛОГИКИ ХОАРА И ТЕОРИИ ТИПОВ

Шиянов К.Е., студент гр. ИИм-231, II курс
Научный руководитель: Башкирцева Е. В., старший преподаватель
Кузбасский государственный технический университет
имени Т.Ф. Горбачева, г. Кемерово

С увеличением роли машинного обучения в критически важных отраслях – от здравоохранения до финансовых технологий – возрастает необходимость в использовании доверенных библиотек, обеспечивающих формально верифицированную корректность и безопасность вычислений. Одним из наиболее надёжных путей достижения этой цели является применение формальных методов, в частности логики Хоара и теории типов. Синтаксико-семантический анализ, основанный на этих подходах, позволяет не только описывать ожидаемое поведение функций и структур, но и обеспечивать доказуемую корректность исполнения программного кода на всех этапах его жизненного цикла [1].

Логика Хоара, с момента её появления, зарекомендовала себя как мощный инструмент верификации императивных программ. Её основное выражение – тройка Хоара {P} C {Q} – позволяет формализовать утверждение о том, что если перед выполнением команды C выполняется предусловие P, то после её выполнения будет выполнено постусловие Q. Такая форма утверждения позволяет логически аннотировать любую функцию библиотеки, определяя её ожидаемое поведение на корректных входных данных. В случае доверенных ML-библиотек, это позволяет точно зафиксировать, при каких условиях модельное преобразование допустимо, каковы ограничения на входные тензоры, и какое поведение гарантировано при корректном использовании интерфейса. Например, спецификация функции нормализации матрицы может включать требования об обратимости, положительности определителя или ограничениях на разрядность, в зависимости от контекста применения.

Пусть функция f реализована в библиотеке как трансформация входных данных в результат $x \in Dy = f(x)D\varphi(x)$а $\psi(x,y)yAx = b$, где $\varphi(x)$ – допустимое пространство входов, и пусть $D\varphi(x)$ – логическое предусловие корректности (например, «матрица x обратима»), – постусловие (например,

«результат – это решение системы»). Тогда корректность функции в логике Хоара формализуется как: [2].

$$\forall x \in D \quad \big(\varphi(x) \Rightarrow \psi\big(x, f(x)\big)\big)$$

Теория типов, в свою очередь, предлагает ещё более мощный уровень верификации за счёт включения семантики программы в саму структуру типов. В зависимости от степени выразительности системы типов – от простых примитивов до зависимых типов – можно описывать и проверять сложнейшие свойства программ до момента их выполнения. В языках с зависимыми типами (Idris, Agda, Coq) тип может включать предикаты, доказывающие, например, что матрица, подаваемая на вход функции, имеет строго положительное определение, или что два тензора совместимы по размерности. Таким образом, компилятор не только проверяет корректность синтаксиса, но и обеспечивает доказательство семантической состоятельности всей цепочки вызовов функций, исключая широкий класс ошибок на стадии компиляции. Это особенно важно для библиотек, обслуживающих системы с повышенными требованиями к надёжности, включая медицинскую диагностику, автономное вождение, государственные системы принятия решений.

Интеграция логики Хоара и типовых систем формирует синтаксико-семантическое ядро доверенного программного обеспечения. В современной практике всё большее распространение получают гибридные подходы, включающие уточнённые типы (refinement types), где каждый тип снабжён логическим предикатом, задающим ограничения на допустимые значения. Так, в языке LiquidHaskell можно описать тип положительного целого числа как {v: Int | v > 0}, после чего функции, принимающие этот тип, автоматически получают верификацию на предмет использования только корректных аргументов. Это позволяет не только описывать API библиотеки в логических терминах, но и формировать контракт, исполняемый и проверяемый на уровне системы типов. В совокупности это создаёт возможность разработки самодоказуемых библиотек – таких, в которых соблюдение всех логических инвариантов обеспечивается самими средствами языка и компилятора, без необходимости ручного тестирования или рецензирования [3].

Синтаксико-семантический анализ доверенных библиотек в этом контексте приобретает форму автоматизированного процесса, включающего этапы логической аннотации, типизации, генерации доказательств и верификации. На этапе аннотации каждой функции сопоставляется логическая спецификация, описывающая входные и выходные условия. На этапе типизации уточняются условия применимости функций, включая допустимые значения, связи между параметрами и ограничения на изменяемость

состояний. Генерация доказательств производится либо вручную (в интерактивных системах вроде Coq), либо автоматически (в системах с SMT-поддержкой, таких как Why3, F*, LiquidHaskell). Финальным этапом является формальная проверка всей системы, включая анализ зависимости между модулями и поиск потенциальных конфликтов между локальными инвариантами [4].

Особое значение подобный анализ приобретает при разработке доверенных компонентов в условиях внешней регуляции, когда библиотека должна соответствовать нормативам (например, ISO/IEC 25010, GDPR, HIPAA) или использоваться в системах, где даже минимальная ошибка недопустима. В таких случаях наличие формальной спецификации и доказательств корректности становится необходимым условием сертификации программного продукта. Синтаксико-семантический анализ с использованием логики Хоара и теории типов позволяет автоматизировать этот процесс и обеспечить высокий уровень доверия к системе как со стороны разработчиков, так и со стороны конечных пользователей или регуляторов.

Таким образом, объединение логики Хоара и теории типов в рамках синтаксико-семантического анализа представляет собой мощный подход к построению доверенных библиотек машинного обучения. Он позволяет описывать поведение программ строго математически, исключать критические ошибки на стадии компиляции, и, главное, формировать воспроизводимые и проверяемые гарантии корректности – неотъемлемое требование к системам, использующимся в чувствительных и высокорисковых сферах [5, 6].

**Список литературы:**

1. Pierce, B. C. Types and Programming Languages / B. C. Pierce. – Cambridge : MIT Press, 2002. – 623 p.

2. Reynolds, J. C. Separation Logic: A Logic for Shared Mutable Data Structures / J. C. Reynolds // Proceedings of the IEEE Symposium on Logic in Computer Science. – 2002. – P. 55–74. – DOI 10.1109/LICS.2002.1029817.

3. Harper, R. Practical Foundations for Programming Languages / R. Harper. – 2nd ed. – Cambridge : Cambridge University Press, 2016. – 513 p. – DOI 10.1017/CBO9781316576892.

4. Winskel, G. The Formal Semantics of Programming Languages: An Introduction / G. Winskel. – Cambridge : MIT Press, 1993. – 361 p.

5. Свидетельство о государственной регистрации программы для ЭВМ № 2026660990 Российская Федерация. Программа для выполнения автоматической аппроксимации оптимальных траекторий на основе метода Декартовых вариаций : заявл. 07.04.2026 : опубл. 16.04.2026 / А. В.

Протодьяконов, Р. В. Майтак ; заявитель Кузбасский государственный технический университет имени Т. Ф. Горбачева. – EDN LFGAYW.

6. Pierce, B. C. Software Foundations. Vol. 1: Logical Foundations / B. C. Pierce, A. Casinghino, M. Greenberg [et al.] [Электронный ресурс]. – URL: https://softwarefoundations.cis.upenn.edu/ (дата обращения: 09.06.2025).