Коцептуальна модель системи доведення теорем

Maksym Sokhatskyi [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnical Institute
18 травна 2018 року

Анотація

Дана робота презентує концептуальну модель системи доведення теорем, яка предствалена як система: 1) числень разом з їх категорними моделями [статика], тобто властивостями та семантикою; 2) програм та базових бібліотек [екземпляри], для програмного забезпечення; 3) програмних трансформацій між різними численнями [динаміка], компіляція, екстракт програм в інші мови програмування, оптимізація, тощо. Значна увага приділена таким проблемам як: 1) масштабування системи доведення теорем на довільні числення; 2) декомпозиції системи мовні рівні. В основу роботи покладені сучасні фундаментальні дослідження з кубічної теорії типів, що дозволяє формулювати і доводити гомотопічні моделі та їх властивості.

Зміст

1 Вступ

1.1 Актуальність роботи

Ціна помилок в індустрії надзвичайно велика. Наведемо відомі приклади: 1) Mars Climate Orbiter (1998), помилка невідповідності типів британської метричної системи, коштувала 80 мільйонів фунтів стерлінгів. Невдача стала причиною переходу NASA повністю на метричну систему в 2007 році. 2) Ariane Rocket (1996), причинан катастрофи – округлення 64-бітного дійсного числа до 16-бітного. Втрачені кошти на побудову ракети та запуск 500 мільйонів 3) Помилка в FPU в перших Pentium (1994), збитки на 300 мільйонів. 4) Помилка в SSL (heartbleed), оцінені збитки у розмірі 400 мільйонів. 5) Помилка у логіці бізнес-контрактів EVM та DAO (неконтрольована рекурсія), збитки 50 мільйонів. Більше того, і найголовніше, помилки у програмному забезпеченні можуть коштувати життя людей.

1.2 Верифікація та валідація

Для унеможливлення помилок на виробництві застосовуються різні методи формальної верифікації. Формальна верифікація — доказ, або заперечення відповідності системи у відношенні до певної формальної специфікації або характеристики, із використанням формальних методів математики.

Дамо основні визначення згідно з міжнародними нормами (IEEE, ANSI)[1] та у відповідності до вимог Європейського Аерокосмічного Агенства[2]. У відповідності до промислового процессу розробки, верифікація та валідація програмного забезпечення є частиною цього процесу. Програмне забезпечення перевіряється на відповідність функціональних властивостей згідно вимог.

Процес валідації включає в себе перегляд (code review), тестування (модульне, інтеграційне, властивостей), перевірка моделей, аудит, увесь комплекс необхідний для доведення, що продукт відповідає вимогам висунутим при розробці. Такі вимоги формуються на початковому етапі, результатом якого є формальна специфікація.

1.3 Формальна специфікація

Для спрощення процесу верифікації та валідації застосовується математична техніка формалізації постановки задачі — формальна специфікація. Формальна специфікація — це математична модель, створена для опису систем, визначення їх основних властивостей, та інструментарій для перевірки властивостй (формальної верифікаціїї) цих систем, побудованих на основі формальної специфікації.

[1] IEEE Std 1012-2016 — V&V Software verification and validation

[2] ESA PSS-05-10 1-1 1995 – Guide to software verification and validation

Існують два фундаментальні підходи до формальних специфікацій: 1) Аглебраїчний підхід, де система описується в термінах операцій, та відношень між ними, та 2) Модельно-орієнотований підхід, де модель створена конструктивними побудовами, як то на базі теорії множин, чи інкаше, а системні операції визначаються тим, як вони змінюють стан системи. Також були створені сімейства послідованих та розподілених мов.

Найбільш стандартизована та прийнята в обсласті формальної верифікації — це нотація Z[3] (Spivey, 1992), приклад модельно-орієнтовоної мови Назавана у честь Ернеста Цермело, роботи якого мали вплив на фундамент математики та аксіоматику теорії множин. Саме теорія множин, та логіка предикатів першого порядку є теорією мови Z.

Інша відома мова формальної специфікації як стандарт для моделювання розподілених систем, таких як телефонні мережі та протоколи, це LOTOS[4] (Bolognesi, Brinksma, 1987), як приклад алгебраїчного підходу. Ця мова побудована на темпоральних логіках, та поведінках залежниих від спостережень. Інші темпоральні мови специфікацій, які можна відзначити тут — це TLA+[5], CSP (Hoare, 1985), CCS[6] (Milner, 1971), Actor Model, Reactive Streams, etc.

2 Формальні методи верифікації

2.1 Історія систем доведення теорем

Перші спроби пошуку формального фундаменту для теорії обчислень були покладені Алонзо Черчем та Хаскелем Каррі у 30-х роках 20-го століття. Було запропоноване лямбда числення як апарат який може замінити класичну теорію множин та її аксіоматику, пропонуючи при цьому обчислювальну семантику. Пізніше в 1958, ця мова була втілена у вигляді LISP лауреатом премії тюрінга Джоном МакКарті, який працював в Прінстоні. Ця мова була побудована на конструктивних примітивах, які пізніше виявилися компонентами індуктивних конструкцій та були формалізовані за допомогою теорії категорій Вільяма Лавіра. Окрім LISP, нетипізоване лямбда числення маніфестується у такі мови як Erlang, JavaScript, Python. До цих пір нетипізоване лямбла числення є одною з мов у які робиться конвертація доведених программ (екстракція).

Перший математичний прувер AUTOMATH (і його модифікації AUT-68 та AUT-QE), який був написаний для комп'ютерів розроблявся під керівництвом де Брейна, 1967. У цьому прувері був квантор загально-

[3] ISO/IEC 13568:2002 — Z formal specification notation

[4] ISO 8807:1989 — LOTOS — A formal description technique based on the temporal ordering of observational behaviour

[5] The TLA+ Language and Tools for Hardware and Software Engineers

[6] J.C.M. Baeten. A Brief History of Process Algebra.

сті та лямбда функція, таким чином це був перший прувер побудрваний на засадах ізоморфізма Каррі-Говарда-Ламбека.

ML/LCF або метамова і логіка обчислювальних функцій були наступним кроком до осягнення фундаментальної мови простору, тут вперше з'явилися алебраїчні типи даних у вигляді індуктивних типів, поліноміальних функторів або термінованих (well-founded) дерев. Роберт Мілнер, асистований Морісом та Н'юві розробив Метамову (ML), як інструмент для побудови прувера LCF. LCF був основоположником у родині пруверів HOL88, HOL90, HOL98 та останньої версії на даний час HOL/Isabell. Пізніше були побувані категорні моделі Татсоя Хагіно (CPL, Японія) та Робіна Кокета (Charity, Канада).

У 80-90 роках були створені інші системи автоматичного доведення теорем, такі як Mizar (Трибулєк, 1989). PVS (Оур, Рушбі, Шанкар, 1995), ACL2 на базі Common Lisp (Боєр, Кауфман, Мур, 1996), Otter (МакКюн, 1996).

2.2 Системи моделювання та верифікації

Можна виділити три підходи до верифікації. Перший застосовується де вже є певна програма написана на певній мові програмування і потрібно довести ізоморфність цієї програми до доведеної моделі. Ця задача вирішується у побудові теоретичної моделі для певної мови програмування, потім програма на цій мові переводиться у цю теоретичну модель і доводить ізоморфізм цієї програми у побудованій моделі до доведеної моделі. Приклади таких систем та піходів: 1) VST (CompCert, сертифікація С програм), 2) NuPRL (Cornell University, розподілені системи, залежні типи), 3) TLA+ (Microsoft Reseach, Леслі Лампорт), 4) Twelf (для верифікації мов програмування), 5) SystemVerilog (для ч'програмного та апаратного забезпечення).

2.3 Мови з залежними типами

Другий підхід можна назвати підходом вбудованих мов. Компілятор основої мови перевіряє модель закодовану у ній же. Можливо моделювання логік вищого порядку, лінійних логік, модальних логік, категорний та гомотопічних логік. Процес специфікації та верифікації відбувається в основній мові, а сертифіковані програми автоматично екстрагуються в довільні мови. Приклади таких систем: 1) Coq побудована на мові OCaml від науково-дослідного інституту Франції INRIA; 2) Agda побудовані на мові Haskell від шведського інституту технологій Чалмерс; 3) Lean побудована на мові C++ від Microsoft Research та Університету Каргені-Мелона; 4) Idris подудована на мові Haskell Едвіна Бреді з шотландського Університету ім. св. Андрія; 5) F* – окремий проект Microsoft Research.

2.4 Системи автоматичного доведення теорем

Третій підхід полягає в синтезі конструктивного доведення для формальної специфікації. Це може бути зроблено за допомогою асистентів доведення теорем, таких як HOL/Isabell, Coq, ACL2, або систем розв'язку задач виконуваності формул в теоріях (Satisfiability Modulo Theories, SMT).

2.5 Обмеження

Незалежно від піходу до верифіції, формальна верифікація неможлива, якщо мова програмування моделі формально не визначена. Це означає шо значна міра програмного забезпечення може бути автоматично верифікована тільки для тих мов, формальні моделі яких побудовані, на даний момент це тільки мова C. Більше того, не завжди можна також формально довести те, що програма завершиться, потрібно звужувати клас програм, якщо формальні специфікації містять такі властивості.

3 Формалізована постановка задачі

3.1 Об'єкт дослідження

Об'єктом дослідження данної роботи є: 1) системи верифікації програмного забезпечення; 2) системи доведення теорем; 3) мови програмування; 4) операційні системи, які виконують обчислення в реальному часі; 3) їх поєднання, побудова формальної системи для унифікованого середовища, яке поєднує середовище виконання та систему верифікації у єдину систему мов та засобів.

3.2 Предмет дослідження

Предметом дослідження такої системи мов є теорія типів, яка вивчає обчислювальні властивості мов. Теорія типів виділилася в окрему науку завдяки роботам Бішопа та Мартіна-Льофа, як запит на вакантне місце у трикутнику теорій, які відповідають ізоморфізму Каррі-Говарда-Ламбека (Логіки, Мови, Категорії), або інтерпретації Брауера-Гейтінга-Колмогорова. Інші дві це: теорія категорій та логіки вищих порядків.

3.3 Мотивація

Одна з причина низького рівня впровадження у виробництво систем верифікації – це висока складність таких систем. Складні системи верифікуються складно. Ми хочемо запропонувати спрощений підхід до верифікації – оснований на концепції компактних та простих мовних

ядер для створення специфікацій, моделей, перевірки моделей, доведення теорем у теорії типів з кванторами.

3.4 Завдання дослідження

Завдання дослідження полягає у: 1) декомпозіції типових систем, побудові єдиної категорії мов програмуваня, де об'єкти — це мови програмування, а морфізми — це мовні перетворення; 2) виділенні мінімальної конфігурації яка дозволяю доводити властивості програмного забезпечання на усіх етапах від проектування до виконання; 3) побудові гнучкої поліморфної системи мовних розширень до мінімального ядра для забезпечення потреб масшабування системи; 4) реалізації мінімального ядра системи та вищих мовних розширень разом із базовими бібліотеками.

3.5 Структура дослідження

Структура дослідження відображає компонентну модель системи доведення теорем: 1) інтерпретатор безтипового лямбда числення; 2) компактне ядро — система з однією аксіомою; 3) мова з індуктивними типами; 4) мова з гомотопічним інтервалом $[\mathbf{0}, \mathbf{1}]$; 5) уніфікована базова бібліотека.

4 Мінімальна верифікована система програмування

Пошук мінімальної конфугурації мов визначається мотивацією дослідження, як запорука прозорості та адаптованості до безкоштовної верифікації або верифікації малими зусиллями.

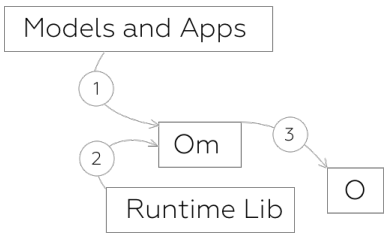

Рис. 1: The Stages of Model Verifications

4.1 Інтерпретатор безтипового лямбда числення

Чому саме інтерпретатор? Як можна побудувати ефективний інтерпретатор? Нормальний та зворотній порядок бета редукції. Векторизація

обчислень.

4.2 Чиста система з однією аксіомою

Чиста система. Операційна семантика. Категорна семантика залежної теорії.

5 Вищі мови програмування

Математична індукція. Топологічні та гомотопічні моделі.

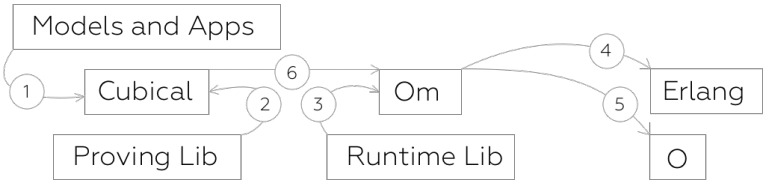

Рис. 2: The Stages of Model Verifications

5.1 Індуктивні мови

Категорна семантика індуктивних типів.

5.2 Гомотопічні мови

Розшарування CCHM. Кубічна система типів.

5.3 Мови числення процесів

Середовище виконання. Бісімуляція.

6 Результати дослідження

6.1 Інтерпретатор нетипізованого λ-числення

6.2 Чиста мова Henk

6.3 Гомотопічна мова Anders

6.4 Гомотопічна базова бібліотека

7 Висновки

[?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]
[?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]
[?] [?] [?] [?]