synrc research center s.r.o.
Roháčova 141/18, Praha 3 13000, Czech Republic

Системна інженерія та верифікація уніфікованого обчислювального середовища

System Engineering and Verification of Unified Execution Environment

Павло Маслянко, Київський Політехнічний Інститут

Максим Сохацький, Synrc Research Center

Листопад 2017

Зміст

1 Вступ

1.1 Системна інженерія та верифікація

Протягом історії обчислювальної техніки було створено різні класи та способи обчислень, різні тоеорії та підходи до програмування таких систем, різні класи систем програмування. Зараз уже стало зрозумілим, що інженіринг систем які не піддаються до верифікації формальними методами не може бути застосований у галузях де вимоги до якості особливо підвищені, як то космонавтика, енергетика та фінанси.

Об'єктом дослідження данної роботи є системи верифікації програмного забезпечення та операційні системи яки виконують обчислення в реальному часі, їх поєднання та побудова формальної системи для унифікованого середовища, яке поєднує середовище виконання та систему верифікації у єдину систему мов.

Предметом дослідження такої системи мов є теорія типів, яка вивчає обчислювальні властивості мов. Теорія типів виділилася в окрему науку Мартіном Льофом як запит на ваканте місце у трикутнику теорій, які відповідають ізоморфізму Каррі-Говарда (Логіки, Мови, Категорії). Інші дві це: теорія категорій та логіка вищих порядків. Сама система доведення теорем є логікою, або аспектом логіки у трикутнику. Імплементація мови програмування, яка релізує логічну семантику здійснюється завдяки теорії типів. Формалізація методів відбувається завдяки теорії категорій, яка є абстрактною алгеброю фунцій, метематичним інструментом для формалізації мов програмування та довільних математичних теорій які описуються логіками вищих порядків.

Завдання цього дослідження є побудова єдиної системи, яка поєднує середовище викодання та систему верифікації програмного забезпечення. Це прикладне дослідження, яке є фьюжином фундаментальної математики та інженерних систем з формальними методами вериціфікації. Методи цього дослідження є суто теоретичними.

1.2 Історія систем верифікації

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 таких примітивах, як: cons, nil, eq, atom, car, cdr, lambda, apply and id. Направді це уривки індуктивних конструкцій які були структуровані пізніше і формалізовані за допомогою теорії категорій Вільяма Лавіра. До цих пір нетипізоване лямбла числення є одною з мов у які робиться екстракт з сучасних пруверів. Окрім LISP, нетипізоване лямбда числення маніфестується у такі мови як Erlang, JavaScript, Python.

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

ML/LCF або метамова і логіка обчисльювальних функцій був наступник крок до осягнення фундаментальної мови простору, тут впреше з'явилися алебраїчні типи даних у вигляді індуктивних типів, поліноміальних функторів або терміновані (well-founded) дерев. Пізніше були побувані категорні моделі Татсоя Хагіно (CPL) то Крокетом (Charity). 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.

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).

1.3 Методи верифікації

Можна виділити два 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. VST (CompCert, сертифікація Ci-програм), NuPRL (Cornell University, розподілені системи, залежні типи), TLA+ (Microsoft Reseach, Леслі Лампорт), Twelf (для верифікації мов програмування).

Інший підхід можна назвати підходом вбудованих DSL. Усе моделювання відбувається в основній мові, а сертифіковані програми автоматично екстрактяться в довільні мови. Приклади таких систем: Coq побудована на мові OCaml від науково-дослідного інституту Франції INRIA; Agda побудовані на мові Haskell від шведського інституту технологій Чалмерс; Lean побудована на мові C++ від Microsoft Research та Університету Каргені-Мелона; Idris подудована на мові Haskell Едвіна Бреді з шотландського Університету ім. св. Андрія; F* – окремий проект Microsoft Research.

2 Метематичне забезпечення

2.1 Теорія категорій

Теорія категорій широко застосовується як інструмент для математиків у тому числі і при аналізі програмного забезпечення. Теорію категорій можна вважати наближенням абстрактної алгебри функцій. Дамо конструктивне визначення категорії. Категорії (програми) визначаються переліком своїх об'єктів (типів) та своїх морфізмів (функцій), а також бінарною операцією композиції, що задовольняє закону асоціативності, та з тотожнім морфізмом (тотжньою функцією — одиницею) який існує для кожного об'єкту (типу) категорії. Аксіоми формації об'єктів не приводяться та авто-постулуються в нижніх аксіомах. Аксіома формації морфізмів буде даватися як введеня експоненти після визначення декартового добутку. Поки що тут буде визначатися тільки композиція морфізмів. Об'єкти A та B морфізма $f : A \to B$ називаються домен та кодомен відповідно.

Інтро аксіоми – асоціативність композиції та права і ліва композиції одниці показують, що категорії є типизованими моноїдами, що складаються з морфізмів та операції композиції. Є різні мови, у тому числі і графічні, представлення категорної семантики, однак у цій роботі ми будемо використовувати теоретико-логічні формулювання.

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash g : B \to C}{\Gamma \vdash g \circ f : A \to C}$$

$$\frac{}{\Gamma \vdash id_A : A \to A}$$

$$\frac{\Gamma \vdash f : B \to A \qquad \Gamma \vdash g : C \to B \qquad \Gamma \vdash h : D \to C}{\Gamma \vdash (f \circ g) \circ h = f \circ (g \circ h) : D \to A}$$

$$\frac{\Gamma \vdash f : A \to B}{\Gamma \vdash f \circ id_A = f : A \to B}$$

$$\frac{\Gamma \vdash f : A \to B}{\Gamma \vdash id_B \circ f = f : A \to B}$$

Композиція показує можливість звязувати область значень попереднього обчислення (кодомен) та область визначення наступного обчислення (домен). Композиція є фундаментальною властивістю морфізмів.

1. $A : *$
2. $A : * \ , \ B : * \implies f : A \to B$
3. $f : B \to C \ , \ g : A \to B \implies f \circ g : A \to C$
4. $(f \circ g) \circ h = f \circ (g \circ h)$
5. $A \implies id : A \to A$
6. $f \circ id = f$
7. $id \circ f = f$

2.2 Алгебраїчні типи даних

Після операції композиції, як способу конструювання нових об'єктів за допомогою морфізмів далі йде операція конструювання добутка двох об'єктів певної категорії, разом з добутком морфізмів зі спільним доменом, необхідних для визначення декартового добутка $A \times B$.

Це є внутрішня мова декартової категорії, у якій для будь яких двох доменів існує їх декартова сума (кодобутку) та декартовий добуток (косума, кортеж), за допомогою яких конструюються суми-протоколи та добутки-повідмоення, а також існує $\bot$ тип-термінал, та $\top$ тип-котермінал. Термінальними типами зручно термінувати рекурсивні типи даних, такі як списки. Ми будемо розглядати тільки категорії які маються добутки та суми.

Добуток має природні елімінатори π зі спільним доменом, які є морфізмами-проекціями об'єктів добутка. Сума має оберненені елімінатори σ зі спільним кодоменом. Як видно добуток є дуальний до суми з точністю до направлення стрілок, таким чином елімінатори π та σ є оберненими, тобто $\pi \circ \sigma = \sigma \circ \pi = id$.

$$\frac{\Gamma\ x : A \times B}{\Gamma \vdash \pi_1\ : A \times B \to A; \Gamma \vdash \pi_2\ : A \times B \to B}$$

$$\frac{}{\Gamma \vdash\ \top}$$

$$\frac{\Gamma \vdash\ a : A \qquad \Gamma \vdash\ b : B}{\Gamma \vdash\ (a, b) : A \times B}$$

$$\frac{\Gamma \vdash\ a : A \qquad \Gamma \vdash\ b : B}{\Gamma \vdash a \mid b : A \otimes B}$$

$$\frac{}{\Gamma \vdash\ \bot}$$

$$\frac{\Gamma\ x : A \otimes B}{\Gamma \vdash \sigma_1 : A \to A \otimes B; \Gamma \vdash \sigma_2 : B \to A \otimes B}$$

Також додамо тут аксіому множення морфізмів, яка випливає з визначення добутка, яка необхідна для забезпечення аплікативного програмування.

$$\frac{\Gamma \vdash\ f : A \to B \qquad \Gamma \vdash\ g : A \to C \qquad \Gamma \vdash\ B \times C}{\Gamma \vdash\ \langle f, g \rangle : A \to B \times C}$$

$$\pi_1 \circ \langle f, g \rangle = f$$
$$\pi_2 \circ \langle f, g \rangle = g$$
$$\langle f \circ \pi_1, f \circ \pi_2 \rangle = f$$
$$\langle f, g \rangle \circ h = \langle f \circ h, g \circ h \rangle$$
$$\langle \pi_1, \pi_2 \rangle = id$$

2.3 Лямбда числення

Будучи внутрішньою мовою декартово-замкненої категорії лямбда числення окрім змінних та констант у вигляді термів пропонує операції абстракції та аплікації, що визначає достатньо лаконічну та потужну структуру обчислень з функціями вищих порядків, та метатипизаціями, такими як System F, яка була запропонована вперше Робіном Мілнером в мові ML, та зараз присутня в більш складних, таких як System Fω, системах Haskell та Scala.

Щоб пояснити функції з категоріальної точки зору потрібно пояснити категоріальні експоненти $f : A^B$, які є аналогами фукціональних просторів $f : A \to B$. Так як ми вже визначили добутки та термінали, то ми можемо визначити і експоненти, опускаючи усі категоріальні подробиці ми визначимо конструювання функції (операція абстракції), яка параметризується змінною x у середовищі Γ; та її елімінатора – операції аплікації функції до аргументу. Так визначаєьтся декартово-замкнена категорія. Визначається також рекурсивний механізм виклику функції з довільною кількістю аргументів.

$$\frac{\Gamma x : A \vdash M : B}{\Gamma \vdash \lambda\ x\ .\ M : A \to B}$$

$$\frac{\Gamma\ f : A \to B \qquad \Gamma\ a : A}{\Gamma \vdash apply\ f\ a\ : (A \to B) \times A \to B}$$

$$\frac{\Gamma \vdash f : A \times B \to C}{\Gamma \vdash curry\ f : A \to (B \to C)}$$

$$apply \circ \langle (curry\ f) \circ \pi_1, \pi_2 \rangle = f$$
$$curry\ apply \circ \langle g \circ \pi_1, \pi_2 \rangle) = g$$
$$apply \circ \langle curry\ f, g \rangle = f \circ \langle id, g \rangle$$
$$(curry\ f) \circ g = curry\ (f \circ \langle g \circ \pi_1, \pi_2 \rangle)$$
$$curry\ apply = id$$

Об'єкти : $\bot \mid\ \to\ \mid \times$
Морфізми : $id \mid f \circ g \mid \langle f, g \rangle \mid apply \mid \lambda \mid curry$

2.4 Числення процесів

Теорія π-числення процесів Роберта Мілнера є основним формалізмом обчислювальної теорії розподілених систем та її імплементації. З часів виникнення CSP числення розробленого Хоаром, Мілнеру вдалося значно розширити та адаптувати теорію до сучасних телекомунікаційних вимог, як наприклад хендовери в мобільних мережах. Основні теорми в моделі π-числення стосуються непротиречивості та неблокованості у синхронному виконанні мобільних процесів. Так як сучасний Web можно розглядати як телекомунікаційну систему, тому у розробці додатків можна покладатися у тому числі і на такі моделі як π-числення. Також ми анонсуємо процес як фундаменльний тип даних, подібний до функції але який здатний тримати певний стан у вигляді типа коротежа та є морфізмом-одиницею типу свого стану.

$$\frac{\Gamma \vdash E, \Sigma, X \qquad \Gamma \vdash action : \Sigma \times X \to \Sigma \times X}{\Gamma \vdash spawn\ action : \pi_\Sigma}$$

$$\frac{\Gamma \vdash pid : \pi_\Sigma \qquad \Gamma \vdash msg : \Sigma}{\Gamma \vdash join\ msg\ pid : \Sigma \times \pi_\Sigma \xrightarrow{\bullet} \Sigma;\ \Gamma \vdash send\ msg\ pid : \Sigma \times \pi_\Sigma \to \Sigma}$$

$$\frac{\Gamma \vdash L : A + B, R : X + Y \qquad \Gamma \vdash M : A \to X, N : B \to Y}{\Gamma \vdash receive\ L\ M\ N : L \xrightarrow{\bullet} R}$$

Алгебра процесів визначає базові операції мультиплексування двох чи декількох протоколів в рамках одного процесу (добуток), а також паралельного та повністю ізольованого запуску включно зі стеком та областю памяті (сума) на віртуальній машині.

$$\begin{aligned} \oplus &: \pi \parallel \pi \\ \otimes &: \pi \mid \pi \end{aligned}$$

2.5 Інтуітіоністична теорія типів Мартіна Льофа

Системи з залежними типами як верифікаційні математичні формальні моделі для доведення корректності. Система Σ та Π типів, як кванторів існування та узагальнення. Системи Mizar, Coq, Agda, Idris, F*, Lean. Ми будемо використовувати автоматизовану систему теорем Lean від Microsoft Research.

Розбудовуючи певний фреймворк чи систему конструктивними методами так чи інакше доведеться зробити певний вибір у мові та способі кодування. Так при розробці теорії абстрактної алгебри в Coq були використані поліморфні індуктивні структури [**?**]. Однак Agda та Idris використовують для побудови алгебраїчної теорії типи класів, а у Idris взагалі відсутні поліморфірні індуктивні структури та коіндуктивні структури. В Lean теж відсутні коіндуктивні структури проте повністю реалізована теорія HoTT на нерекурсивних поліморфних структурах що обєднує основні чотири класи математичних теорій: логіка, топологія, теорія множин, теорія типів. Як було показано Стефаном Касом [**?**], одна з
стратегій імплементації типів класів — це використання поліморфних структур. Хоча в Lean також підтримуються типи класів нами була вибрана стратегія імплементації нашої теорії з використаннями нерекурсивних індуктивних структур, що дозволить нам оперувати з персистентними структурами на низькому рівні. Крім того такий спосіб кодування ієрархій повністю відповідає семантиці Erlang, де немає типів класів, а дані передаються запаковані в кортежі-структури.

2.6 Логіка та квантори

Далі йдуть квантори $\forall$ та $\exists$ які теж виражаються як конструкції типів:

$$\frac{\Gamma\ x : A \vdash B \qquad \Gamma\ \vdash A}{\Gamma\ \vdash \Pi(x : A)B} \qquad \frac{\Gamma\ \vdash a : A \qquad \Gamma\ x : A \vdash B \qquad \Gamma\ b : B(x = a)}{\Gamma\ \vdash (a, b) : \Pi(x : A)B}$$

$$\frac{\Gamma\ x : A \vdash B \qquad \Gamma\ \vdash A}{\Gamma\ \vdash \Sigma(x : A)B} \qquad \frac{\Gamma\ \vdash a : A \qquad \Gamma\ x : A \vdash B \qquad \Gamma\ b : B(x = a)}{\Gamma\ \vdash (a, b) : \Sigma(x : A)B}$$

$$\frac{\Gamma\ \vdash x : A \qquad \Gamma\ \vdash x' : A}{\Gamma\ \vdash Id_A(x, x')}$$

рефлексивність	:	$Id_A(a, a)$
підстановка	:	$Id_A(a, a') \to B(x = a) \to B(x = a')$
симетричність	:	$Id_A(a, b) \to Id_A(b, a)$
транзитивність	:	$Id_A(a, b) \to Id_A(b, c) \to Id_A(a, c)$
конгруентність	:	$(f : A \to B) \to Id_A(x, x') \to Id_B(f(x), f(x'))$

Список литературы

Category Theory

[1] S.MacLane *Categories for the Working Mathematician* 1972

[2] W.Lawvere *Conceptual Mathematics* 1997

[3] P.Curien *Category theory: a programming language-oriented introduction* 2008

Pure Type Systems

[4] P.Martin-Löf *Intuitionistic Type Theory* 1984

[5] T.Coquand *The Calculus of Constructions.* 1988

[6] E.Meijer *Henk: a typed intermediate language* 1997

[7] H.Barendregt *Lambda Calculus With Types* 2010

Inductive Type Systems

[8] F.Pfenning *Inductively defined types in the Calculus of Constructions* 1989

[9] P.Wadler *Recursive types for free* 1990

[10] N.Gambino *Wellfounded Trees and Dependent Polynomial Functors* 1995

[11] P.Dybjer *Inductive Famalies* 1997

[12] B.Jacobs *(Co)Algebras) and (Co)Induction* 1997

[13] V.Vene *Categorical programming with (co)inductive types* 2000

[14] H.Geuvers *Dependent (Co)Inductive Types are Fibrational Dialgebras* 2015

Homotopy Type Systems

[15] T.Streicher *A groupoid model refutes uniqueness of identity proofs* 1994

[16] T.Streicher *The Groupoid Interpretation of Type Theory* 1996

[17] B.Jacobs *Categorical Logic and Type Theory* 1999

[18] S.Awodey *Homotopy Type Theory and Univalent Foundations* 2013

[19] S.Huber *A Cubical Type Theory* 2015

[20] A.Joyal *What is an elementary higher topos* 2014

[21] A.Mortberg *Cubical Type Theory: a constructive univalence axiom* 2017