Math-for-industry
Education & Research Hub

ISSN2188-1200

2024 年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会

コンピュータによる定理証明支援とその応用

編集：Jacques Garrigue

九州大学マス・フォア・インダストリ研究所

MI Lecture Note Vol.99 : Kyushu University

2024年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会

コンピュータによる定理証明支援とその応用

TPP2024:20th Theorem Proving and Provers Meeting

編集：Jacques Garrigue

About MI Lecture Note Series

The Math-for-Industry (MI) Lecture Note Series is the successor to the COE Lecture Notes, which were published for the 21st COE Program "Development of Dynamic Mathematics with High Functionality," sponsored by Japan's Ministry of Education, Culture, Sports, Science and Technology (MEXT) from 2003 to 2007. The MI Lecture Note Series has published the notes of lectures organized under the following two programs: "Training Program for Ph.D. and New Master's Degree in Mathematics as Required by Industry," adopted as a Support Program for Improving Graduate School Education by MEXT from 2007 to 2009; and "Education-and-Research Hub for Mathematics-for-Industry," adopted as a Global COE Program by MEXT from 2008 to 2012.

In accordance with the establishment of the Institute of Mathematics for Industry (IMI) in April 2011 and the authorization of IMI's Joint Research Center for Advanced and Fundamental Mathematics-for-Industry as a MEXT Joint Usage / Research Center in April 2013, hereafter the MI Lecture Notes Series will publish lecture notes and proceedings by worldwide researchers of MI to contribute to the development of MI.

October 2022
Kenji Kajiwara
Director, Institute of Mathematics for Industry

Computer Assisted Theorem Proving and Applications

MI Lecture Note Vol.99, Institute of Mathematics for Industry, Kyushu University
ISSN 2188-1200
Date of issue: March 17, 2025
Editor: Jacques Garrigue
Publisher:
Institute of Mathematics for Industry, Kyushu University
Graduate School of Mathematics, Kyushu University
Motooka 744, Nishi-ku, Fukuoka, 819-0395, JAPAN
Tel +81-(0)92-802-4402, Fax +81-(0)92-802-4405
URL https://www.imi.kyushu-u.ac.jp/

序 文

ソフトウェア検証・数学の形式化・数学の証明のコンピュータによる検証といった事項をテーマとし, 本研究集会を開催しました. 実問題の中でプログラムとして実装される様々な数学理論, そして, 数理論理学などの数学研究者, プログラム言語理論, ソフトウェア工学などの計算機科学者, さらには, 現実にソフトウェア開発に従事している開発技術者らが約 80 名集い, 25 件の講演発表が行われ, 充実した情報交換と討論を行うことが出来ました.

特に, 定理証明支援系の理論と実践について Pierre-Marie Pédrot 氏 (仏 INRIA) に特別講演を行って頂き, 分野全体の現状と課題を明確にし, そして多数の一般講演者がの定理証明支援系の様々な応用を紹介し, 定理証明器支援系の開発を行うための方向性のアイデアを出し合うことが出来ました.

また, TPPmark2024 と称して具体的な問題に既存の定理証明器による形式証明を与え, その証明方法や定理証明器の特徴についての意見交換会も行いました.

近年では定理証明器はプログラム検証やシステム検証だけでなく, 証明の正しさの確認が難しい数学定理の証明の検証のためにも利用されて来ています. 数学理論の形式証明は数学理論そのものの発展のためだけではなく, 数学理論を用いた応用プログラムの実用性のための検証にも不可欠なものです. 今後, 数学理論とプログラム言語理論が定理証明器を介して同時並行的に進化し, 本研究集会参加者を中心に多くの研究分野が広がって行くことを祈念します. 本年度は九州大学で開催されましたが, 本研究集会は TPP (Theorem Proving and Provers Meeting) として, 多くの有志らにより毎年継続して開催されて来た研究集会です. 今年が丁度 20 回目になりますので, 記念に過去の開催地を列挙します:

> 北陸先端大学 (2005,2006), 筑波大学 (2007), 東北大学 (2008), 関西学院大学 (2009), 名古屋大学 (2010), 産業技術総合研究所 (2011), 千葉大学 (2012), 信州大学 (2013), 九州大学 (2014), 神奈川大学 (2015), お茶の水女子大学 (2016), 京都大学 (2017), 東北大学 (2018), 国立情報学研究所 (2019), 山口大学 (2020), 北見工業大学 (2021), 永平寺 (2022), 東京工業大学 (2023),

各開催回のホームページは今回ホームページ[1] からリンクされています. また, TPPmark2014 の問題と解答についても, 今後の定理証明器の発展を祈念しつつ, レポジトリ[2] に保存し記録として公開し続けます.

開催にあたり, 溝口佳寛先生, Reynald Affeldt さんに多大なるご尽力を賜りました。ここに深く感謝申し上げます。

本研究は九州大学マス・フォア・インダストリ研究所 共同利用・共同研究拠点の支援を受けた.（2024 年度プロジェクト研究-短期共同研究「コンピュータによる定理証明支援とその応用」（2024a024））

2024 年 12 月

Jacques Garrigue (名古屋大学)
中正和久 (山口大学)
南出和彦 (東京科学大学)

[1] https://www.math.nagoya-u.ac.jp/~garrigue/tpp2024/
[2] https://github.com/garrigue/tppmark2024

Preface

This workshop, held at the Kyushu University Institute of Mathematics for Industry on November 25–26, 2024, was intended as a forum for discussions about software verification, formalization of mathematics and formal verification of mathematical proofs. Participants included both researchers studying mathematics, logic, programming languages and software engineering, and engineers developing software. The total number of participants was around 80, and they listened to 25 talks and lectures.

In particular, Pierre-Marie Pédrot (INRIA, France), gave a special lecture about the underpinnings of interactive theorem provers, both from a theoretical and practical perspective. We shared our experiences about the present uses of formal proofs and theorem provers in each field. We also discussed ideas for future research to develop reliable software.

As customary for the TPP workshop series, TPPmark2024 allowed us to compare proofs of the same theorem using different approaches.

Recently, proof assistant systems are used not only for program verification, but also to verify formal mathematical proofs, which are hard to verify by humans alone. Formalization of mathematics is becoming more and more important for the verification of programs relying on mathematics. We hope that this will lead to progress in research on mathematics, computer science and software engineering, using formal proofs and proof assistant systems.

This year, TPP (Theorem Proving and Provers Meeting) was held in Kyushu University. This was the twentieth edition, and we wish to thank the organizers and participants of previous TPP workshops, held in the following locations:

> JAIST (2005,2006), Tsukuba Univ.(2007), Tohoku Univ.(2008), Kwansei Gakuin Univ.(2009), Nagoya Univ.(2010), AIST(2011), Chiba Univ.(2012), Shinshu Univ.(2013), Kyushu Univ.(2014), Kanagawa Univ.(2015), Ochanomizu Women Univ.(2016), Kyoto Univ.(2017), Tohoku Univ.(2018), NII (2019), Yamaguchi Univ.(2020), Kitami Institute of Technology (2021), Eiheiji (2022), Tokyo Institute of Technology (2023),

You can find links to those from the TPP2024 homepage [3]. The problem and answers of the TPPmark2014 are available from a repository [4].

This workshop could not have taken place without the continuous support of Professor Yoshihiro Mizoguchi at Kyushu University, and Reynald Affeldt of AIST.

This work was supported by the Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2024 Grant for Project Research–Short-term Joint Research "Computer Assisted Theorem Proving and Applications" (2024a024)).

December, 2024.

Jacques Garrigue	(Nagoya University)
Kazuhisa Nakasho	(Yamaguchi University)
Yasuhiko Minamide	(Institute of Science Tokyo)

[3] `https://www.math.nagoya-u.ac.jp/~garrigue/tpp2024/`
[4] `https://github.com/garrigue/tppmark2024`

contents

TPP2024 Program

25min（20min + 5min questions）per talk, except where otherwise indicated.

Monday, November 25

12:50-13:00: Opening

13:00-14:00

A Kernel of Truth（Invited talk）
Pierre-Marie Pédrot（INRIA Atlantique）

In this talk we will dive into the innards of a proof assistant based on type theory. We will focus more precisely on the technical constraints that this foundational setting generates, notably about the importance of computation at large. Contrarily to the dry and out-of-fashion feeling that this kind of topic may convey, it actually is a lively research area with many consequences, both for design and usage of such software.

14:15-16:10

Formalizing Premium Calculation of Life Insurance（30mn）
Yosuke Ito（Sompo Himawari Life Insurance Inc.）

Calculating premiums of life insurance is not an easy task. It requires mortality rates, interest rates, and future projections. The area of numerical analysis of insurance is called actuarial science. I report the ongoing work of formalization of life insurance mathematics, which is the life insurance part of the actuarial science.

Coq/SSReflect でコンストラクタ数が引数に依存して変化する Inductive 型と seqT 型の cons の型を T -> seq（seqT）-> seqT に変更した型を定義したい
Kenta Inoue

Coq/SSReflect に関する 2 つのショートトークです . 1 つ目は「コンストラクタ数が引数に依存して変化する（ある形式の）Inductive で定義された型」を定義したいときに Coq ではどのように表現すればいいのか考察しました . 特に引数として型と自然数の対のリストを引数にとり , このリストの各要素（T_i,n_i）に対し , T_i 型の引数と n_i 個の再帰呼び出しを行う引数からなるコンストラクタを構成するような Inductive 型について考察します . 2 つ目はリストの型である seq 型のコンストラクタ cons の型は型引数 T に対し , T -> seq T -> seq T で定義されますが , これを T -> seq（seq T）-> seq T に 変更して定義した型について考察します .

Verification of the Garsia-Wachs algorithm
Makoto Kanazawa（法政大学理工学部）

The Garsia-Wachs algorithm is an algorithm for finding a binary leaf tree with a given leaf sequence whose cost is as small as possible, where the cost is the sum of the costs associated with the leaf labels weighted by the depths of the leaves. The algorithm, along with a proof of correctness due to Kingston, is given in Knuth's The Art of Computer Programming, Vol. 3. I outline a formalization of this correctness proof in Dafny and make a few observations about the algorithm.

ブロックチェーンにおける支払い認証ポリシーの静的検証ツールの形式 検証
Yoshihiro Imai（株式会社 proof ninja）

本発表では、株式会社 DMM Crypto において管理している支払い認証ポリ シー TAP の設定を検証するために開発したツールについて述べる。TAP は Fireblocks によって提供される、ブロックチェーン上の資産管理および支払 い認証における信頼性の高いソリューションを提供しているが、認証ポリシーが正確に設定され、安全に運用されていることを保証するための形式的検証 が求められる。本研究では、Coq を用いて支払い認証ポリシーの安全性を形 式的に証明するツールを開発し、ポリシーが意図通りに動作し、潜在的な脆 弱性を排除することを実証した。これによりブロックチェーンにおける支払 い認証ポリシーの信頼性と安全性を向上させるとともに、業務効率の向上に も寄与することが期待される。

16:25-17:55（short talks, 15min each）

Automated Theorem Proving by HyperTree Proof Search with Retrieval-Augmented Tactic Generator
Sho Sonoda（Riken）

We developed an automated theorem proving system using a large language model（LLM）. The LLM generates proofs（precisely, sequences of tactics）by interacting with the Lean theorem prover. Generation of the tactic sequences is based on a Monte Carlo tree search called HyperTree Proof Search（HTPS）combined with a retrieval-augmented generator（RAG）called ReProver.

Isabelle/HOL を用いた差分プライベートなアルゴリズムの安全な実装
松岡和貴（東京科学大学　理工学系情報理工学院数理・計算科学系数理・計算科学コース）

離散ラプラス分布を用いた差分プライベートなアルゴリズムを対象に、定理証明支援系 Isabelle/HOL を用いてその形式化と検証を行う。さらに、Isabelle/HOL のコード生成機能を用いて差分プライバシーが検証された安全 な実装を得る。

標準的な様相論理の Lean での形式化について
野口真柊（神戸大学システム情報学科）

命題論理に様相演算子□と◇を導入した標準的な様相論理の Lean での形 式化について，現在の進捗と今後の展望について軽く紹介する.

Post の対応問題のインスタンスの証明生成
大森章裕（東京科学大学 情報理工学院 数理計算科学系 南出研究室）

Post の対応問題のインスタンスの集合である PCP [3,4] を全て解決すると いう取り組みの過程で、その結果の信頼性を高めるために Isabelle/HOL の証 明を生成した試みを紹介します。インスタンスの数が 3170 個と大量にあり、それぞれへの結果を証明するためには適切な証拠を発見し、それを元に証明 を自動生成する必要がありました。自動生成の方針と、必要だったオートマ トン理論の形式化について話します。

重複を除き辞書順で最小の部分リストを返す効率的なアルゴリズムの Agda による検証
城戸道仁（法政大学大学院理工学研究科システム理工学専攻）

Richard Bird（2014）は、著作、関数プログラミングによる珠玉のアルゴ リズムデザイン（原題 :Pearls of Functional Algorithm Design）にて、Haskell のライブラリ関数 nub の型を変更することで、計算量が Θ（n log n）であり、リストから重複を取り除いた上で、辞書順で最小のものを返す関 数のプログラムが構成できることを示し、実際に構成したプログラムを著書 に著した。この関数の正当性を検証を定理証明支援系 Agda を用いて行う。

Complete graphs and independence numbers

Takafumi Saikawa（Nagoya University）

I will report on an ongoing formalization of graph-theoretic invariants. This is a continuation of the work presented at TPP2023, this time expanded by complete graphs and their characterization by independence numbers of graphs. This is a joint work with Kazunori Matsuda and Yosuke Tsuji.

Tuesday, November 26

9:00-10:30

λ- 計算の代数と幾何（40min talk）

Masahiko Sato（京都大学 情報学研究科）

It is well-known that the set Λ of open terms of type-free λ-calculus dose not behave naturally when we regard Λ as an algebra equipped with the operation of function application. For example, the standard translation of λ-terms into combinatory terms does not admit the ξ-rule of the λ-calculus（[1; page 153]）. To be more precise, the situation is that（1）for the Combinatory Logic, there are combinatory algebras which are sound and complete with respect to CL, but（2）these algebras are not sound for the λ-calculus because of theξ-rule. Selinger［2; page 558］introduced the notion of "absolute interpretation" and gave a lambda algebra which is closed under the ξ-rule. However, the ξ-rule still remains in the original λ-calculus. In this talk, we will introduce a modified version of the λ-calculus without the ξ-rule but can prove exactly the same set of equations（under the β-equality）as those provable in the original λ-calculus. The key idea is to re-define Λ as an algebra on which the free monoid generated by the set of variables acts. The idea is obtained by analyzing the difference between Frege style canonical judgments and Martin-Löf style hypothetical judgments. We proved our main results in Coq.

［1］Barendregt, H. P.（1984）The Lambda Calculus, its Syntax and Semantics. 2nd edn. North-Holland.

［2］PETER SELINGER, The lambda calculus is algebraic, JFP 12（6）: 549–566, November 2002.

（Joint work with Keisuke Nakano（Tohoku University））

ZF+ ¬ AC の相対的無矛盾性証明の Isabelle/ZF による形式化

Daiki Funane（東北大学大学院情報科学研究科）

We formalize the relative consistency proof of ZF+ ¬ AC using Isabelle/ZF proof assistant. Our approach assumes the existence of a transitive countable model of ZF and uses forcing to construct a symmetric extension which is a model of ZF+ ¬ AC. We show that the symmetric extension satisfies ZF+ ¬ AC by formalizing a relativized forcing relation based on the formalization of forcing by Gunther et al.

Lean を用いた Gödel の第一・第二不完全性定理の形式化

齋藤彰悟（東北大学大学院理学研究科数学専攻）

Gödel の第一・第二不完全性定理の Lean4 を用いた形式化について報告する．不完全性定理の形式化は 80 年代からすでに何度か行われているが，ここでは過去のものより強い結果である Cobham の R_0 上の第一不完全性定理と IΣ_1 上の第二不完全性定理の証明の形式化を行った．

10:45-12:25

暗号プロトコルの形式記述に向けた LLM チャットボットの活用
櫻田英樹（NTT コミュニケーション科学基礎研究所）

LLM チャットボットを活用して暗号プロトコルの形式記述を効率的に作成することを試みた．具体的には LLM チャットボットが自然言語で記述されたプロトコル仕様を理解し，これを形式的な記述に変換する過程を説明する．この手法を用いることで，形式検証ツールへの入力作成の最初のステップを支援し，形式検証ツールを使い始める際の労力を削減することを目指す.

A Formalization of Prokhorov's Theorem in Isabelle/HOL
平田路和（東京科学大学）

This talk will be presented in Japanese with English slides.
本講演は，ITP2024 で発表した "A Formalization of the Lévy-Prokhorov Metric in Isabelle/HOL" において，口頭発表では詳細に触れなかったプロホロフの定理を題材とする．プロホロフの定理によれば，ポーランド空間上 の一様に有界な有限測度の集合について, 相対コンパクト性と緊密性は同値 である．プロホロフの定理は，中心極限定理や Sanov の定理，輸送理論における最適カップリングの存在性を示すために用いられる確率論における主要定理の一つである．本発表では，プロホロフの定理とその証明で必要となるリースの表現定理，アラオグルの定理の特殊な場合の Isabelle/HOL における形式化について論じる.

Delay モナドを用いた一般再帰関数に対する等式変形による検証
川上竜司（名古屋大学 多元数理学研究科）

副作用を含む計算は、モナドと呼ばれる構造を用いることで関数型プログラムの中でうまく表現できることが広く知られている。Coq のライブラリ Monae は、そういったモナド構造に着目した、等式変形による副作用を含む計算に対する検証ツールである。一方で、Coq では無矛盾性を保証するため停止性を確認できる関数しか定義することができない。従って本研究では、Monae に Delay モナドを追加することで、非停止な関数を含む一般再帰関数を表現し検証することに取り組んでいる。本講演では、1.Delay モナドをどのように実装したか　2. それらを用いた検証例　3. 現在取り組んでいる Delay モナドと他のモナドとの組み合わせにつ いて説明する。

On Representability of Multiple-Valued Functions by Linear Lambda Terms Typed with Second-order Polymorphic Type System
Satoshi Matsuoka（産業技術総合研究所工学計測標準部門データサイエンス研究グループ）

We show that any many-valued function can be represented by a linear lambda term that is typed in second-order polymorphic type system.

13:30-15:10

ペトリネットにおける到達可能性問題の変種間の還元可能性の形式化
手塚凜（千葉大学大学院 融合理工学府）・山本光晴（千葉大学大学院 理学研究院）

ペトリネットにおけるいくつかの到達可能性問題の変種間の還元可能性 が 1976 年に Hack によって示されている。本研究では、この証明を MathComp 上で形式化した。この中には、還元先の決定問題に用いる新しいペトリネットを構成する必要がある非自明なものもいくつか含まれる。これらをアドホックに形式化するのではなく、系統的に証明するための共通手順を作成し、それに従って形式化を行った。

Axiomatic real numbers for verified exact real-number computation
Sewon Park（Kyoto University）

In this talk, I present cAERN, our axiomatic formalization of real numbers of computable analysis in dependent type theory, and the Coq proof assistant. The axioms argued to be sound for realizability interpretation enable us to extract certified exact real-number computation programs from proofs. I will further introduce our recent progress in cAERN in hyperspace computations and ordinary differential equation solving. This talk is based on my joint work with Holger Thies and Michal Konečný.

Combining cost and behavior in type theory
Yue Niu（National Institute of Informatics）

Lean を用いたスイッチング回路の安全性検証
瀬川秀一（北陸先端科学技術大学院大学）

スイッチング回路の安全性検証における客観性と網羅性を確保するため、定理証明支援系 Lean および数学ライブラリ Mathlib を用いた検証を試行している。検証にあたり、すでに構築された特殊関数や ODE に関する定理を活用し、製品ごとに求められる特性を証明する。証明を効率的に実現するには、状態空間の理論やハイブリッドシステムの検証理論の定式化が必要である。本発表では、検証手法の概要および証明の進行状況について紹介する。

15:25-16:40

TPPmark 2024（20min）
Jacques Garrigue（Nagoya University）and provers

導出原理の逆適用による定理の自動生成手法の提案（25min）
西島海斗（山口大学）

本研究では，一階述語論理における証明技法の一つである導出原理を逆 方向に適用することで，新たな定理を自動生成する手法を提案する．現在，定理証明コーパスが量・質ともに不足しており，データ不足が機械学習型の 自動定理証明器の性能向上を妨げていた．本手法では，結論から仮定を逆推 論することで多様な定理を生成し，データセットを拡充することを目指す．提案手法の具体的なアルゴリズムの設計と実装について詳述し，今後の実験 計画についても述べる．

Coq 証明支援系による DNA 計算の形式化（15min）
早川銀河（九州大学 数理学府 数理学専攻）

DNA 分子のもつ性質を利用して計算するモデルの一つとしてスティッカーシステムというものがあり、一部のスティッカーシステムは有限オートマトンと同等の計算能力が有ることが知られている。今回、Coq,ssreflect を用いてオートマトンを模倣するスティッカーシステム を構成し、その正しさ を形式的に示した。

Coq による実解析のための実用的な補題の開発（15min）
石黒吉洋（名古屋大学）

Coq で確率プログラムの意味論を検証するために、我々は MathComp-Analysis を拡張してきた。特に、基本的な確率分布を形式化するために、微積分の基本定理や積分の変数変換などの限定的なバージョンなど、微分と積分に関する様々な定理を作成した。本講演では、Coq での確率論における重要な補題であるガウス積分の定式化に焦点を当て、連続関数の積分の評価をより扱いやすくする非有界区間への補題の一般化についても説明する。

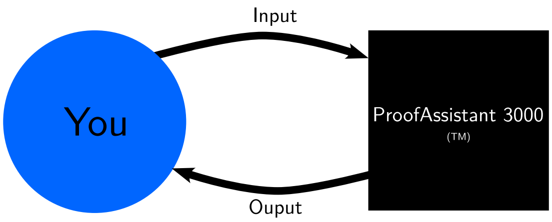
A Kernel of Truth

Pierre-Marie Pédrot

(Gallinette, INRIA)

TPP'24

On proof assistants and the extasis of their use†



Laborious interactive input:

- Theorem statements and object definitions
- Proof arguments e.g. tactics
- Copious amounts of insults

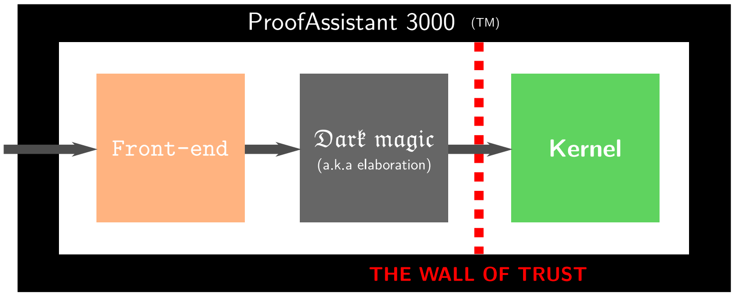
Cold mechanical output:

- *Good!* Here's what remains to do.
- `Nope.`
- (undecypherable dump of some low-level data)

† Your mileage may vary.

Highway to LCF

A well-known design: The LCF Model



- Clearly delineated Trusted Code Base
- All fancy stuff is outside the TCB
- Soundness is reduced to a *small* hopefully understandable kernel

Dicotyledons

Two standard kind of kernels in the wild

First kind: textbook LCF, still living in the HOL family

- Kernel is a trusted minimal API of HOL tactics
- Proofs not recorded, soundness ensured by the metalanguage

Second kind: when proofs matter as e.g. in dependent type theories

- Kernel is a type-checker
- Proofs are well-typed terms

Official Position

In this talk, we care about dependent type theories

It is Time to CIC Ass

CIC, the calculus of inductive constructions.

- A powerful dependent type theory
- Programming language or logical foundation?
- The idealized basis of two famous proof assistants

Full Disclaimer

I am a core Coq developer.

One type theorist is a system†

Full Disclaimer

I am an **opinionated** core Coq developer.

Thankfully, most of what I am saying should apply to other proof assistants based on dependent type theories.

- Kernel-wise, Coq and Lean are very close.
- I can still rant forever about some subtle differences in design
- Hint: Coq does it right (most of the time)

Some of what I will say even applies to Agda

... even if Agda has no separate kernel.

† — ***Two is a school. Three is a fork.***

A Monism in Two Minutes

CIC: Programming language or logical foundation?

Both!

Level 0: $\vdash (\lambda(x : A).\, x) : A \to A$

- the identity function on A?
- the canonical proof that A implies A?

Level 1: $\vdash M : \Pi(m : \mathbb{N}).\, \Sigma(p : \mathbb{N}).(m = 2p) + (m = 2p + 1)$

- an implementation of division by 2?
- a proof that division by 2 exists?

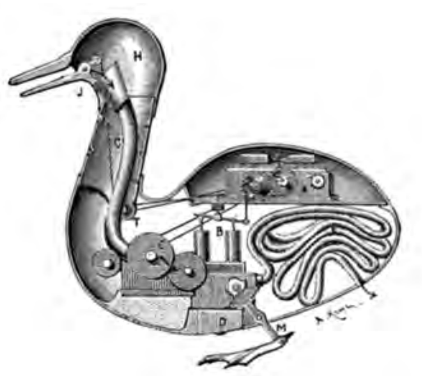
Level 2: $\vdash M : \Pi(b : \mathbb{B}).\, \texttt{if}\ b\ \texttt{then}\ \mathbb{N}\ \texttt{else}\ (\mathbb{N} \to \mathbb{N})$

Level 42: *(spec of CompCert)*

The pinacle of the Curry-Howard correspondence

If it Quacks like a Duck

The Coq kernel is *just* a CIC type-checker



Famous Last Words

"Surely it should be enough to understand CIC to understand the kernel."

Ceci n'est pas un Π

"Surely it should be enough to understand CIC to understand the kernel."

I Lied

There is no such thing as CIC.

- A bunch of rules spread across dozen of articles spanning decades
- No single source of truth
- A lot of implicit or contradictory stuff
- Folklore / unwritten knowledge

Not better implementation-wise.

- Takes some suspect liberties w.r.t. the spec
- It keeps changing

Our best bet: the MetaCoq project. But that's not today's topic.

Ora Pro Nobis

"The Non-Existent Type Theory is based upon both logic and faith. We have faith that it is implemented; we logically know that it is does not exist because it is not well-defined."

We will pretend that CIC exists in the remainder of the talk.

Persevere Diabolicum

"Surely it should be enough to understand CIC to understand the kernel."

... for some good notion of CIC.

Still Wrong

Reality has many asperities.

A specification is not an implementation.

Dialectics for Dummies

The Good Properties of CIC

Consistency: The Logician

There is no proof $\vdash M : \bot$.

... but the logic must be as expressive as possible.

Canonicity: The Programmer

If $\vdash M : \mathbb{N}$ then M evaluates to a numeral.

... but I want pointer equality, exceptions and Turing-completeness.

Implementability: The Maintainer

Type-checking is decidable.

... but I am going to add orthogonal features X, Y and Z.

Power Balance and Multiple Personalities

A Type Theory implementation is a delicate equilibrium.

$\rightsquigarrow$ Expanding the logic must preserve computation.

- One cannot just add axioms here and there
- Some constructions are not even axiomatizable (e.g. cubicalTT)

$\rightsquigarrow$ Expanding the programming language must preserve consistency.

- Extremely strong constraints, e.g. functions are total
- A lot of stuff from PLT just doesn't apply

$\rightsquigarrow$ Expanding any of these must keep the implementation tractable.

- Understandable spec / small implementation
- Efficient algorithmics
- Backwards compatibility

The logician, programmer and maintainer are often the same individual.

Pardon My French

— *Un noyau, c'est comme une andouillette:*
ça doit sentir un peu la merde, mais pas trop.

「カーネルはアンドゥイレットのようなものです。少し
糞のような臭いがするはずですが、あまり強くはありません」

The Core of the Kernel

The setting is now pinned down

In this talk we will discuss three interesting components of the Coq kernel.

Conversion | **Universes** | **Guard**

All while keeping the andouillette principle in mind!

Conversion

Convert and Repent

THE MOST IMPORTANT RULE OF CIC

(If you were snoozing away, now is the time to wake up.)

Meet CONVERSION:

$$\frac{\Gamma \vdash M : B \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash M : A}$$

$$\texttt{refl}_A : \Pi(x : A).\, x = x \qquad \rightsquigarrow \qquad (\texttt{refl}_{\mathbb{N}}\ 2) : 1 + 1 = 2$$

CONVERSION internalizes computation in the logic

- Not common in usual PL
- Irremediably ties the runtime to the type system
- A landmark of dependent types

Computation, computation everywhere

$$\Gamma \vdash A \equiv B$$

What is conversion exactly?

Generated by hardwired basic equations on the language e.g.

- β-reduction: $(\lambda(x : A).\, M)\ N \equiv M\{x := N\}$
- pattern-matching reduction on constructors
- constant unfolding

Remember, type-checking should be decidable, so conversion as well.

$\rightsquigarrow$ in particular the kernel must implement conversion.

Liberty, equality, provability

Why is conversion critical?

After all:

- Simple type theories like HOL do not have conversion*
- There are even "weak" dependent type theories without conversion

Conversion is both a blessing and a curse

$\rightsquigarrow$ Why not take advantage of something that is automagic?

- Delegating from the user to the machine is the point of an assistant
- In HOL you must provide a proof (e.g. by rewriting tactics)
- Inefficient: you have to store it somehow

$\rightsquigarrow$ Writing explicitly conversion derivations in CIC is not humanly possible.

Calculo Ergo Sum

Much better: we can take advantage of conversion.

Remember that CIC is a programming language? Let's put this to use.

Reflection

Replace logic by computation.

Idea: Assume some theory $\mathcal{T}$ that is decidable (or admits checkable proofs)

- define a CIC AST `formula` representing $\mathcal{T}$-formulas
- write a CIC embedding $\texttt{eval} : \texttt{formula} \to \texttt{Prop}$
- write a CIC function $\texttt{check} : \texttt{formula} \to \mathbb{B}$
- prove in CIC that $\Pi(\varphi : \texttt{formula}).\, \texttt{check}\ \varphi = \texttt{true} \to \texttt{eval}\ \varphi$

To prove $\Phi \in \mathcal{T}$ s.t. $\Phi := \texttt{eval}\ \varphi$, it is thus enough to **compute** $\texttt{check}\ \varphi$.

Sum Ergo Calculo

Reflection is a very powerful technique.

- Historically used for performance reasons
- Abstractly, a way to teach the kernel any theory (a logical JIT)

Many use cases: equational reasoning, arithmetic, SAT solving...

A related (but distinct) technique: small scale reflection.

- Similar idea of computing away trivial reasoning steps
- At the core of the SSReflect framework
- Famously used by Mathcomp and friends (e.g. Feit-Thompson proof)

Morale

Computation matters!

Trinitarian Apostasy

The Coq kernel has not one but **three**† conversion algorithms.

⇝ The "reference" one

- Untyped, call-by-need reduction machine in OCaml
- Tailored for symbolic conversion, extremely sensitive

⇝ The "heavy-duty" ones: VM and native

- Compilation to ZINC-like bytecode / machine code
- The VM is implemented in C, native reuses the OCaml runtime
- Tailored for brutal computation (typically, compute a boolean)

Different kind of trade-offs. What is the design space?

(For instance, Lean has an ad-hoc native-like process that only works on closed terms.)

I Have no Proof and I Must Qed

We are not done with conversion yet!

Let $p, q : \{n : \mathbb{N} \mid \texttt{isEven}\ n\}$.

If $p.1 \equiv q.1$ then we do not have in general $p \equiv q$.

In pen-and-paper proofs one *never ever* cares about that.

The Dependent Hell

Proofs are programs, and thus relevant.

We would like **more** conversion!

Not just a theoretical issue, this is a real PITA in practice.

Sibylline errors in innocuous scripts that require a PhD in type theory to understand.

```
Abstracting over the term "n" leads to a term
fun n0 : nat ⇒ exist (fun n1 : nat ⇒ isEven n1) n0 p = exist (fun n1 : nat ⇒ isEven n1) m q
which is ill-typed.
Reason is: Illegal application:
The term "exist" of type "forall (A : Type) (P : A → Prop) (x : A), P x → {x : A | P x}"
cannot be applied to the terms
"nat" : "Set" "fun n : nat ⇒ isEven n" : "nat → Prop" "n0" : "nat" "p" : "isEven n"
The 4th term has type "isEven n" which should be a subtype of "(fun n : nat ⇒ isEven n) n0".
(cannot unify "isEven n" and "isEven n0")
```

(This is just after `rewrite` e where $m, n : \mathbb{N}$ and $e : m = n$.)

STOP DOING TYPE THEORY

- Variables were not supposed to be given Greek names
- Years of OCaml yet no real-world use for functionals of order > 2
- Please give me Π(n : ℕ) (p q : eq ℕ n n). eq (eq ℕ n n) p q of something.

(This is REAL CIC, done by REAL type theorists):

$$[\![\Gamma]\!]_p \vdash [M]^{\Gamma}_p : [A]^{\Gamma}_p.\mathsf{T}\ p\ \mathsf{id}_p$$
$$[\![\Gamma]\!]_p \vdash \{M\}^{\Gamma}_p : [A]^{\Gamma}_p.\mathsf{R}\ (\lambda(q\,\alpha : p).\,\mathsf{rw}\ (\{A\}^{\Gamma}_p\ q\ \alpha)\ (\alpha \bullet_{\Gamma} [M]^{\Gamma}_q))$$

???

```
Trm Γ
(@typ_sub
  (ext (ext Γ A)
    (@Idn (ext Γ A) (@typ_sub Γ (ext Γ A) A (@wkn Γ Γ A (idn Γ)))
      (@trm_sub Γ (ext Γ A) A t (@wkn Γ Γ A (idn Γ))) (@rel0 Γ A)))
  Γ P
  (@cns (ext Γ A) Γ
    (@Idn (ext Γ A) (@typ_sub Γ (ext Γ A) A (@wkn Γ Γ A (idn Γ)))
      (@trm_sub Γ (ext Γ A) A t (@wkn Γ Γ A (idn Γ))) (@rel0 Γ A))
    (@cns Γ Γ A (idn Γ) u) e)).
```

????????

$$\frac{M \Downarrow_k M_0 \qquad \mathbf{clos}\ M_0}{♀\, M \downarrow_{k+1} [\ulcorner M_0 \urcorner]_{\mathbb{N}}}$$

?????

"Hello I would like  apples please"

They have played us for absolute fools

A well-known problem that has plagued CIC for years

- Famous hazing for PhD students
- SSReflect even has a design pattern to work around the issue
- Outside of the kernel, not completely satisfactory

Some are More Equal than Others

Recently solved by the introduction of a universe of strict propositions

- After all, proofs are not quite programs
- We don't care about proof contents: "all proofs are born equal."

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : A \qquad \Gamma \vdash A : \mathtt{SProp}}{\Gamma \vdash M \equiv N : A}$$

The rules for SProp are tricky

- The feature was inspired by foundational work in HoTT
- Required non-trivial changes in the kernel
- Lean notoriously ~~doesn't give a shit~~ is practically-minded

I Want to eq Free

SProp is a game changer

- An elegant solution to perennial issues
- Critical, but not enough

There are many more limitations to conversion!

$$\texttt{hd} : \Pi(n : \mathbb{N}).\,\texttt{vec}\ A\ (1+n) \to A \qquad v : \texttt{vec}\ A\ (n+1)$$

$\texttt{hd}\ n\ v$ does not type-check because $1 + n \not\equiv n + 1$

This is much harder to solve.

Back to the Future

Interestingly, these questions were fashionable in the '90s and 00's

- Extensionality in type theory (Hofmann)
- Observational type theory (Altenkirch-McBride)
- Coq Modulo Theory (Strub)

... then nobody cared

- Systems were either not implemented or not used / not maintained

... but now it is making a comeback

- strict propositions
- rewrite rules
- extension types

Will the cycle continue?

Universes

Encounter of the Third Type

In CIC, types are first-class citizens.

$\rightsquigarrow$ in particular, types have a universe type, traditionally called Type.

What is the type of Type?

Martin-Löf '71: Type : Type.

Girard '71 + ε: Type : Type is inconsistent.

Standard solution: one has to stratify.

$$\mathtt{Type}_0 : \mathtt{Type}_1 : \mathtt{Type}_2 : \ldots : \mathtt{Type}_n : \mathtt{Type}_{n+1} : \ldots$$

Good Old BASIC

We theoretical computer scientists love natural numbers!

$$(\mathtt{Type}_i)_{i \in \mathbb{N}}$$

- Simple
- Elegant
- Universal
- **... and completely anti-modular!**

You have to pick all your levels upfront.

If you have two universes $\mathtt{Type}_n : \mathtt{Type}_{n+1}$, you better not realize that you need some m s.t. $n < m < n + 1$.

This is why you should number your levels by increments of 100.

Good Old BASIC

We theoretical computer scientists love natural numbers!

$$(\mathtt{Type}_i)_{i \in \mathbb{N}}$$

- Simple
- Elegant
- Universal
- **... and completely anti-modular!**

You have to pick all your levels upfront.

If you have two universes $\mathtt{Type}_n : \mathtt{Type}_{n+1}$, you better not realize that you need some m s.t. $n < m < n + 1$.

~~This is why you should number your levels by increments of 100.~~

Putting Things in Order

Floating universes (antediluvian trick)

Just introduce variables

The language is simple:

- Variable universe levels $i, j\ldots$
- Constraints $i < j$ and $i \leq j$ which must form a DAG

Minor tweaks to the kernel

- Generate fresh levels (outside of the kernel)
- Accumulate constraints (outside of the kernel)
- Send the graph to the kernel for acyclicity checking

Voilá, you never have to care about universes again

(By the way Agda and Lean have a different approach.)

Cosmic Radiation

This is a very wasteful technique

$\rightsquigarrow$ In 99% of the actual developments:

You have to generate a bazillion universe levels

- Most of them are transient
- They are generated by tactics and unified away immediately
- You only have to send a gazillion levels to the kernel

The gazillion-sized graph is a fiction. Collapsing $\leq$ constraints gives:

$$i \quad < \quad j \quad < \quad k$$

Three levels ought to be enough for anybody!

In Space No One Can Hear You Scream

This is an actually very annoying problem

Users have been complaining about related performance issues for ages.

Even when you are not aware, you pay for this.

Universe generation is also very much inscrutable.

It is effectful and incompatible with desirable features.

Good luck debugging a stray constraint.

Horror Story

The MetaCoq and QuickChick are (were?) not loadable together.

The Untold Eldritch Horrors

Worse!

Serious Question: What is the type of $\mathtt{Type}_i$?
Timid Answer: $\mathtt{Type}_j$ for some $j > i$?

No! You shouldn't be able to generate fresh levels from within the kernel.

I lied (again)

We still need algebraic universe expressions in types.

- In Coq, types are *actually* not terms!
- Some kind of adjunction between types and terms

$$\exists j > i.\, \mathtt{Type}_j \qquad \sim \qquad \mathtt{Type}_{i+1}$$

The Andouillette Principle

I am not sure I have seen this really publicized anywhere.

A New Hope

This universe business is currently a hot topic

- Yet another universe checking algorithm
- That handles algebraic universes natively

Resonates with the multiverse project

- More complex sorts, expanding the logic
- All of this is tied together inextricably

Yet another revival of dormant questions

Guard

Logical Totalitarianism

In CIC, all functions must terminate.

- Otherwise the theory becomes inconsistent.
- Arbitrary fixpoints allow $n : \mathbb{N}$ s.t. $n := \mathsf{S}\ n$

In paper presentations, for simplicity one uses recursors.

$$\begin{array}{c}\mathtt{rec}_{\mathbb{N}} : P\ \mathsf{O} \to (\Pi(n:\mathbb{N}).\ P\ n \to P\ (\mathsf{S}\ n)) \to \Pi(n:\mathbb{N}).\ P\ n \\ \begin{array}{rcl}\mathtt{rec}_{\mathbb{N}}\ p_O\ p_S\ \mathsf{O} & \equiv & p_O \\ \mathtt{rec}_{\mathbb{N}}\ p_O\ p_S\ (\mathsf{S}\ n) & \equiv & p_S\ n\ (\mathtt{rec}_{\mathbb{N}}\ n\ p_O\ p_S)\end{array}\end{array}$$

The Category Terrorist

"$\mathtt{rec}_{\mathbb{N}}$ is universal, because this is the universal property of $\mathbb{N}$."

Commutatively Squaring the Circle

The Category Terrorist

"$\mathtt{rec}_{\mathbb{N}}$ is universal, because this is the universal property of $\mathbb{N}$."

This is only true extensionally!

- This only holds for propositional equality
- Further assuming various extensionality principles

Computation matters

Would you conflate a $O(2^n)$ algorithm with $O(1)$ one?

- Intensional behaviour is critical for programming
- Recursors are very bad in call-by-value
- It is not even clear what universality means for conversion
- Whatever this means, recursors are not universal for it

Fixpoints All The Way Down

Good news: recursors are not fundamental in Coq.

Instead, Coq relies on fixpoints + pattern-matching.

$$\texttt{rec}_{\mathbb{N}}\ p_O\ p_S \quad := \quad \texttt{fix}\ F\ (n : \mathbb{N}) := \texttt{match}\ n\ \texttt{with}\ \mid \mathsf{O} \Rightarrow p_O\ \mid \mathsf{S}\ m \Rightarrow p_S\ m\ (F\ m)$$

This is a historical design choice motivated by extraction

- Similar to OCaml
- The extracted terms look like what the user wrote
- Critical for efficiency in call-by-value

One can write fixpoints that are not **intensionally** recursor-encodable.

$$\texttt{even} : \mathbb{N} \rightarrow \mathbb{B}$$

$$\begin{array}{lcl} \texttt{even}\ \mathsf{O} & := & \mathsf{true} \\ \texttt{even}\ (\mathsf{S}\ \mathsf{O}) & := & \mathsf{false} \\ \texttt{even}\ (\mathsf{S}\ (\mathsf{S}\ n)) & := & \texttt{even}\ n \end{array}$$

Fixing the Fixpoints

Bad news: recursors are not fundamental in Coq.

Consider the following:

- Coq relies on fixpoints + pattern-matching.
- Arbitrary fixpoints are inconsistent.

Hence there must be some mechanism to restrict to *good* fixpoints

The Guard!

Total Programming is Totally Nuts

What does the guard enforce?

Minimal service:

- The theory must be consistent
- Hence functions ought to be total.

Once again, the MetaCoq people worked this out a bit.

- Various closure conditions
- Some surprisingly non-necessary properties

The more expressive the guard, the better.

(Right?)

Let's Put an End to This

The guard condition is probably the least understood kernel component.

- Specification not quite clear, stay tuned
- Organic implementation — *it would be nice if this worked...*
- Decades of tweaks and RFC from users
- **... and obviously critical for consistency**

I want to give you a foretaste of kern-hell.

The Guard Dies But Does Not Surrender

Ingenuous question

What does total mean?

In CIC, the equational theory is call-by-name.
$\rightsquigarrow$ In particular, we only care about weak-head reduction.

This used to be accepted

$$\texttt{fix loop}\ (i : \texttt{unit}) := \texttt{let}\ \_ := \texttt{loop}\ ()\ \texttt{in}\ ()$$

- Perfectly fine in call-by-name
- Not inconsistent, this is just a constant function
- Not quite so in call-by-value, e.g. through extraction

As of Coq 8.19, not accepted, but still morally OK in the abstract.

Omae wa Mô Shinde Iru

Logically, what is the worse you could get from a defective guard?

"Morally, you could be inconsistent. There should not be anything in between. Apart from more functions, that is." — Sweet Summer Child.

The guard condition used to negate propositional extensionality.

$$\texttt{PropExt} := \Pi(P\,Q : \texttt{Prop}).\,(P \leftrightarrow Q) \to P = Q$$

$\rightsquigarrow$ this is inconsistent with both HoTT and the **Set** model

In the name of Gödel, what does this have to do with termination?

(Mumble something about size issues.)

The Sad Truth

Consistency is not enough!

We want the guard to be as neutral as possible w.r.t. model validity...

- HoTT
- **Set**
- Something else?

... but we also want it to be as expressive as possible.

- Some people out there make a living of this

This is not a formal specification!

The Unescapable Wall of Reality

The only idealized model we understand is recursor-based†.

(†This is not even completely true. If you hear the word *nested*, run.)

⇝ We should be able to justify the guard by compilation to recursors.

- This could even be done outside of the kernel
- This is actually used by e.g. Equations

But this is doomed to fail: there are fixpoints not recursor-encodable.

⇝ At least not with an intensional notion of encodable.

- Some Coq-definable fixpoints conflict with recursor models
- Effects are pretty much incompatible with some guard assumptions
- What is a good notion of encodable?

Who shall guard the guard?

Conclusion

Bad Tripes?

The confidence of the theoretician crashes against the wall of reality

- The real has much more asperities
- Even in an idealized kernel lurk unknown monsters
- Diachronical and interindividual dialectics are pervasive

So is the essence of andouillette.

- Proof assistants are still an invaluable tool
- The andouillette principle should not be feared
- Rather, this is a never ending material for a thriving research

Do not be afraid and join us

Scribitur ad narrandum, non ad probandum.

L'absurde ne délivre pas, il lie. Il n'autorise pas tous les actes. Tout est permis ne signifie pas que rien n'est défendu.

Albert Camus.

Thank you for your attention.

Formalizing Premium Calculation of Life Insurance

Yosuke ITO

The 20th Theorem Proving and Provers meeting (TPP 2024)
Nov. 25th, 2024

Sompo Himawari Life Insurance Inc.

Disclaimer

1. The contents presented here are solely the speaker's opinions and do not reflect the views of Company.
2. There are some inaccuracies in explaining actuarial mathematics due to the priority on intuitive understanding.
3. This presentation is a progress report on the library "Actuarial Mathematics" in the Archive of Formal Proofs of Isabelle. The future version might be different from the one presented below.

Self Introduction (Professional Experience)

Sompo Himawari Life Insurance Inc., Dec. 2020 – Present.

- Aggregates the business results of life insurance products.

Meiji Yasuda Life Insurance Company, Apr. 2014 – Nov. 2020.

- Revised the reinsurance contracts.
- Determined the prices of life insurance products.
- Attended the approval negotiations with Financial Services Agency.
- Qualified as an actuary (Fellow of the Institute of Actuaries of Japan).
- Aggregated the business results of group life insurance.
- Calculated retirement benefit obligations of enterprises.
- Validated the financial soundness of Pension Plans.

3/33

Self Introduction (Education)

Nagoya University

- Master of Mathematical Sciences, Mar. 2014.

The University of Tokyo

- Bachelor of Science, Mathematics Course, Mar. 2012.

4/33

Contents

5/33

Contents

6/33

What Is Actuarial Mathematics?

Actuarial mathematics is a branch of applied mathematics, which is used to evaluate financial risks of undesirable events.

It is related to

- calculus,
- probability theory,
- statistics,
- financial theory.

The traditional actuarial roles are considered as

- determining the prices of insurance products,
- estimating the liability of a company associating with the insurance contracts.

Recently, the risk management skill of actuaries is required in a wider range of businesses.

7/33

How We Determine the Premium of a Life Insurance

A **premium** (**保険料**) is paid to a life insurance company by the **policyholder** (**保険契約者**).

A **benefit** (**保険金**) is paid by a life insurance company according to the life insurance policy.

The **equivalence principle** (**収支相等の原則**) requires that the benefit outgo be equal to the premium income;

$$\text{EPV (Expected Present Value) of the benefit outgo} = \text{EPV of the premium income.}$$

8/33

Progress Report

I have sorted out the theory of the equivalence principle on paper in a rigorous, general and explicit manner.

- **Rigor**
  In order to formalize the equivalence principle, firstly we have to write it down rigorously.
- **Generality**
  The formalized principle should be applicable to all types of life insurance.
- **Explicitness**
  All the typical calculation formulas should be derived from the formalized equivalence principle.

Contents

Present Value

If we can easily earn 2% returns annually, \$100 at present is worth \$102 next year.

If an asset produces incomes C_t at the time t (in years), their **present value** is defined to be

$$\sum_t \frac{C_t}{(1+i)^t}$$

where i is an annual interest rate.

11/33

Pricing a Term Life Insurance (1-year policy)

"Term life insurance provides a death benefit for a specified period of time that pays the policyholder's beneficiaries" [1].

Assumption

- amount insured: \$10000
- policy period: 1 year
- time of payment: the end of the year
- annual mortality rate: 1%
- annual interest rate: 2%

The expected payment after 1 year is $\$10000 \times 1\% = \$100$. If the insurance company earns a 2% investment yield annually, the present value of this insurance is $\$100/(1+2\%) \approx \$98$.

12/33

Notations

T_x : future lifetime random variable of a person aged x

${}_tp_x := P(T_x > t \mid T_x > 0)$ (survival rate)

$p_x := {}_1p_x$

${}_tq_x := P(T_x \leq t \mid T_x > 0)$ (mortality rate)

$q_x := {}_1q_x$

$\mu_x := \lim_{\Delta t \searrow 0} \frac{{}_{\Delta t}q_x}{\Delta t}$ (force of mortality)

i : annual interest rate

$v := \frac{1}{1+i}$ (discount factor)

13/33

Pricing a Term Life Insurance

Definition

The present value of a term life insurance on a person aged x payable at the end of the year of death within n years is written as $A^1_{x:\overline{n}|}$ per unit insurance amount:

$$A^1_{x:\overline{n}|} = \sum_{k=1}^{n} v^k \cdot {}_{k-1}p_x \cdot q_{x+k-1}.$$

The present value means nothing other than the **single premium (一時払保険料)**.

14/33

Pricing a Continuous Term Life Insurance

Definition

The present value of a term life insurance on a person aged x payable at the moment of death within n years is written as $\bar{A}^{1}_{x:\overline{n}|}$ per unit insurance amount:

$$\bar{A}^{1}_{x:\overline{n}|} = \int_{(0,n]} v^t \cdot {}_tp_x \cdot \mu_{x+t}\, dt.$$

Pricing a Term Life Annuity-Due

Definition

The present value of a term life annuity on a person aged x payable at the beginning of each year within n years is written as $\ddot{a}_{x:\overline{n}|}$ per unit annual payment:

$$\ddot{a}_{x:\overline{n}|} = \sum_{k=1}^{n} v^{k-1} \cdot {}_{k-1}p_x.$$

Calculating Level Premiums

Level premiums (平準払保険料) are the premiums paid regularly whose amount is constant throughout the term.

The policyholder has to keep paying the level premium as long as the **insured (被保険者)** is alive.

The EPV (Expected Present Value) of the annually-payable level premium P of an n-year life insurance is $P \cdot \ddot{a}_{x:\overline{n}|}$.

Equivalence Principle

Equivalence Principle

It is required that

$$\text{EPV of the benefit outgo} = \text{EPV of the premium income.}$$

Example

The level premium P of an n-year continuous term life insurance should satisfy $\bar{A}^{1}_{x:\overline{n}|} = P \cdot \ddot{a}_{x:\overline{n}|}$. Therefore,

$$P = \frac{\bar{A}^{1}_{x:\overline{n}|}}{\ddot{a}_{x:\overline{n}|}}.$$

Contents

Formulating EPV (Expected Present Value) i

The goal is to express the equivalence principle by a mathematical formula.

It suffices to uniformly define the EPV for any type of life insurance.

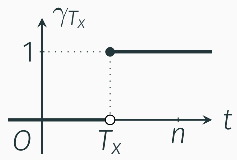
We introduce a function $\gamma(t)$ representing the **total** amount of benefits to be paid until the time t.

The function $\gamma(t)$ may depend on T_x. We henceforth write it as $\gamma_{T_x}(t)$.

Formulating EPV (Expected Present Value) ii

For a continuous term life insurance,

$$\gamma_{T_x}(t) := \begin{cases} 1, & \text{if } t \geq T_x \text{ and } T_x \leq n, \\ 0, & \text{otherwise.} \end{cases}$$

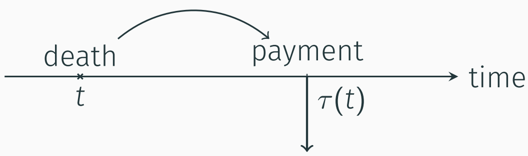


The EPV of this life insurance can be expressed as

$$E\left[\int_{\mathbb{R}} v^t \, d\gamma_{T_x}(t) \;\middle|\; T_x > 0\right].$$

Formulating EPV (Expected Present Value) iii

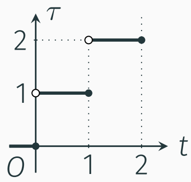
We also introduce a function $\tau(t)$ representing the actual time when the benefit is paid whose obligation is incurred at the time t.



Formulating EPV (Expected Present Value) iv

Consider a term life insurance payable at the end of each year. Let $\gamma_{T_x}(t)$ be the same as the previous one, and

$$\tau(t) := \lceil t \rceil \quad \text{(ceil function)}.$$



Then the EPV of this life insurance can be expressed as

$$E\left[\int_{\mathbb{R}} v^{\tau(t)}\, d\gamma_{T_x}(t) \,\middle|\, T_x > 0\right].$$

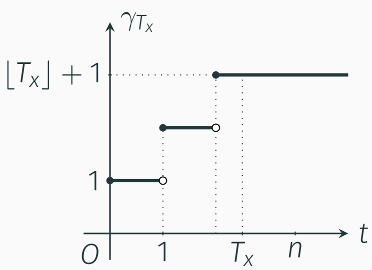
23/33

Formulating EPV (Expected Present Value) v

This framework is applicable to a term life annuity-due:

$$\gamma_{T_x}(t) := \begin{cases} \min\{n, \max\{0, \lfloor t \rfloor + 1\}\}, & \text{if } t < T_x, \\ \lim_{s \nearrow T_x} \gamma_{T_x}(s), & \text{if } t \geq T_x, \end{cases}$$

$$\tau(t) := t.$$



24/33

Formulating EPV (Expected Present Value) vi

The EPV of the life annuity can be expressed in the same formula as the life insurance:

$$E\left[\int_{\mathbb{R}} v^{\tau(t)}\, d\gamma_{T_x}(t) \,\middle|\, T_x > 0\right].$$

25/33

Axioms of the Framework of EPV

The specification of life insurance is determined by $\gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $\tau : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

These functions should satisfy the following axioms.

1. For any $\theta \in \mathbb{R}$, $\gamma(\theta, \cdot) : \mathbb{R} \to \mathbb{R}$ is right-continuous and nondecreasing.
2. For any $\theta \in \mathbb{R}$, $\tau(\theta, \cdot) : \mathbb{R} \to \mathbb{R}$ is Borel measurable.
3. The function $\mathbb{R} \to \mathbb{R}; \theta \mapsto \displaystyle\int_{\mathbb{R}} v^{\tau(\theta,t)}\, d\gamma_\theta(t)$ is Borel measurable.
4. For any θ and $t \in \mathbb{R}$, $t < 0$ implies $\gamma(\theta, t) = 0$.

These are sufficient to derive calculation formulas such as

$$\ddot{a}_{x:\overline{n}|} = \sum_{k=1}^{n} v^{k-1} \cdot {}_{k-1}p_x.$$

26/33

Contents

Formulating the Equivalence Principle

Let (γ^B, τ^B) denote the specification of the life insurance benefits. Let (γ^P, τ^P) denote the specification of the premium payments.

The (level) premium of the life insurance satisfying the equivalence principle is

$$\frac{E\left[\int_{\mathbb{R}} v^{\tau^B(T_x,t)}\, d\gamma^B_{T_x}(t) \,\middle|\, T_x > 0\right]}{E\left[\int_{\mathbb{R}} v^{\tau^P(T_x,t)}\, d\gamma^P_{T_x}(t) \,\middle|\, T_x > 0\right]}.$$

Implementing the Equivalence Principle in Isabelle

I plan to implement the equivalence principle by `locale` and `sublocale`.

Cf. Snapshot of Valuation.thy.

Future Work

1. Complete the implementation of the equivalence principle in Isabelle/HOL.
   - The current theory of actuarial mathematics is in the AFP: `https://www.isa-afp.org/entries/Actuarial_Mathematics.html`
2. Simultaneously proceed with Coq (Rocq) formalization.
   - The probability-free elementary formalization of actuarial mathematics can be found in `coq-actuary`: `https://github.com/Yosuke-Ito-345/Actuary`
   - This is based on Coquelicot, but I am upgrading it based on MathComp-Analysis.

Future Applications

1. Prevent errors in examination papers of actuarial mathematics.
2. Detect errors in actuarial documents, e.g. statement of calculation procedures.
3. Formally verify programs in actuarial software.

31/33

References

[1] Julia Kagan.
Term life insurance: What it is, different types, pros and cons.
`https://www.investopedia.com/terms/t/termlife.asp`, 2024.

32/33

Q&A

Any questions or comments? (Both in Japanese and English.)

You can also reach me at GitHub:
`https://github.com/Yosuke-Ito-345`

33/33

1 Inductive type with number of constructors depending on input in Coq/SSReflect

In Coq/SSReflect, we can't define the following inductive type whose number of constructors depend on the input `s:seq (Type * nat)`.

```
 Inductive myType (s:seq (Type * nat)) : Type :=
 | MyType_0th : T_0 → myType s → ... → myType s → myType s
 | MyType_1st : T_1 → myType s → ... → myType s → myType s
       :
 | MyType_kth : T_k → myType s → ... → myType s → myType s.
             (* T_k → iter n_k (myType s →) (myType s) *)

(* where s = [:: (T_0,n_0); (T_1,n_1); ...; (T_k,n_k)] *)
```

The number of constructors of `myType s` is the size of input `s = [:: (T_0,n_0); (T_1,n_1); ...; (T_k,n_k)]` and each i-th constructor's type is `T_i` → `myType s` → `...` → `myType s` → `myType s` where the number of `myType s` arguments is `n_i`. However this type cannot be defined in Coq directly, so we use another definition that is equivalent to `myType`. We prove the equivalence by showing the axioms of inductive types.

As a result, we can substantially define `myType` using an axiom weaker than excluded middle.

Blog in Japanese. `https://qiita.com/nekonibox/items/3cd3652e5a1d1dd734b8`

2 List type that has cons : $A \to \text{list}(\text{list } A) \to \text{list } A$ instead of $A \to \text{list } A \to \text{list } A$ in Coq/SSReflecct

In Haskell, we can define the following type, that is, a list type that has a constructor `Cons: T` → `seq (seq T)` → `seq T` instead of `Cons: T` → `seq T` → `seq T`.

```
data SSeq a = SNil | SCons a (SSeq (SSeq a))
```

However, we cannot define the type in coq/SSReflect directly.

```
 Inductive sseq : Type → Type :=
 | snil T: sseq T
 | scons T: T → sseq (sseq T) → sseq T.
 (* Error: Universe inconsistency *)
```

So, we define the type by adding another `nat` argument as:

```
 Inductive sseqn (A:Type) : nat → Type :=
 | sseqn0 : A → sseqn A 0
```

```
  | sseqnNil d: sseqn A d.+1
  | sseqnCons d: sseqn A d → sseqn A d.+2 → sseqn A d.+1.
```

we define this type so that ∀ `A n, sseqn A n = iter n sseq A`. Therefore `sseq A` can define as `sseqn A 1`. Then, we prove `sseq` satisfies the axioms of inductive type.

Blog in Japanese. `https://qiita.com/nekonibox/items/b3c0485090e65bd95ae0`

1

Verification of the Garsia–Wachs Algorithm

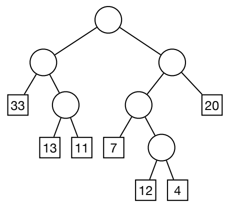
Makoto Kanazawa
Hosei University

2

Garsia–Wachs Algorithm

Input 33 13 11 7 12 4 20 weight sequence

Output

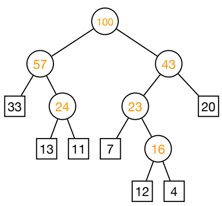


optimum “alphabetic” binary tree

“Alphabetic” means that the sequence of leaf labels is exactly the input sequence.

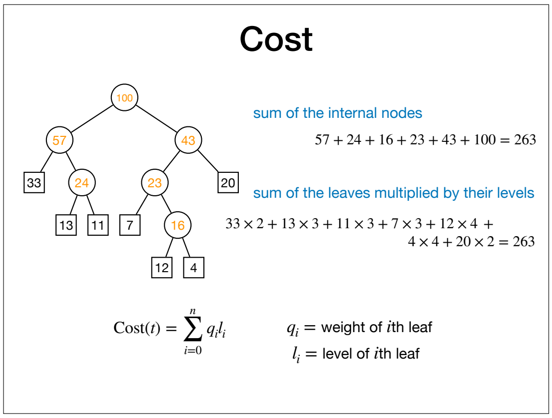
3

Cost



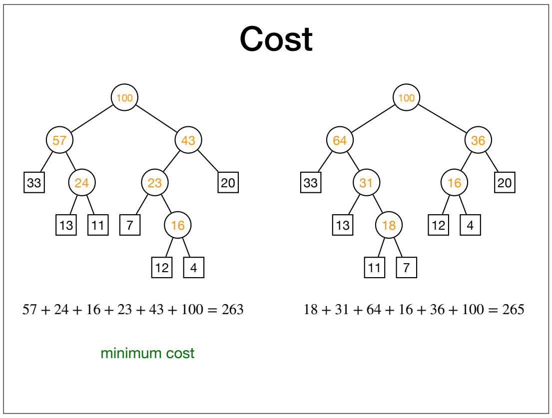
- The value attached to an internal node is the total weight of the subtree rooted at that node.
- The **cost** of the tree is the sum of all internal node values.

The algorithm finds an alphabetic tree with minimum cost.
The intuition behind the definition of cost is the following. Suppose you traverse this tree from the root to some leaf, and the weight associated with a leaf corresponds to the probability of reaching that leaf. Then the cost corresponds to the expected number of nodes that are visited before reaching a leaf.



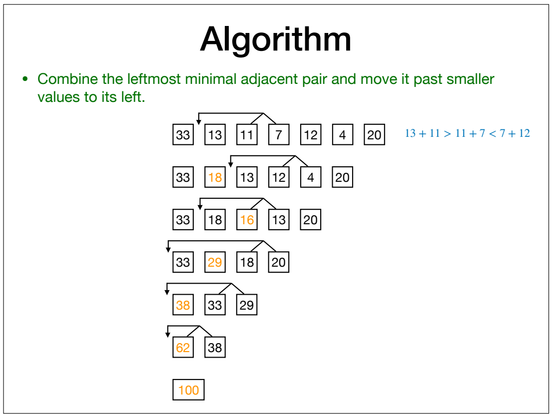
4

The cost can be computed in two equivalent ways. The *n* in the formula is the length of the input sequence minus one.



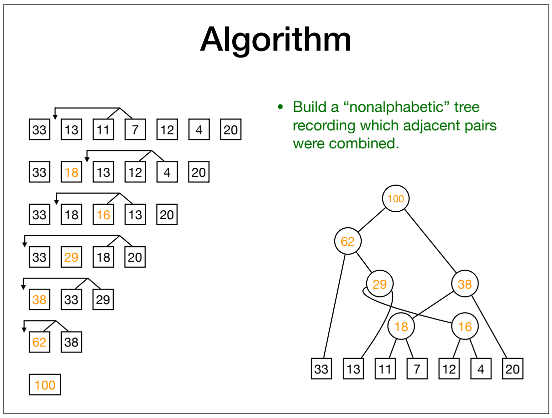
5

In this input sequence, (12, 4) is the pair of adjacent weights whose sum is the smallest. When (12, 4) is replaced by their sum, the minimum weight adjacent pair is (11, 7). For the purpose of obtaining an optimum alphabetic tree, the strategy of repeatedly combining minimum weight adjacent pairs does not work.



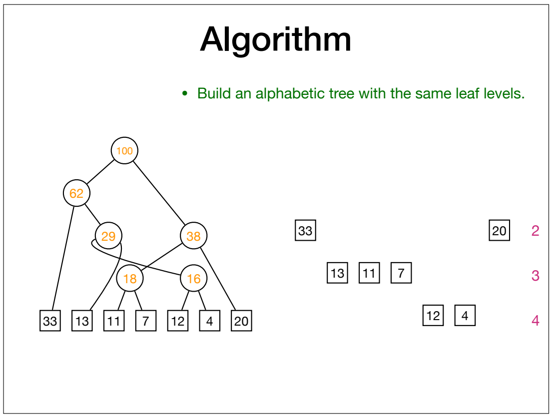
6

The Garsia-Wachs algorithm works as follows. You pick the leftmost adjacent pair of weights whose sum is “locally minimal”. You combine it into one weight and move it past the smaller weights to its left.



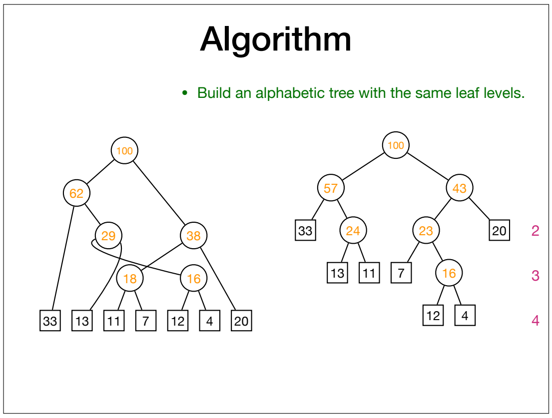
7

If you build a tree by constructing an internal node for each adjacent pair that was combined, you get a “nonalphabetic” binary tree.



8

Each element of the input sequence corresponds to a leaf of this tree. Compute each leaf’s level, i.e., its distance from the root.



9

There is exactly one alphabetic tree each of whose leaves is at the required level.

History

a quadratic-time algorithm in [11]. A different method was described by Hu and Tucker in [8]; it is also presented in [5] and in the first edition of [12]. According to Knuth, “no simple proof [of the Hu–Tucker algorithm] is known, and it is quite possible that no simple proof will ever be found.” Then along came a modification of the Hu–Tucker algorithm, the Garsia–Wachs algorithm [3], which was adopted in the second edition of [12]. The best proof of its correctness, while still not exactly simple, is discussed in [10]; see also [9].

[3] Adriano M. Garsia and Michelle L. Wachs. A new algorithm for minimum cost binary trees. *SIAM Journal on Computing*, 6(4):622–642, 1977.

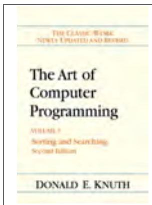
[9] Marek Karpinski, Lawrence L. Larmore, and Wojciech Rytter. Correctness of constructing optimal alphabetic tree revisited. *Theoretical Computer Science*, 180(1–2):309–324, 1997.

[10] Jeffrey H. Kingston. A new proof of the Garsia-Wachs algorithm. *Journal of Algorithms*, 9(1):129–136, 1988.

[12] Donald E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, Reading, MA, second edition, 1998.

10

Bird and Gibbons give a purely functional implementation of the Garsia–Wachs algorithm whose time complexity is O(n log n), but do not prove its correctness.



History

Hu and Tucker 1971
Garsia and Wachs 1977
Hu, Kleitman, and Tamaki 1979
Kingston 1988
Karpinski, Larmore, and Rytter 1997

T. C. Hu had been studying his own algorithm for the case $p_j = 0$ for several years; a rigorous proof of the validity of that algorithm was difficult to find because of the complexity of the problem, but he eventually obtained a proof jointly with A. C. Tucker [*SIAM J. Applied Math.* **21** (1971), 514–532]. Simplifications leading to Algorithm G were found several years later by A. M. Garsia and M. L. Wachs, *SICOMP* **6** (1977), 622–642, although their proof was still rather complicated. Lemmas W, X, Y, and Z above are due to J. H. Kingston, *J. Algorithms* **9** (1988), 129–136. Further properties have been found by M. Karpinski, L. L. Larmore, and W. Rytter, *Theoretical Comp. Sci.* **180** (1997), 309–324. See also the paper by Hu, Kleitman, and Tamaki, *SIAM J. Applied Math.* **37** (1979), 246–256, for an elementary proof of the Hu–Tucker algorithm and some generalizations to other cost functions.

11

Knuth was apparently happy with the proof given by Kingston, and presents it in the second edition of Volume 3 of *The Art of Computer Programming*.

Lemma W. *If $q_{k-1} > q_{k+1}$ then $l_k \le l_{k+1}$ in every optimum tree. If $q_{k-1} = q_{k+1}$ then $l_k \le l_{k+1}$ in some optimum tree.*

$q_{k-1} > q_{k+1} \iff q_{k-1} + q_k > q_k + q_{k+1}$

Lemma X. *Suppose j and k are indices such that $j < k$ and we have*
i) $q_{i-1} > q_{i+1}$ for $1 \le i < k$;
ii) $q_{k-1} \le q_{k+1}$;
iii) $q_i < q_{k-1} + q_k$ for $j \le i < k-1$; and
iv) $q_{j-1} \ge q_{k-1} + q_k$.
Then there is an optimum tree in which $l_{k-1} = l_k$ and either
a) $l_j = l_k - 1$, or
b) $l_j = l_k$ and $\boxed{j}$ is a left child.

(q_{k-1}, q_k) is a minimal adjacent pair.

Also, $l_j \le \ldots \le l_{k-1} = l_k$.

Lemma Y. *Let j and k be as in Lemma X, and consider the modified probabilities $(q'_0, \ldots, q'_{n-1}) = (q_0, \ldots, q_{j-1}, q_{k-1} + q_k, q_j, \ldots, q_{k-2}, q_{k+1}, \ldots, q_n)$ obtained by removing q_{k-1} and q_k and inserting $q_{k-1} + q_k$ after q_{j-1}. Then*

$$C(q'_0, \ldots, q'_{n-1}) \le C(q_0, \ldots, q_n) - (q_{k-1} + q_k). \quad (29)$$

Lemma Z. *Under the hypotheses of Lemma Y, equality holds in (29).*

The proofs of Lemmas X, Y, Z involve destructive operations on trees.

12

From Knuth 1998.
$C(q_0,\ldots,q_n)$ is the cost of an optimal alphabetic tree for $q_0,\ldots,q_n$.
Knuth’s proofs of these lemmas are slightly more detailed than Kingston’s, but are still quite informal and sketchy. Some details are relegated to exercises.

Dafny

- Syntax looks like an ordinary (imperative) programming language.
- Specification of
  - pre- and post-conditions of methods and functions,
  - termination metrics,
  - loop invariants.
- Lemmas are methods of a special kind.
- Automatic proof via SMT solver Z3.

13

I have formalized Knuth's proofs using Dafny.
Dafny web site (https://dafny.org/) says: "Dafny is a verification-aware programming language that has native support for recording specifications and is equipped with a static program verifier. By blending sophisticated automated reasoning with familiar programming idioms and tools, Dafny empowers developers to write provably correct code (w.r.t.

```
module LemmaW {

lemma Part1(t: Tree, qs: seq<int>, k: nat)
  requires |qs| >= 2
  requires forall i: nat :: i < |qs| ==> qs[i] >= 0
  requires 1 <= k < |qs| - 1
  requires |qs| == NumberOfLeaves(t)
  requires Level(t, k) > Level(t, k + 1)
  ensures t.Node?
  ensures var t1 := DeleteLeaf(t, k);
          var t2 := SplitLeaf(t1, k);
          Cost(t, qs) >= Cost(t2, qs) + qs[k - 1] - qs[k + 1] &&
          Level(t2, k) == Level(t2, k + 1)
{
  ...
}

lemma Part2(t: Tree, qs: seq<int>, k: nat)
  requires |qs| >= 2
  requires forall i :: 0 <= i < |qs| ==> qs[i] >= 0
  requires 1 <= k < |qs| - 1
  requires qs[k - 1] > qs[k + 1]
  requires |qs| == NumberOfLeaves(t)
  requires IsMinimalCostTree(t, qs)
  ensures Level(t, k) <= Level(t, k + 1)
{
  if Level(t, k) > Level(t, k + 1) {
    Part1(t, qs, k);
  } else {
```

14

The requires clauses (pre-conditions) are assumptions of the lemma. The ensures clauses are conclusions.

Some Formalization Decisions

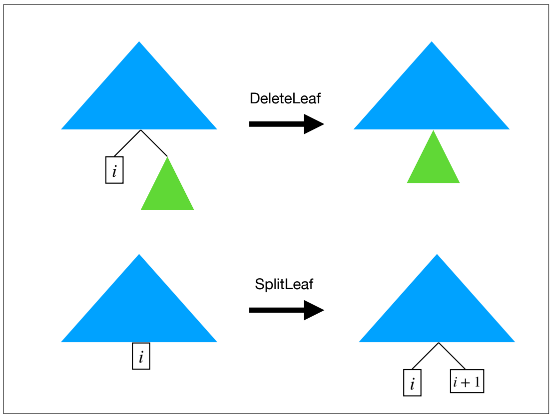
- Use unlabelled trees.

```
datatype Tree = Leaf | Node(left: Tree, right: Tree)

function Cost(t: Tree, qs: seq<int>): int
  requires |qs| == NumberOfLeaves(t)
```

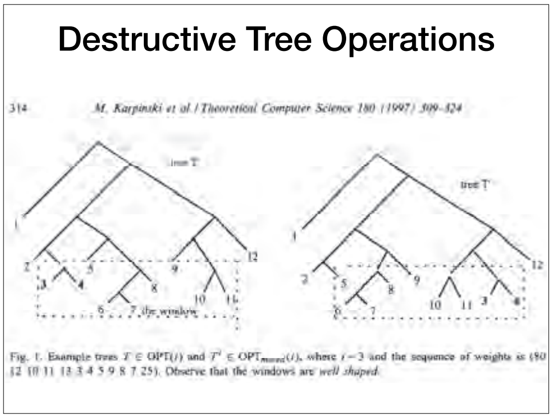
- Avoid complicated operations on trees.

```
opaque function DeleteLeaf(t: Tree, i: nat): (result: Tree)
  requires t.Node?
  requires i < NumberOfLeaves(t)
  ensures NumberOfLeaves(result) == NumberOfLeaves(t) - 1

opaque function SplitLeaf(t: Tree, i: nat): Tree
  requires i < NumberOfLeaves(t)
  ensures NumberOfLeaves(SplitLeaf(t, i)) ==
NumberOfLeaves(t) + 1
```

15

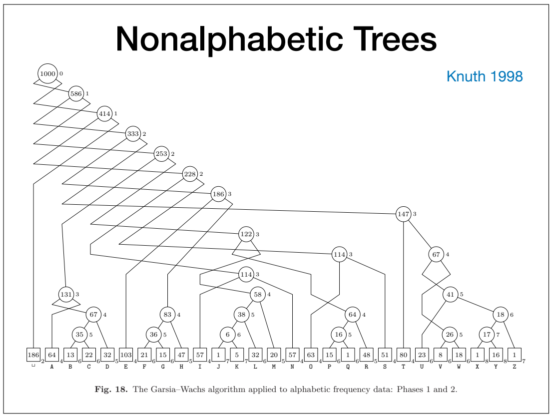


16



17

Karpinski et al.'s paper uses rather complex destructive operations on trees in their proof of correctness of the Garsia–Wachs algorithm. Such operations seem very hard to formalize. Luckily, the operations used by the Karpinski–Knuth proof are not as complicated. If you strip trees of their leaf labels, these operations are simple combinations of DeleteLeaf and SplitLeaf.



18

There is one aspect of the Garsia–Wachs algorithm that everyone seems to have overlooked. It concerns the intermediate nonalphabetic tree that is built by the algorithm.

Knuth 1998

G5. [Do phase 3.] By changing the links of $X_{n+1}, \ldots, X_{2n}$, construct a new binary tree having the same level numbers l_k, but with the leaf nodes in symmetric order $X_0, \ldots, X_n$. (See exercise 44; an example appears in Fig. 19.) ▮

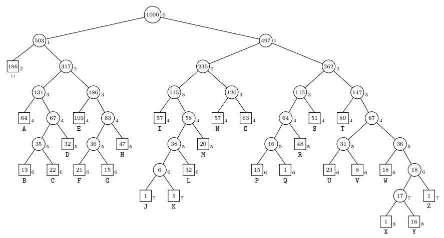


Fig. 19. The Garsia–Wachs algorithm applied to alphabetic frequency data: Phase 3.

▸ **44.** [25] Explain how to implement phase 3 of the Garsia–Wachs algorithm efficiently: Construct a binary tree, given the levels $l_0, l_1, \ldots, l_n$ of its leaves in symmetric order.

19

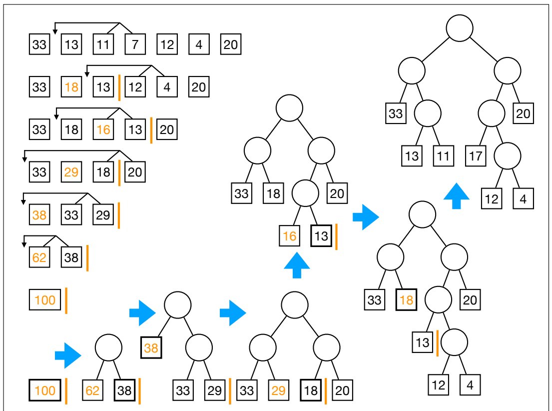
Direct Construction of the Alphabetic Tree

- The proof of Lemma Z in fact directly produces an alphabetic tree, and Lemma Y guarantees its optimality.

```
method Algorithm(qs: seq<int>) returns (t: Tree)
  decreases |qs|
  requires forall i: nat :: i < |qs| ==> qs[i] >= 0
  requires |qs| > 0
  ensures NumberOfLeaves(t) == |qs|
  ensures IsMinimalCostTree(t, qs)
{
  ...
    var qs' := qs[.. j] + [qs[k − 1] + qs[k]] + qs[j .. k − 1] + qs[k
+ 1 ..];
    ...
    var t' := Algorithm(qs');
    ...
    t := LemmaZ.Main(t', qs, j, k);
    assert NumberOfLeaves(t) == |qs|;
    ghost var u: Tree := MinimalCostTree(qs);
    ghost var u': Tree := LemmaY.Main(u, qs, j, k);
    assert Cost(t', qs') <= Cost(u', qs');
    assert Cost(t, qs) <= Cost(u, qs);
    assert IsMinimalCostTree(t, qs);
  ...
}
```

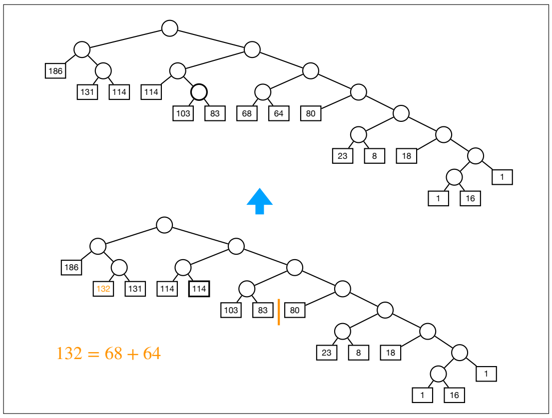
20

You pass the modified sequence qs' to the algorithm and get an optimum tree for qs'. Then the proof of Lemma Z turns it to an alphabetic tree for the original sequence qs for you. Its optimality is guaranteed by Lemma Y and Lemma Z.

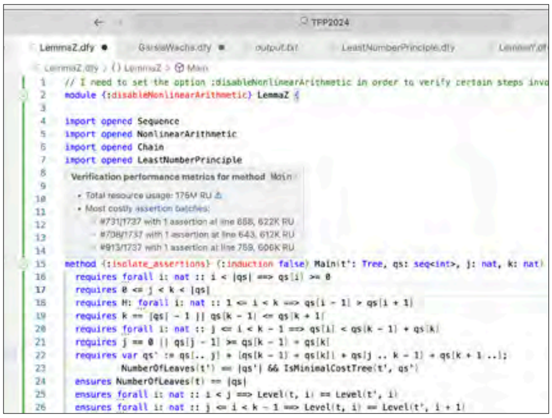


21

Between the moved item and its original position, you look for the **rightmost leaf** that is at the same level as the moved item. You split that leaf and relabel the leaves. This procedure produces an optimum alphabetic tree.



22



23

Experienced Dafny uses seem to want to limit the resource used to prove each assertion to 200K RU (resource units). I'm still far exceeding that limit.

Conclusion

- Verified the correctness of the Garsia–Wachs algorithm for building minimum cost alphabetic tree in Dafny.
- Complex destructive tree operations are not necessary in the proofs of the lemmas.
- The alphabetic tree can be constructed directly, without detour through a nonalphabetic tree; the method is implicit in the proof of Lemma Z.
- The fact that the Dafny verifier is fully automatic is both handy and frustrating.

24

Formal Verification of Payment Authentication Policies in the Blockchain

- Yoshihiro Imai : proof ninja, Inc.
- Mirai Ikebuchi : Kyoto University, proof ninja, Inc.

PROOF NINJA INC.

- DMM: Blockchain Engineer
- NII: OCaml Engineer
- Aist: Coq related tool
- NII: TopSE Lecturer

Participation in DMM’s web3 business as a subcontractor

This presentation will introduce one attempt at DMM.

PREREQUISITE KNOWLEDGE

Common accounting procedures in a company

1. Person in charge applies for expenses
2. The department manager approves the application
3. The accountant settles the payment

DOING BLOCKCHAIN PAYMENTS RIGHT

We want to ensure secure payments for digital assets on the blockchain.

FIREBLOCKS TAP

#	Initiator	Type	Source	Destination	Signer	Minimum	Asset	Action
01	Group 1	Transfer	Any vault	Any [illegible]	Ben Dover, Group 1	$0 per transaction	All	Require approval of 1 group
02	Group 1	Transfer	Any	Any Whitelisted address	Initiator	$10M in 1 day	All	Block
03	Group 1	Transfer	2 venues	3 venues Whitelisted address	Initiator	$100K per transaction	All	Require approval of 1 group
04	Group 1	Transfer	Any	Any Whitelisted address	Initiator	$0 per transaction	All	Allow

ISSUES WITH TAP CONFIGURATION

- Human error in configurations.
- Maintain safety while involving multiple departments.

REQUESTS FROM THE CEO

- Approval by someone other than the requester is required for large payments.
- Transfer destinations must be registered in the whitelist.
- Swap transactions can only be executed with pre-approved DEX contracts and currency pairs

I LOVE COQ

VERIFICATION WITH COQ

Rule.v

```
Record t := {
  initiator: Initiator;
  typ: Typ;
  source: Addresses;
  destination: Addresses;
  minimum: Minimum;
  action: Action;
  }.

Inductive Matches : Rule.t -> Payment.t -> Prop := ...
```

VERIFICATION WITH COQ

```
Definition req1 amount rules := ∀ payment,
    Rule.Available rules payment ->
    payment.(typ) = Transfer ->
    payment.(amount) >= amount ->
    ∃ approver, payment.(initiator) <> approver
        /\ IsApprover approver payment.
```

IMPLEMENTATION OF AN AUTOMATIC CHECKER

```
Definition check3 : whitelist -> list Rule.t -> sumbool _ _.

Lemma check3_sound whitelist rules:
  bool_of_sumbool (check3 whitelist rules) = true ->
  Requirement.req3 whitelist rules.

Extraction "coq_main.ml" Calc.check3 Draft2.draft2.
```

IMPLIMENTATION WITH OCAML AND COQ EXTRACTION

```
.
├── _CoqProject
├── bin
│   ├── Extr.v
│   ├── dune
│   ├── main.ml
│   └── ruleParser.ml
└── theories
    ├── Calc.v
    ├── Payment.v
    ├── Rule.v
    ├── Requirement.v
    └── dune
```

```
dune exec bin/main.exe
```

ANOTHER ATTEMPT IN DMM: OCAML2EVM

A platform to write Ethereum smart contracts in OCaml.

- GitHub ← Source Code
- OCamlでEthereumのスマートコントラクトを開発できるようにするプロジェクトの紹介 (Japanese article)

ANNOUNCEMENT: ROCQNAVI

An Extension of coq2html, Alternative to `coqdoc1.

- Generates HTML documentation from Coq source files
- Markdown and TeX Notations in doc Comments
- Hyperlinks of Notations

ANNOUNCEMENT: COQTOKYO

Coq study meeting

- Beginners are warmly welcome.
- Next session: Tuesday, 12/10, 7:00 PM~
- Join us here: https://readcoqart.connpass.com/

https://github.com/auto-res/HTPS-RAG

Automated Theorem Proving by HyperTree Proof Search with Retrieval-Augmented Tactic Generator

Sho Sonoda (RIKEN AIP), Naoto Onda (OMRON SINIC X),
Kei Tsukamoto (The University of Tokyo), Fumiya Uchiyama (The University of Tokyo),
Akiyoshi Sannai (Kyoto University), Wataru Kumagai (OMRON SINIC X)

https://github.com/auto-res/HTPS-RAG

Motivations

- **Large Language Models (LLMs) do best at generating languages ever**
- **So, we want to develop an AI system for automated ML research**

Input:
"Estimate the generalization error of the following network $f(x;\theta) = W\sigma(Vx)$"
→ LLM →
Output:
"$Risk[f] \leq EmpRisk[f] + \sqrt{\frac{c}{n}}$ with high probability where c is …"

- **But, LLMs are known to hallucinate, and have limited math abilities**

- **Possible solutions are**
  - **Prompt engineering --- in natural language**
  - **Using external computers --- in formal language (this study)**

- **In this study, we develop an LLM that perform Automated Theorem Proving (ATP) by interacting with Proof Assistant (Lean)**

https://github.com/auto-res/HTPS-RAG

Sequence Diagram of interactive proof by LLM and Proof Assistant

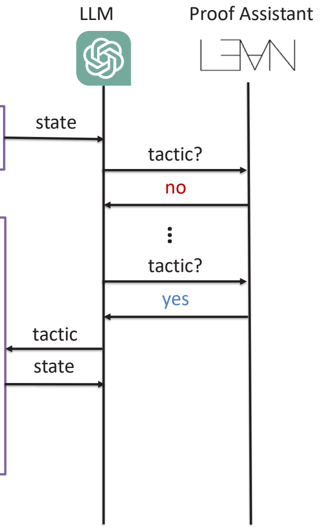
Eg. Markov inequality $P\{x \mid g(x) - f(x) \geq \varepsilon\} \leq \frac{1}{\varepsilon} E[g(X) - f(X)]$

Input (theorem)

```
theorem lintegral_add_mul_meas_add_le_le_lintegral {f g : α → ℝ≥0∞} (hle : f ≤ᵐ[μ] g)
  (hg : AEMeasurable g μ) (ε : ℝ≥0∞) :
  ∫⁻ a, f a ∂μ + ε * μ { x | f x + ε ≤ g x } ≤ ∫⁻ a, g a ∂μ := by
```

Sequential Output (proof)

```
rcases exists_measurable_le_lintegral_eq μ f with ⟨ϕ, hϕm, hϕ_le, hϕ_eq⟩
 calc
  ∫⁻ x, f x ∂μ + ε * μ { x | f x + ε ≤ g x } = ∫⁻ x, ϕ x ∂μ + ε * μ { x | f x + ε ≤ g x } := by
   rw [hϕ_eq]
  _ ≤ ∫⁻ x, ϕ x ∂μ + ε * μ { x | ϕ x + ε ≤ g x } := by
   gcongr
   exact fun x => (add_le_add_right (hϕ_le _) _).trans
  _ = ∫⁻ x, ϕ x + indicator { x | ϕ x + ε ≤ g x } (fun _ => ε) x ∂μ := by
   rw [lintegral_add_left hϕm, lintegral_indicator₀, setLIntegral_const]
   exact measurableSet_le (hϕm.nullMeasurable.measurable'.add_const _) hg.nullMeasurable
  _ ≤ ∫⁻ x, g x ∂μ := lintegral_mono_ae (hle.mono fun x hx₁ => ?_)
 simp only [indicator_apply]; split_ifs with hx₂
 exacts [hx₂, (add_zero _).trans_le <| (hϕ_le x).trans hx₁]
```



- This proof is written by human (obtained from Mathlib4)
- This "proof" is a sequence of tactics

---

https://github.com/auto-res/HTPS-RAG

Design Policy

- **Task:**
  - **Theorem Proving by LLM in Lean**
- **Why Lean?**
  - **The Lean math library, Mathlib, is well-developed and rapidly developing**
  - **supports practical math objects such as**
    - **concentration inequalities, and stochastic processes on $\mathbb{R}^n$**
- **Major Technologies:**
  - **Hyper-Tree Proof Search (HTPS)**
    - **Monte-Carlo Tree Search (MCTS)**
    - **+ Reinforcement Learning of networks**
  - **Premise Selection by Retrieval-Augmented Generator (RAG)**
    - **Vector search with machine-learned high-dim vector-embedding**

https://github.com/auto-res/HTPS-RAG

Hyper-Tree Proof Search (HTPS): AlphaZero-like formulation for Theorem Proving

	AlphaZero	HTPS
Task	Shogi, Go, …	Theorem Proving
State	Board state	Subgoals, or Proof States
Action	Putting stones	Tactics, or Proof Steps
Policy NW	suggests where to put the stone	suggests tactics
Critic NW	estimates the probability which player wins	estimates the provability of the given state

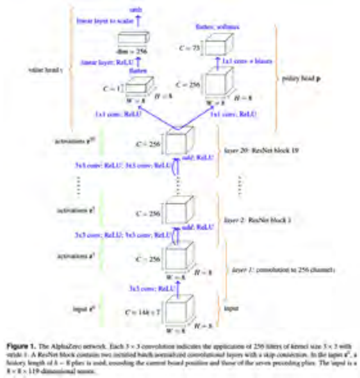


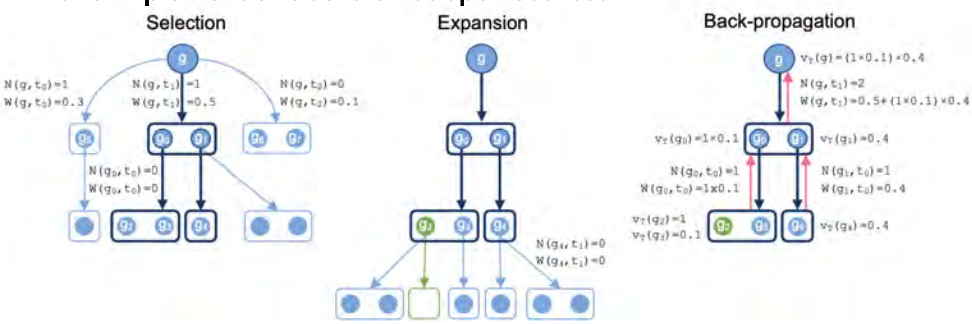
Figure from McGrath et al. pnas 119(47) 2022

[1] Lample et al., HyperTree Proof Search for Neural Theorem Proving. NeurIPS2022

https://github.com/auto-res/HTPS-RAG

Monte-Carlo Tree Search (MCTS) --- HTPS by Lample et al. (NeurIPS2022)

- **Repeat Select-Expand-Backpropagate to grow proof tree**
  - **Select a leaf node by running the policy NW to reach the leaf**
  - **Expand the leaf by applying the tactics suggested by the policy NW**
  - **Back-propagate the node values according to critic NW**
- **We re-implement this as it is not open-sourced**

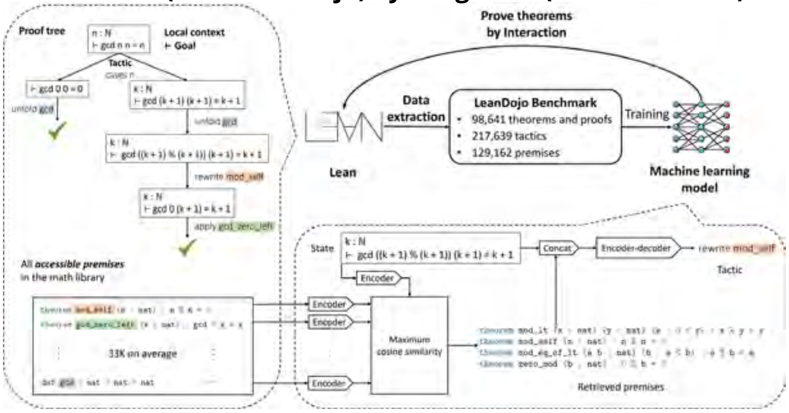


[1] Lample et al., HyperTree Proof Search for Neural Theorem Proving. NeurIPS2022

https://github.com/auto-res/HTPS-RAG

Premise Selection by Retrieval-Augmented Generator (RAG)

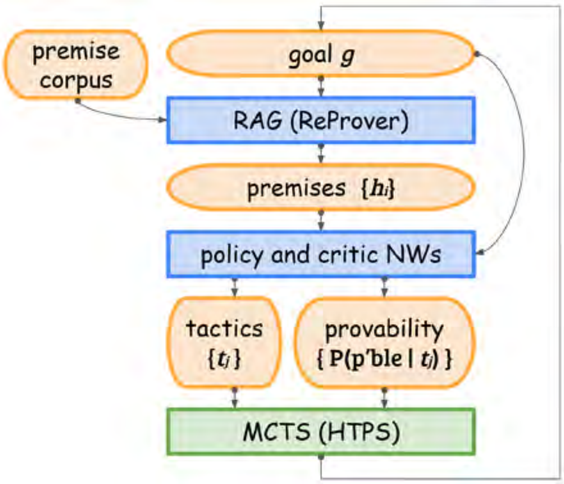
- **= vector search with machine-learned high-dim vector-embedding**
- **We use ReProver (from LeanDojo) by Yang et al. (NeurIPS2023dt)**



[2] Yang et al., LeanDojo: Theorem Proving with Retrieval-Augmented Language Models, NeurIPS2023 dataset track

https://github.com/auto-res/HTPS-RAG

Total Network Architecture



ReProver
ByT5 (encoder-only)

Policy NW:
ByT5 (encoder-decoder)
[g, h1, ..., hn] \mapsto [t_1,...,t_m]
(fixed-dimension input)
300M params

Ciritic NW:
ByT5 (endoder-decoder)
g \maptso value
300M params

https://github.com/auto-res/HTPS-RAG

Experimental Details and Results

- **ITP: Lean3**
- **NVIDIA A100-SXM4-80GB for 24 hours**
  - **Timeout for each run: 150 seconds**
  - **The number of premise selection: 20**

- **Training Dataset: Mathlib3**
- **Benchmark Datasets: MiniF2F and ProofNet**
  - **MiniF2F is high-school level**
  - **ProofNet is undergrad level**

- **Our model marked**
  - **26.2% on miniF2F by pass@1**
  - **10.0% on ProofNet by pass@1**

https://github.com/auto-res/HTPS-RAG

Score Boards

model	ITP	miniF2F	ProofNet
HTPS (2022)	Lean3	41.0%(pass@64)	-
LEGO Prover. (2023)	Isabelle	47.1%(pass@100)	-
ReProver (2023)	Lean3	26.5%	13.8%
Deep-Seek Prover V1.5 (2024.08.15)	Lean4	63.5% (SOTA)	23.5% (SOTA)
HTPS + RAG (**proposed**)	Lean3	26.2%(pass@1)	10.0%(pass@1)

https://github.com/auto-res/HTPS-RAG

Experimental Results

After a 24-hours of training,

- **The training loss for the critic model has decreased,**
- **but the training accuracy did not significantly improved**
- **Suggesting that we need a more machine power**

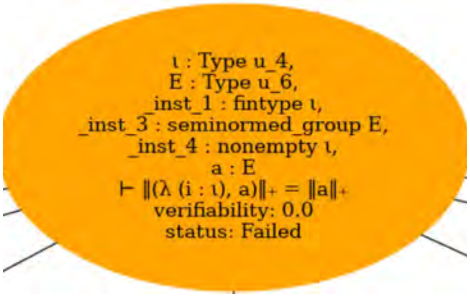
https://github.com/auto-res/HTPS-RAG

An Example of Success: Injectivity of homomorphism of fields

```
• theorem exercise_3_2_7 {F : Type*} [field F] {G : Type*} [field G]
(φ : F →+* G) : injective φ := begin
by_cases hφ : function.injective φ
intros x y h
exact hφ h
by_contra H
apply hφ
contrapose! hφ
rw injective_iff_map_eq_zero φ
contrapose! hφ
obtain ⟨x, hx1, hx2⟩ := hφ
exact φ.injective
end
```

https://github.com/auto-res/HTPS-RAG

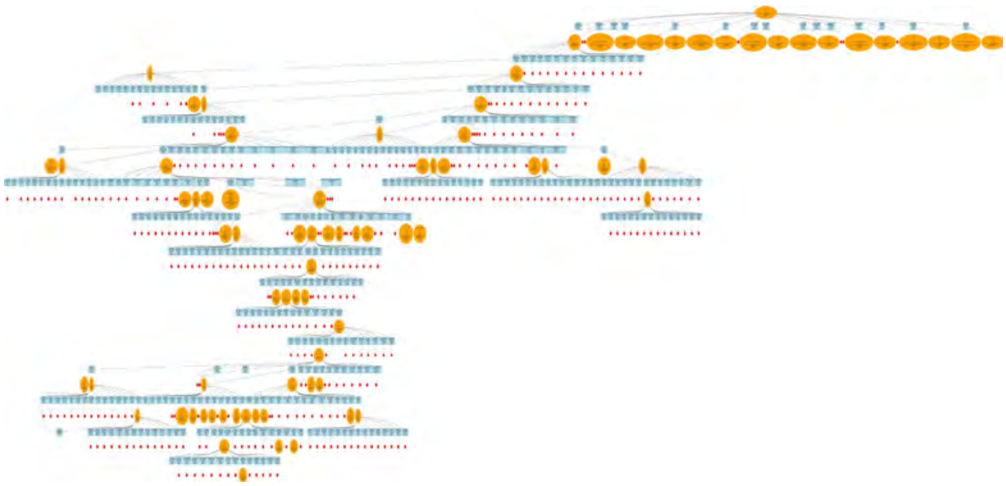
An Example of Failure: failed early (about 30%)



All suggested tactics failed to apply

https://github.com/auto-res/HTPS-RAG

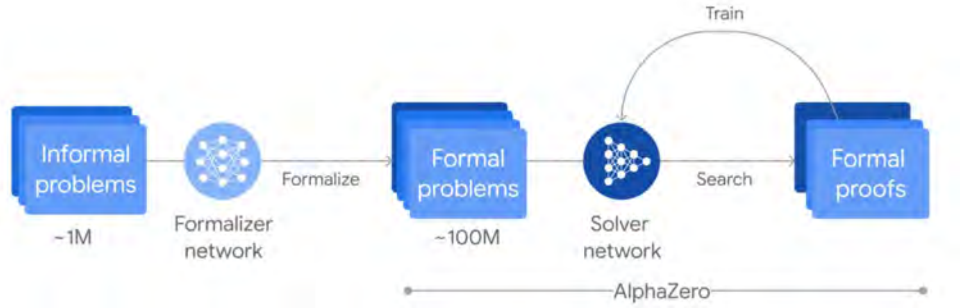
Another Example of Failure: search timed out (about 70%)



https://github.com/auto-res/HTPS-RAG

Latest Models: AlphaProof (Jul 25, 2024)

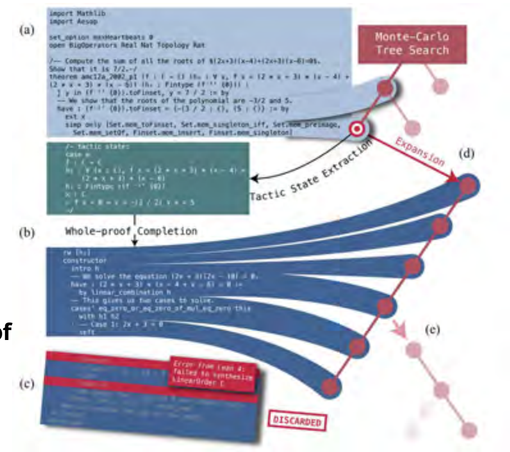
- **Augment the dataset by using the Formalizer Network**
- **“Our teams are continuing to explore multiple AI approaches for advancing mathematical reasoning and plan to release more technical details on AlphaProof soon.”**



https://github.com/auto-res/HTPS-RAG

Latest Models: Deep-Seek Prover V1.5 (Aug 15, 2024)

- **Using**
  - **Lean**
  - **Reinforcement Learning**
  - **Monte-Carlo Tree Search**
- **Additionally,**
  - **Chain of Thought reasoning as a guide of proof search**
  - **Use Lean’s feedback**
  - **Generates a tentative whole proof at each step for computational efficiency**



https://github.com/auto-res/HTPS-RAG

Conclusion/Limitation/Future Works

- **We need more computational resources and datasets**

- **E.g., we may increase the dataset by**
  - **auto-formalization**
  - **theorem generation**

- **Toward our final goal:**
  - **Auto-formalizer (on going)**
    - **How to verify the equivalence???**
  - **Manual dataset preparation (on going)**
    - **Particularly Hoeffding, Azuma-Hoeffding, Rademacher, Massart, …**
  - **Self-improvement by competitive game**
    - **How?**

Thank you for your attention

Isabelle/HOL による差分プライベートなアルゴリズムの安全な実装

松岡和貴 (東京科学大学　南出研究室)

TPP 2024　　2024/11/25

差分プライバシー

- プライバシーを守りながら、データを用いた計算結果を公開する際に用いる枠組みとして開発された概念 [Dwork et al., TCC 2006]
- 決定的な統計などの計算に確率的な操作を導入することで差分プライバシーは実現される
- 近年、実用化も進んでいる
    - Google Maps の混雑状況
    - 2020 年のアメリカの国勢調査 (10 年に 1 度行われる)

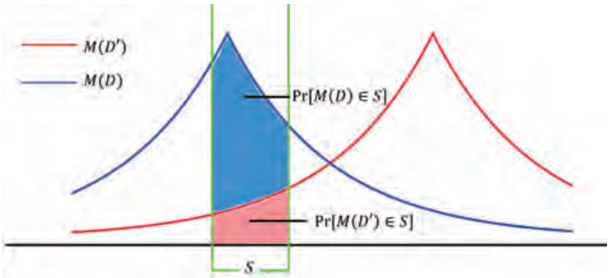
差分プライバシー

定義 (ϵ-差分プライバシー)

確率的アルゴリズム M が ϵ-差分プライバシーを満たすとは、エントリーが１つ異なるデータセット D, D' について以下が成り立つこと

$$\forall S \subset \mathrm{range}(M). \quad \Pr[M(D) \in S] \leq \exp(\epsilon) \cdot \Pr[M(D') \in S]$$

- ϵ として用いられる値は、0.1 – 20 の間がよく見られる



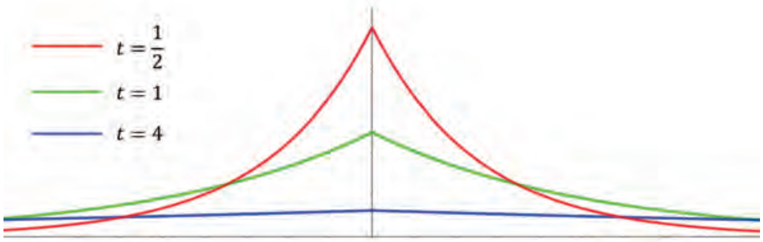
ラプラス分布

定義 (ラプラス分布)

ラプラス分布はスケールパラメータ t を持つ以下で定義される分布

$$\Pr[\mathrm{Lap}(t) = x] = \frac{1}{2t}\exp\left(-\frac{|x|}{t}\right)$$

- 差分プライベートなアルゴリズムにラプラス分布がしばしば用いられる



ラプラスメカニズム

定義 (ラプラスメカニズム)

データセットから決定的な計算をするアルゴリズム f に対して定まるラプラスメカニズム M_L とは、以下で定まる確率的アルゴリズム

$$M_L(D, f(\cdot), \epsilon) \overset{\text{def}}{=} f(D) + Y$$

ただし、Y は $\text{Lap}(\frac{\Delta f}{\epsilon})$ に従う確率変数、Δf は、f の感度

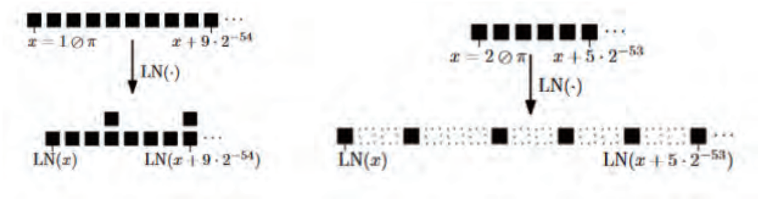
- ラプラス分布を用いた差分プライベートなアルゴリズムの代表例
- ラプラスメカニズムは、ϵ-差分プライバシーを満たす

本研究の概要

- 差分プライバシーが保証された実行可能なアルゴリズムのコードを得る
  - ラプラスメカニズム、Report Noisy Max アルゴリズム、Sparse Vector Technique
  - これらはラプラス分布を利用した差分プライベートなアルゴリズム
- このために以下を行う
  - 定理証明支援系である Isabelle/HOL を用いて、差分プライベートなアルゴリズムの証明を形式化する
  - 証明した差分プライベートなアルゴリズムから Isabelle/HOL のコード生成機能を用いる

ラプラスメカニズムの浮動小数点実装

- 連続確率分布を用いるアルゴリズムは、計算機上では近似的な方法でしか実装できない
- 浮動小数点数を用いた実装において、差分プライバシーが破れることが指摘されている [Mironov, CCS 2012]



浮動小数点数計算において丸めにより近似分布の形が崩れている様子

[Mironov. CSS 2012.
On significance of the least significant bits for differential privacy]

安全な実装のための対策

ラプラスメカニズムを同じ効果を持つような他のものに置き換える

- Snapping メカニズム [Mironov, CCS 2012]
  (浮動小数点数の計算誤差を考慮したアルゴリズム)
- 正確なサンプリングが可能な離散ラプラス分布を用いる
  [Ghosh et al., SICOMP 2012], [Canonne et al., JPC 2022]

本研究では、この離散ラプラス分布による実装を採用する

- ラプラス分布を用いたほかの差分プライベートなアルゴリズムにも用いることができる
- 証明がシンプル

離散ラプラス分布

定義 (離散ラプラス分布 [Ghosh et al., SICOMP 2012], [Canonne et al., JPC 2022])

離散ラプラス分布はスケールパラメータ t を持つ整数を台とする離散確率分布

$$\forall z \in \mathbb{Z}. \quad \Pr[\mathrm{Lap}_{\mathbb{Z}}(t) = z] = \frac{\exp(1/t) - 1}{\exp(1/t) + 1} \cdot \exp\left(-\frac{|z|}{t}\right)$$

- 離散ラプラスメカニズムは ϵ-差分プライバシーを満たす
- スケールパラメータが有理数の場合に、計算機上での正確なサンプリングが可能

本研究の概要

- 差分プライバシーが保証された実行可能なアルゴリズムのコードを得る
  - 離散バージョンのラプラスメカニズム、Report Noisy Max アルゴリズム、Sparse Vector Technique
  - これらは離散ラプラス分布を利用した差分プライベートなアルゴリズム
- このために以下を行う
  - 定理証明支援系である Isabelle/HOL を用いて、差分プライベートなアルゴリズムの証明を形式化する
  - 証明した差分プライベートなアルゴリズムから Isabelle/HOL のコード生成機能を用いる

本研究の概要

- 差分プライバシーが保証された実行可能なアルゴリズムのコードを得る
  - 離散バージョンのラプラスメカニズム、Report Noisy Max アルゴリズム、Sparse Vector Technique
  - これらは離散ラプラス分布を利用した差分プライベートなアルゴリズム
- このために以下を行う
  - 定理証明支援系である Isabelle/HOL を用いて、差分プライベートなアルゴリズムの証明を形式化する
  - 証明した差分プライベートなアルゴリズムから Isabelle/HOL のコード生成機能を用いる
- 離散ラプラス分布のサンプリングアルゴリズムの形式化から始める

確率的アルゴリズムの検証

Isabelle/HOL での検証は以下のようにして行う

- 検証の対象は離散確率アルゴリズム
- Isabelle/HOL 上の Probability ライブラリの劣確率質量関数の型 (spmf) への関数として定義
- このように定義した関数が、期待する性質を満たすことを証明
- 形式化した関数定義をコード生成で実行可能なコードを得る

サンプリングアルゴリズム

Algorithm 2 Algorithm for Sampling a Discrete Laplace

Input: Parameters $s, t \in \mathbb{Z}$, $s, t \geq 1$.
Output: One sample from $\mathrm{Lap}_{\mathbb{Z}}(t/s)$.
loop
  Sample $U \in \{0, 1, 2, \cdots, t-1\}$ uniformly at random.
  Sample $D \leftarrow \mathsf{Bernoulli}(\exp(-U/t))$.
  if $D = 0$ **then** reject and restart.
  Initialize $V \leftarrow 0$.
  loop
    Sample $A \leftarrow \mathsf{Bernoulli}(\exp(-1))$.
    if $A = 0$ **then** break the loop.
    if $A = 1$ **then** set $V \leftarrow V + 1$ and continue.
  Set $X \leftarrow U + t \cdot V$.
  Set $Y \leftarrow \lfloor X/s \rfloor$
  Sample $B \leftarrow \mathsf{Bernoulli}(1/2)$.
  if $B = 1$ and $Y = 0$ **then** reject and restart.
  return $Z \leftarrow (1 - 2B) \cdot Y$.

[Canonne et al. JPC 2022. The Discrete Gaussian for Differential Privacy]

サンプリングアルゴリズムの形式化

Algorithm 1 Algorithm for Sampling $\mathsf{Bernoulli}(\exp(-\gamma))$.

Input: Parameter $\gamma \geq 0$.
Output: One sample from $\mathsf{Bernoulli}(\exp(-\gamma))$.
if $\gamma \in [0, 1]$ **then**
  Set $K \leftarrow 1$.
  loop
    Sample $A \leftarrow \mathsf{Bernoulli}(\gamma/K)$.
    if $A = 0$ **then** break the loop.
    if $A = 1$ **then** set $K \leftarrow K + 1$ and continue the loop.
  if K is odd **then return** 1.
  if K is even **then return** 0.
else
  for $k = 1$ **to** $\lfloor \gamma \rfloor$ **do**
    Sample $B \leftarrow \mathsf{Bernoulli}(\exp(-1))$
    if $B = 0$ **then** break the loop and **return** 0.
  Sample $C \leftarrow \mathsf{Bernoulli}(\exp(\lfloor \gamma \rfloor - \gamma))$
  return C.

[Canonne et al. JPC 2022. The Discrete Gaussian for Differential Privacy]

```
definition bernoulli_exp_minus_rat :: "rat  ⇒ bool spmf" where
  "bernoulli_exp_minus_rat p =
  (
    if p < 0 then return_spmf True
    else if 0 ≤ p & p≤1  then bernoulli_exp_minus_rat_from_0_to_1 p
    else
     do {
        b ← bernoulli_exp_minus_rat_loop p (nat (floor p));
        if b then bernoulli_exp_minus_rat_from_0_to_1 (p- (of_int (floor p))) else return_spmf b
      }
  )"
```

サンプリングアルゴリズムの検証

- サンプリングアルゴリズムの定義

```
partial_function (spmf) discrete_laplace_rat :: "nat ⇒ nat ⇒ int spmf" where
"discrete_laplace_rat t s = do {
    x::nat ← discrete_laplace_rat_unit t;
    b::bool ← bernoulli_rat 1 2;
    let y = calculate_y x s in
    if (¬b ∧ (y=0)) then discrete_laplace_rat t s
    else if b then return_spmf (-y)
         else return_spmf y
}"
```

- 確率 1 で停止することの証明

```
lemma lossless_discrete_laplace_rat[simp]:
  assumes "1≤t" and "1≤s"
  shows "lossless_spmf (discrete_laplace_rat t s)"
```


サンプリングアルゴリズムの検証

- 離散ラプラス分布からのサンプリングであることの証明

```
lemma spmf_discrete_laplace_rat:
  assumes "1≤t" and "1≤s"
  shows "spmf (discrete_laplace_rat t s) z = (exp(s/t)-1)/(exp(s/t)+1) * exp(- s*¦z¦/t)"
```

定義 (離散ラプラス分布 [Ghosh et al., SICOMP 2012], [Canonne et al., JPC 2022])

スケールパラメータ $\frac{t}{s}$ の離散ラプラス分布は以下で定義される

$$\forall z \in \mathbb{Z}.\quad \Pr[\mathrm{Lap}_{\mathbb{Z}}\left(\frac{t}{s}\right) = z] = \frac{\exp(s/t)-1}{\exp(s/t)+1} \cdot \exp\left(-\frac{s \times |z|}{t}\right)$$

差分プライバシーアルゴリズムの検証

- 離散確率分布に対する差分プライバシーの形式化

```
definition pure_dp_inequality :: "'a spmf ⇒ 'a spmf ⇒ real ⇒ bool" where
"pure_dp_inequality M N ε = (∀A::'a set. measure (measure_spmf M) A ≤ exp(ε) * measure (measure_spmf N) A)"

definition pure_dp :: "('a,'b) mechanism ⇒ real ⇒ bool" where
"pure_dp M ε = (∀l1 l2. neighbour l1 l2 ⟶ pure_dp_inequality (M l1) (M l2) ε)"
```

- 離散ラプラスメカニズム

```
definition discrete_laplace_mechanism :: "('a list ⇒ int) ⇒ nat  ⇒ nat ⇒ nat ⇒ 'a list ⇒ int spmf" where
"discrete_laplace_mechanism f Δ epsilon1 epsilon2 x =
  do { noise ← discrete_laplace_rat (epsilon2 * Δ) epsilon1;
       return_spmf (noise + f x)
    }"
```

- 離散ラプラスメカニズムが差分プライバシーを満たすことの証明

```
lemma pure_dp_discrete_laplace_mechanism:
  assumes "is_sensitivity f Δ"
and "1 ≤ epsilon1"
and "1 ≤ epsilon2"
and "1 ≤ Δ"
shows "pure_dp (discrete_laplace_mechanism f Δ epsilon1 epsilon2) (epsilon1/epsilon2)"
```


実行可能なコード生成

- Isabelle/HOL には標準で ML 系の言語や Scala に対するコード生成機能がある
- コード生成機能を用いると、spmf は値とその値の確率の組の Map の型となり、確率的な実行はできない
- Archive of Formal Proofs の Executable Randomized Algorithms を利用して解決
  - 遅延評価の bit 乱数列を受け取って、結果と残りの bit 乱数列を返す型 random_alg を定義
  - spmf とのモナドとしての対応がライブラリ内で証明済み
  - コード生成後に、生成先のプログラムで遅延評価の bit 乱数列だけ用意すればいい

Executable Randomized Algorithms

```
typedef 'a random_alg = "{(r :: 'a random_alg_int). wf_random r}"
type_synonym 'a random_alg_int = "coin_stream ⇒ ('a × coin_stream) option"
```

```
partial_function (spmf) binary_geometric_spmf :: "nat ⇒ nat spmf"
  where
    "binary_geometric_spmf n =
      do { c ← coin_spmf;
           if c then (return_spmf n) else binary_geometric_spmf (n+1)
      }"
```

型random_algの
対応する関数を定義

```
partial_function (random_alg) binary_geometric :: "nat ⇒ nat random_alg"
  where
    "binary_geometric n =
      do { c ← coin_ra;
           if c then (return_ra n) else binary_geometric (n+1)
      }"
```

- 実行可能なコードが spmf での定義と正しく対応していることを証明

```
lemma binary_geometric_ra_correct:
  "spmf_of_ra (binary_geometric x) = binary_geometric_spmf x"
```


SampCert(関連研究)

- AWS の研究グループにより公開されている OSS
- 定理証明支援系である Lean による差分プライベートな
  アルゴリズムの安全な実装
- 対象としているのは離散ラプラス分布を用いるもの
- 2024 年の 5 月に Ver1.0.0 としてリリースされ、現在も開発が
  進んでいる

まとめ・今後の展望

- 定理証明支援系である Isabelle/HOL を用いて、差分プライベートなアルゴリズムの安全な実装を得る
  - 離散バージョンのラプラスメカニズム、Report Noisy Max アルゴリズム、Sparse Vector Technique
  - これらは離散ラプラス分布を利用した差分プライベートなアルゴリズム
- 離散ラプラスメカニズムまで完了、今後残りの差分プライベートなアルゴリズムに取り組んでいく

標準的な様相論理の Lean での形式化について

野口 真柊

神戸大学システム情報学研究科 研究生

2024-11-24

このスライドは https://sno2wman.github.io/slides-for-tpp2024/main.pdf で閲覧出来ます.

Outline

1.1 様相論理

定義 1.1.1: 古典的な命題論理に 1 項の様相演算子 $\Box, \Diamond$ を加えて拡張した論理体系一般を，標準的な様相論理という．

例: $\Box p$ を「pは必然的に起こる」と解釈する． $\Diamond$ を $\Box$ の双対として定義すると（$\Diamond p :\equiv \neg\Box\neg p$）「$p$ が起こらないことは必然的でない」すなわち「p が起こる可能性がある」と解釈できる．

1.1 様相論理

例: $\Box p$ をどのような様相として解釈するかで，様々な様相論理が定義できる．

- 義務「p は義務である」.
  - $\Box p \to \Diamond p$「p でなければならないなら，p でもよい」
- 知識「p を知っている」.
  - $\Box p \to \Box\Box p$「p を知っているなら，p を知っていることも知っている」
  - $\Box p \to p$「p を知っているなら p は正しい.」
- (形式的)証明可能性「p は証明可能である」

様相論理について　様相論理の Lean での形式化　参考文献

1.1 様相論理

定義 1.1.2 (Kripke 意味論):

- 世界の非空集合 W と W 上の 2 項関係 $R : W \times W$ の組 $\langle W, R \rangle$ を Kripke フレームという．
- フレーム $\langle W, R \rangle$ とその上の付値関数 $V : \mathrm{Var} \times W \to 2$ の組 $\langle W, R, V \rangle$ を Kripke モデルという．
- 論理式の充足関係は世界に依存して決まる．$M = \langle W, R, V \rangle$ と $x \in W$ に対して，
  - $M, x \vDash p \iff V(x, p) = 1$「$x$ 上で p が真である」
  - $M, x \vDash \Box\varphi \iff \forall y, xRy \implies V(w, p) = 1$「$x$ から行ける全ての世界で φ は真」
- モデル M に対し，全ての世界で φ が充足されるなら，$M \vDash \varphi$「φ はモデル M で妥当」という．
- フレーム F 上の任意のモデルで φ が妥当であるなら，$F \vDash \varphi$「φ はフレーム F で妥当」という．

Outline

2.1 Formalized Formal Logic

Formalized Formal Logic https://github.com/FormalizedFormalLogic

数理論理学の様々な事実や定理を Lean4 で形式化するプロジェクト．自分と齋藤氏が中心となって進めている．

2.1 Formalized Formal Logic

齋藤氏によってこれまで形式化された事実．[1]

- 古典命題論理
  - 健全性と完全性
  - 自動証明
- 古典 1 階述語論理と算術
  - 健全性と完全性
  - カット除去定理
  - Gödel の不完全性定理
    - 明日の齋藤氏の発表で詳しく説明される．

[1] 自動証明とカット除去定理と発表時では一旦書き直しのため破棄されている．

様相論理について　様相論理の Lean での形式化　参考文献

2.1 Formalized Formal Logic

自分がこれまでに形式化した事実

- 直観主義命題論理
  - Kripke 意味論
  - 選言特性
  - Gödel-McKensey-Tarski の定理
- 標準的な様相論理
  - Kripke 意味論と，いくつかの論理に対しての完全性
    - Geach 論理
    - 証明可能性論理 GL
  - 濾過法

今回の発表では強調した箇所について手短に説明する．

2.2 様相論理

定義 2.2.1:
- 以下の公理と推論規則を持つ Hilbert 流証明体系で証明可能な論理式の集合を論理 $\mathbf{K}$ とする．
  - 古典命題論理のトートロジー
  - 公理 K : $\Box(\varphi \to \psi) \to \Box\varphi \to \Box\psi$
  - モーダスポネンス (MP) : $\varphi \to \psi, \varphi \mid \psi$
  - ネセシテーション (Nec) : $\varphi \mid \Box\varphi$

様相論理について　様相論理の Lean での形式化　参考文献

2.2 様相論理

定義 2.2.2: 以下は $\mathbf{K}$ では証明できず，公理と呼ばれる．

$$\begin{aligned} \mathsf{T} &: \Box\varphi \to \varphi & \mathsf{D} &: \Box\varphi \to \Diamond\varphi \\ \mathsf{B} &: \varphi \to \Diamond\Box\varphi & \mathsf{4} &: \Box\varphi \to \Box\Box\varphi \\ \mathsf{5} &: \Diamond\varphi \to \Box\Diamond\varphi & .\mathsf{2} &: \Diamond\Box\varphi \to \Box\Diamond\varphi \\ \mathsf{L} &: \Box(\Box\varphi \to \varphi) \to \Box\varphi \end{aligned}$$

定義 2.2.3: 公理 K に加えて更に公理を加えて拡張することで得られる論理を挙げる．

$$\begin{aligned} \mathbf{KT} &= \mathsf{K} + \mathsf{T} \\ \mathbf{S4} &= \mathsf{K} + \mathsf{T} + \mathsf{4} \qquad \mathbf{S5} = \mathsf{K} + \mathsf{T} + \mathsf{5} \\ \mathbf{KT4B} &= \mathsf{K} + \mathsf{T} + \mathsf{4} + \mathsf{B} \\ \mathbf{GL} &= \mathsf{K} + \mathsf{L} \end{aligned}$$

2.3 Geach 論理

定義 2.3.1: 2 項関係 R に対し $\langle i, j, m, n \rangle \in \mathbb{N}^4$ として以下が成り立つなら $\langle i, j, m, n \rangle$-合流的であるという．

$$\forall x, y, z. [xR^i y \land xR^j z \implies \exists u. [yR^m u \land zR^n u]]$$

ただし 2 項関係の n 回合成 R^n は以下のように定める．

- $xR^0 y \iff x = y$.
- $xR^{n+1} y \iff \exists z, xR^n z \;\&\; zRy$

例: 2 項関係のいくつかの性質は $\langle i, j, m, n \rangle$-合流性で一般化できる．

- 反射性 $xRy \implies yRx$ は $\langle 0, 0, 1, 0 \rangle$-合流性
- 推移性 $xRy \;\&\; yRz \implies xRz$ は $\langle 0, 2, 1, 0 \rangle$-合流性

様相論理について　様相論理の Lean での形式化　参考文献

2.3 Geach 論理

定義 2.3.2: $\langle i, j, m, n \rangle$ に対して以下を Geach 公理 $\mathsf{ga}_{i,j,m,n}$ という．

$$\mathsf{ga}_{i,j,m,n} \equiv \Diamond^i \Box^m \varphi \to \Box^j \Diamond^n \varphi$$

定義 2.3.3: $\mathbf{K}$ にいくつかの Geach 公理 $\mathsf{ga}_{i_1,j_1,m_1,n_1}, ..., \mathsf{ga}_{i_k,j_k,m_k,n_k}$ 入れて拡張した論理を Geach 論理 $\mathbf{Ge}(\langle i_1, j_1, m_1, n_1 \rangle, ..., \langle i_k, j_k, m_k, n_k \rangle)$ と書く．

2.3 Geach 論理

例: 公理 T, D, B, 4, 5, .2 は Geach 公理であり，$\mathbf{K}$ 及び 定義 2.2.3 のいくつかの論理は Geach 論理である．

$$
\begin{aligned}
\mathbf{K} &= \mathbf{Ge}() \\
\mathbf{KT} &= \mathbf{Ge}(\langle 0,0,1,0\rangle) \\
\mathbf{S4} &= \mathbf{Ge}(\langle 0,0,1,0\rangle, \langle 0,2,1,0\rangle) \\
\mathbf{S5} &= \mathbf{Ge}(\langle 0,0,1,0\rangle, \langle 1,1,0,1\rangle) \\
\mathbf{KT4B} &= \mathbf{Ge}(\langle 0,0,1,0\rangle, \langle 0,2,1,0\rangle, \langle 0,1,0,1\rangle) \\
\mathbf{S4.2} &= \mathbf{Ge}(\langle 0,0,1,0\rangle, \langle 0,2,1,0\rangle, \langle 1,1,1,1\rangle)
\end{aligned}
$$

2.3 Geach 論理

定理 2.3.4 (Geach 論理の Kripke 完全性): $\mathbf{Ge}(t_1, ...t_k)$ は t_1-合流性,⋯,t_k-合流性を満たすフレームのクラスに対して健全かつ完全である． すなわち，次は同値である．

1. $\mathbf{Ge}(t_1, ...t_k)$ で φ が証明可能: $\mathbf{Ge}(t_1, ..., t_k) \vdash \varphi$
2. t_1-合流性,⋯,t_k-合流性を満たす全てのフレーム F で φ は妥当: $F \vDash \varphi$

系 2.3.5: $\mathbf{K}$ に 公理 T, D, B, 4, 5, .2 を適当に付け加えた論理たちは全て対応するフレームクラスに対して完全である． 例えば次のことが成り立つ．

- $\mathbf{S4}$ は 反射的/推移的，すなわち前順序のフレームのクラスについて完全である．
- $\mathbf{KT4B}$ は 反射的/推移的/対称的，すなわち同値関係のフレームのクラスについて完全である．
- $\mathbf{S5}$ は 反射的/推移的/Euclid 的なフレームのクラスについて完全である．

```
instance S4.Kripke.complete : Complete (Hilbert.S4 ℕ) ReflexiveTransitiveFrameClass

instance S5.Kripke.complete : Complete (Hilbert.S5 ℕ) ReflexiveEuclideanFrameClass
```

2.3 Geach 論理

系 2.3.6:

- 同値関係なフレームは反射的/推移的/Euclid 的であり，逆も成り立つ．そのため，**KT4B** と **S5** は証明能力が等しい．
- 前順序だが同値関係ではないフレームが存在するため，**KT4B** は **S4** より真に強い．

$$\mathbf{S4} \subsetneq \mathbf{KT4B} = \mathbf{S5}$$

```
theorem S4_strictlyWeakerThan_S5 : (Hilbert.S4 ℕ) <ₛ (Hilbert.S5 ℕ)

theorem equiv_S5_KT4B : (Hilbert.S5 ℕ) =ₛ (Hilbert.KT4B ℕ)
```


2.4 GL

様相論理 **GL** は不完全性定理で言及される証明可能性を分析するための様相論理：証明可能性論理として盛んに研究が行われている． まず，Kripke 意味論に対して以下の性質が成り立つ[1]．

定理 2.4.1: **GL** は 推移的かつ非反射的な有限なフレームのクラスに対して健全かつ完全，すなわち有限フレーム性を持つ．[2]

```
instance : Kripke.FiniteFrameProperty (Hilbert.GL ℕ) TransitiveIrreflexiveFiniteFrameClass
```

[1] この事実は M. Maggesi and C. Perini Brogi [1] によって HOL Light で既に形式化が成されている．
[2] なお，有限性を要請せず単に非反射的なフレームのクラスに対して完全な様相論理は存在しない．この事実も形式化されている．

2.4 GL

更に強い事実が成り立つ.

定理 2.4.2: 次は同値.
1. $\mathbf{GL} \vdash \varphi$
2. 任意の推移的な木構造を持つ有限フレーム上のモデル M と，その根 r で $M, r \vDash \varphi$

```
theorem provable_iff_satisfies_at_root_on_FiniteTransitiveTree
  : (Hilbert.GL ℕ) ⊢! φ ↔ (∀ M : FiniteTransitiveTreeModel, Satisfies M.toModel M.root φ) :
```


様相論理について 様相論理の Lean での形式化 参考文献

2.4 GL

ここから更に次の系が導ける.

系 2.4.3: GL ではデネセシテーションは許容規則．すなわち,

$$\mathbf{GL} \vdash \Box\varphi \Longrightarrow \mathbf{GL} \vdash \varphi$$

```
theorem GL_unnecessitation!  : (Hilbert.GL ℕ) ⊢! □φ → (Hilbert.GL ℕ) ⊢! φ
```

系 2.4.4: GL は様相選言特性を持つ．すなわち,

$$\mathbf{GL} \vdash \Box\varphi \lor \Box\psi \Longrightarrow \mathbf{GL} \vdash \varphi \parallel \mathbf{GL} \vdash \psi$$

```
theorem GL_MDP (h : (Hilbert.GL ℕ) ⊢! □φ₁ ⋎ □φ₂) : (Hilbert.GL ℕ) ⊢! φ₁ ∨ (Hilbert.GL ℕ) ⊢! φ₂
```

2.4 GL

定理 2.4.5 (第 2 不完全性定理): T を **PA** の無矛盾な再帰的可算な拡大理論とする．このとき，次を満たす証明可能性述語 $\mathbf{Pr}_T$ を構成できる．

T の無矛盾性を表す文 $\mathbf{Con}_T :\equiv \mathbf{Pr}_T(\ulcorner\bot\urcorner)$ は T で証明できない．すなわち $T \nvdash \mathbf{Con}_T$.

```
abbrev bewₐ (σ : Sentence ℒₒᵣ) : Sentence ℒₒᵣ := U.provableₐ/[⌜σ⌝]

abbrev consistentₐ : Sentence ℒₒᵣ := ~U.bewₐ ⊥

theorem goedel_second_incompleteness [System.Consistent T] : T ⊬ T.consistentₐ
```

不完全性定理と GL の形式化は成されている．よって，証明可能性論理の礎となる算術的完全性という定理も形式化したい．

様相論理について　様相論理の Lean での形式化　参考文献

2.4 GL

定義 2.4.6:

- 様相論理の命題変数を 1 階述語論理上の算術の文へと写す写像 $* : \mathrm{Ver}_{\mathcal{L}_\mathrm{M}} \to \mathrm{Sent}_{\mathcal{L}_\mathrm{A}}$ を解釈と呼ぶ．
- $\Box$ を証明可能性述語 $\mathbf{Pr}_T$ として見るように解釈 $*$ を様相論理式へと拡張した写像 $*_{\mathbf{Pr}_T} : \mathrm{Form}_{\mathcal{L}_\mathrm{M}} \to \mathrm{Sent}_{\mathcal{L}_\mathrm{A}}$ を $\mathbf{Pr}_T$-翻訳と呼ぶ．すなわち次を満たす．

$$p^{*_{\mathbf{Pr}_T}} \mapsto p^*$$
$$\bot^{*_{\mathbf{Pr}_T}} \mapsto \bot$$
$$(\varphi \to \psi)^{*_{\mathbf{Pr}_T}} \mapsto \varphi^{*_{\mathbf{Pr}_T}} \to \psi^{*_{\mathbf{Pr}_T}}$$
$$(\Box\varphi)^{*_{\mathbf{Pr}_T}} \mapsto \mathbf{Pr}_T(\ulcorner\varphi^{*_{\mathbf{Pr}_T}}\urcorner)$$

2.4 GL

定理 2.4.7 (GL の算術的完全性定理): Pr_T は 定理 2.4.5 で構成できる証明可能性述語であるとする. このとき次は同値である.
1. $\mathbf{GL} \vdash \varphi$
2. 任意の解釈 $*$ に対して $T \vdash \varphi^{*\mathrm{Pr}_T}$.

1.から 2.は帰納法を回せば良いだけなので簡単.

```
lemma arithmetical_soundness_GL [𝔅.HBL] (h : 𝐆𝐋 ⊢! p) : ∀ {f : Realization α L}, U ⊢!. (f.interpret 𝔅 p)
```

2.から 1.は難しく, 形式化出来ていない. 道具立て（多変数版の対角化補題, GL の木フレームに対する完全性, etc.）自体は既に済んでいるので, あとは地道にやるだけだとは思うが難航している.

2.5 様相論理：その他

話せなかったが形式化が済んでいる内容

- 濾過法と有限フレーム性
  - ただし有限フレーム性が言えたからといって, Lean の中で決定可能性が形式化できそうかは不明 (「形式化された決定可能性」のようなものを定義する必要がありそう).
- 直観主義論理およびいくつかの中間論理の Kripke 完全性
  - その系として, 排中律の非成立と選言特性
- 直観主義論理と様相論理の関連性
  - Gödel-McKensey-Tarski の定理：$\mathbf{Int}, \mathbf{S4}$ の Modal Companion.
- 規則 (Löb), (Henkin) を用いた GL の別定義
- $\mathbf{Triv}$ と $\mathbf{Ver}$ が古典命題論理に帰着できること
- $\mathbf{Grz}$ の Kripke 完全性
- $\mathbf{GL}, \mathbf{Grz}$ の Boxdot Companion[1]
- Pure Logic of Necessitation $\mathbf{N}$ の意味論とその完全性

[1] GL で証明できる論理式の □ の出現を全て ⊡（ただし $\boxdot\varphi :\equiv \varphi \land \Box\varphi$）に置き換えた論理式は Grz で証明できる.

2.6 今後の展望

まだまだ多くのことが残されている！

- 強完全性
- GL の算術的完全性
- GL の不動点定理
- 様相論理の種々の計算体系および自動証明
- 様相論理の代数的意味論
- Modal Companion
- Kripke 不完全な様相論理
- その他多くの様相論理の事実
  - Makinson の定理
  - Boxdot Companion
- 補間定理
- その他の論理体系: Hybrid Logic, 様相 μ 計算など

2.7 今後の展望: 様相論理の種々の計算体系の形式化および自動証明

GL の普通のシークエント計算は論理式の集合を多重集合にするか集合にするかで議論の微細な違いが生じ，その他にも複雑な帰納法を回すため，証明が正確に行われているのか議論の余地があった[2], [3].

近年 R. Goré, R. Ramanayake, and I. Shillito [4]らによって GL のシークエント計算の形式化が Coq でなされた．まずはこれを Lean で再実装してみたい．

その他にも様相論理にはいくつかの計算体系：シークエント計算の拡張，あるいはタブロー計算がある．拡張されたシークエント計算の形式化は知る限りほぼ見たことがなく，紙とペンの証明では上手く証明が回っていないことがある（らしい）ので そういったところを形式化することで厳密に詰めてみたい．

余談: なお近年 Coq による証明可能性論理（特に直観主義証明可能性論理 iGL, iSL など）に関連する証明論の形式化が近年盛んに行われている．

2.8 今後の展望: 様相論理の代数的意味論 / Modal Companion

Boolean 代数に適当な □ 演算を入れて拡張した代数を考えることで様相論理に代数的による意味論を与えられる． この代数を定理証明支援系で形式化するモチベーションはいくつかある．

1. Kripke フレームをコードで扱うのはやや面倒.
   - 逆に代数的な操作は形式化の上では扱いやすいという話がある[1].
2. より一般的に中間論理と様相論理の Modal Companion を考えるときにほぼ必須の知識.
3. その他にも Goldblatt-Thomason の定理や Sahlqvist の定理なども形式化するなら避けられないと思われる.

[1] 実際にそうなのかは不明

2.9 今後の展望: 様々な論理体系の形式化

今 Formalized Formal Logic で扱っているのは

- 古典命題論理， 直観主義命題論理
- 古典 1 階述語論理とその算術
- 標準的な □, ◇ の様相論理

もっと様々な論理体系を形式化してみたい[1].

- 様相 μ-計算
- Hybrid Logic
- Lax Logic
- (多エージェント)認識論理
- 動的論理
- 線形論理

特に Lean はプログラミング言語としても扱える（ことになっている）ので， これらの論理体系を形式化して応用的なソフトウェアなどを作ることも可能かもしれない．

[1] 様々な定理証明支援系でいくつかの実装がある． https://formalizedformallogic.github.io/Book/references.html 参照.

Outline

3. 参考文献

[1] M. Maggesi and C. Perini Brogi, "Mechanising Gödel–Löb Provability Logic in HOL Light," Journal of Automated Reasoning, vol. 67, no. 3, p. 29, Sep. 2023, doi: 10.1007/s10817-023-09677-z.

[2] S. Valentini, "The modal logic of provability: Cut-elimination," Journal of Philosophical Logic, vol. 12, no. 4, pp. 471–476, Nov. 1983, doi: 10.1007/BF00249262.

[3] R. Goré and R. Ramanayake, "Valentini's Cut-Elimination for Provability Logic Resolved," The Review of Symbolic Logic, vol. 5, no. 2, pp. 212–238, Jun. 2012, doi: 10.1017/S1755020311000323.

[4] R. Goré, R. Ramanayake, and I. Shillito, "Cut-Elimination for Provability Logic by Terminating Proof-Search: Formalised and Deconstructed Using Coq," Automated Reasoning with Analytic Tableaux and Related Methods, vol. 12842. Springer International Publishing, Cham, pp. 299–313, 2021. doi: 10.1007/978-3-030-86059-2_18.

Certificate Generation for Instances of Post Correspondence Problem

TPP 2024 Nov 25 2024

Keywords: Isabelle/HOL, Automata, Transition System, PCP

大森 章裕 Akihiro Omori (Science Tokyo)

Post Correspondence Problem

“You are given **s** types of tiles. Is it possible to arrange these tiles in such a way the top string matches exactly with the bottom one?” (**undecidable**)

Example:

100	0	1
1	100	00

“1311322” is solution:

100	1	100	100	1	0	0
1	00	1	1	00	100	100

Post Correspondence Problem

PCP[s, w]

- s types of tiles
- the length of the longest string on tiles = w

PCP[3,4]

1001	00	1
00	1	1001

0011	01	01
1	011	0

NOT PCP[3,4]

1001	00	1	10
00	1	1001	00

11101	00	1
00	100	100

This study aims to solve all problems in **PCP[3,4]**

PCP[3,4]

13,603,334 problems (normalized)

1110	01	1
1	11	011

1	0	1011
0	011	1

10	111	1
1	111	01

100	0	1
1	100	00

Not solved in [Zhao 2003]

3170 problems

1001	1	1
1	111	100

1010	110	01
10	01	1010

1110	101	1
01	1	1011

1111	1101	1
101	1	1111

1001	0	0
0	010	1

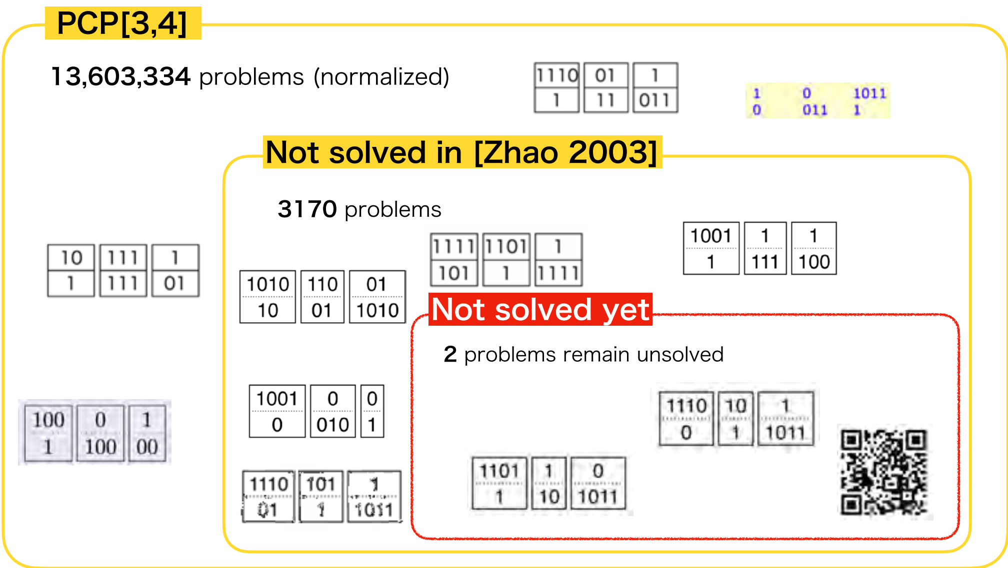
1110	10	1
0	1	1011

1101	1	0
1	10	1011

[1] Ling Zhao (2003): Tackling Post's Correspondence Problem. In: Computers and Games, Springer Berlin Heidelberg, pp. 326–344

Contribution

- **Solve all PCP[3,4] problems except 2**
- Propose an algorithm to show the unsolvability of PCP instances
- Algorithm for finding the shortest solution of PCP instances
  - Found the most difficult instance in PCP[3,4]
- Automatically generated proof of unsolvability for each instance



[1] Ling Zhao (2003): Tackling Post's Correspondence Problem. In: Computers and Games, Springer Berlin Heidelberg, pp. 326–344

Contribution

- Solve all PCP[3,4] problems except 2
- Propose an algorithm to show the Unsolvability of PCP instances
- Algorithm for finding the shortest solution of PCP instances
  - **Found the most difficult instance in PCP[3,4]**
- Automatically generated proof of Unsolvability for each instance

The most difficult instance (provisional)

1001	01	0
1	0	001

Length of the shortest solution: 452

```
[0, 1, 1, 0, 0, 1, 0, 1, 2, 0, 2, 0, 0, 1, 2, 0, 2, 2, 0, 1, 2, 0, 2, 0, 2, 0, 1, 1, 2, 2, 0, 2, 2, 0, 1, 2, 0, 1, 2, 0, 0, 1, 2, 1, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0, 2, 0, 2, 2, 1, 1, 2, 2,
0, 1, 1, 2, 2, 0, 1, 2, 2, 0, 1, 2, 1, 2, 0, 0, 1, 2, 1, 2, 0, 2, 1, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 0, 1, 2, 2, 2, 1, 2, 0, 1, 2, 2, 1, 2, 0, 2, 1, 1, 2, 2, 0, 2, 1, 2, 2, 0, 1, 0, 1,
2, 1, 2, 0, 1, 2, 1, 2, 0, 0, 1, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 1, 2, 2, 1, 2, 0, 1, 2, 2, 1, 2, 1, 2, 2, 0, 2, 1, 2, 2, 1, 2, 0, 1, 2, 1, 2, 2, 0, 2, 2, 1, 2, 0, 1, 1, 2,
2, 0, 0, 1, 2, 1, 2, 0, 2, 0, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 2, 1, 2, 2, 0, 1, 2, 1, 0, 1, 1, 0, 2, 0, 1, 2, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2, 0, 2, 0, 0, 1, 2, 0, 2, 2, 0, 2, 2, 2, 2,
0, 1, 2, 0, 2, 0, 2, 2, 2, 0, 1, 1, 2, 1, 1, 2, 0, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 1, 2, 0, 1, 2, 0, 1, 1, 2, 2, 0, 2, 1, 2, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 0, 1, 2, 1, 2, 0, 1, 2, 1, 2, 0,
1, 2, 2, 0, 1, 1, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 1, 2, 0, 0, 1, 2, 1, 2, 0, 1, 2, 2, 1, 2, 2, 1, 2, 0, 1, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 1, 2, 1, 2, 2, 0, 2, 1, 2, 0, 1, 2, 2,
1, 2, 0, 1, 2, 2, 1, 2, 0, 1, 2, 2, 0, 2, 1, 2, 2, 0, 2, 1, 2, 2, 0, 2, 1, 2, 0, 1, 2, 1, 2, 0, 1, 2, 1, 2, 0, 1, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 1, 2, 2, 1, 2, 2, 1, 2, 0, 1, 2, 1,
2, 1, 2, 2, 0, 2, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2]
```

- To show the shortestness, you have to examine that there is no solution for everything up to 451
  - Number of naive search nodes: estimated $10^{37.8}$
  - You need to do your best to mow the branches (details omitted)

Contribution

- Solve all PCP[3,4] problems except 2
- Propose an algorithm to show the Unsolvability of PCP instances
- Algorithm for finding the shortest solution of PCP instances
    - Found the most difficult instance in PCP[3,4]
- **Automatically generated proof of Unsolvability for each instance**

Reliability of Results

- I solved 3168 instances, but is the result reliable?
    - Are there any bugs in the program?

Possible Approaches

- Prove the program correctness
- **Create proofs that can be automatically verified for each (formal proof)**

Prove There Is a Solution

100	0	1
1	100	00

I'll try to prove that there is a solution for this

The actual solution 1,3,1,1,3,2,2 is an evidence: simple

```
definition ins::"pcp" where "ins ≡
[([C1,C0,C0],[C1]), ([C0],[C1,C0,C0]), ([C1],[C0,C0])]"

lemma solvable:
  shows "is_solution ins [0,2,0,0,2,1,1]"
  unfolding ins_def is_solution_def
  by simp
```

Prove There Is No Solution

0011	01	01
1	0101	0

```
definition pcp_instance::"pcp" where
  "pcp_instance ≡ [([C0,C0,C1,C1],[C1]), ([C0,C1],[C0,C1,C0,C1]), ([C0,C1],[C0])]"

theorem pcp_no_solution:
  "¬ have_solution pcp_instance.pcp_instance"
using this_is_invariant no_solution_if_exists_invariant by auto
```

TransitionSystem's Inductive Invariant is an evidence (described later)

For all the newly solved problems, such a certificate was automatically generated.

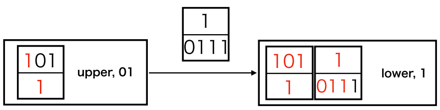
note: 2 of them haven't been checked yet due to its unreasonable execution time

Method

Transition System

Considering the transition “arranging tiles one by one, from left to right”, we can define a transition system.

- State Set $Q = \{\text{upper}, \text{lower}\} \times \Sigma^*$
- Transition Function $T : Q \to 2^Q$ $\cdots$ referring to arrange a tile



- Initial States $Init = T(\epsilon)$
- Bad States $Bad = \{(\text{upper}, \epsilon), (\text{lower}, \epsilon)\}$

Inductive Invariant

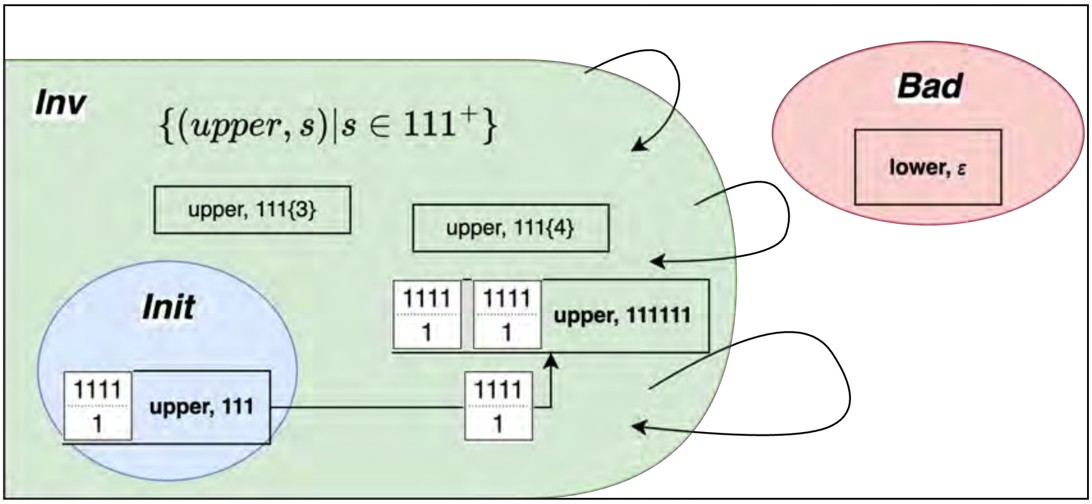
If there is some state set $Inv \subseteq Q$ satisfying following **3 conditions**

- $\forall s \in Inv\,.\,T(s) \subseteq Inv$ ⋯ closed under T
- $Init \subseteq Inv$
- $(\text{upper}, \epsilon) \notin Inv \wedge (\text{lower}, \epsilon) \notin Inv$

⋯ then this instance has no solution!

Example: 1111 / 1

Reachable States: $\{(upper, s) \mid s \in (111)^+\}$
We use regular language to represent state set



Finding Invariant

The prover doesn't care about how to find Invariant

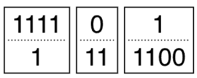
Many methods are proposed to solve this reachability problem such as PDR(Property Directed Reachability)[Bradley 11][Eén+ 11].

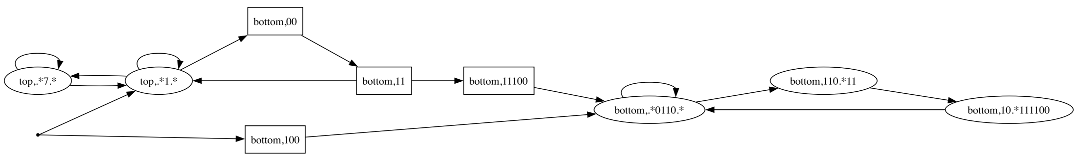
Here, we developed a new PCP-specific method (omitted)

Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In Verification, Model Checking, and Abstract Interpretation

Niklas Eén, Alan Mishchenko, and Robert K. Brayton. 2011. Efficient implementation of property directed reachability

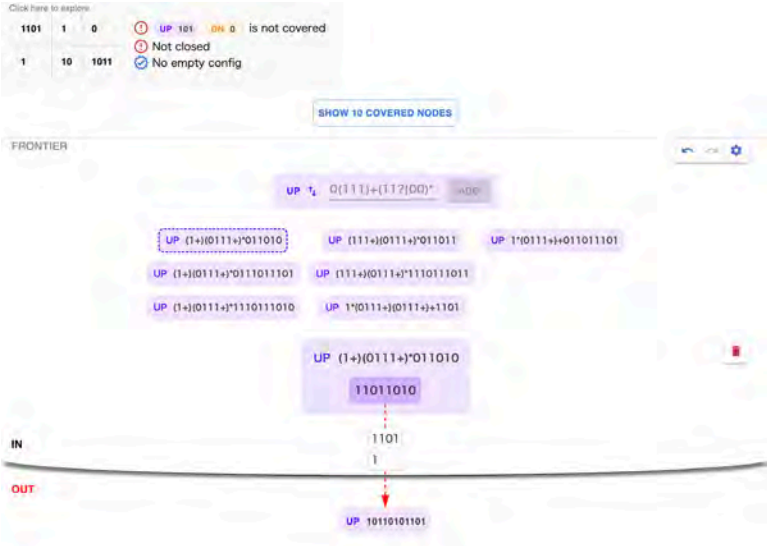
Output





- Set of state sets $V = \{V_1, V_2, \ldots, V_n\}$ and transition-inclusion relations
  - Example. $T(V_1) \subseteq V_2 \cup V_2 \cup V_3 \quad T(V_2) \subseteq V_1 \cup V_3 \quad T(V_3) \subseteq V_1 \cup V_3$
- Each node is represented as regular expression
  - $\cup V$ is an invariant ($T(\cup V) \subseteq \cup V$, etc⋯)

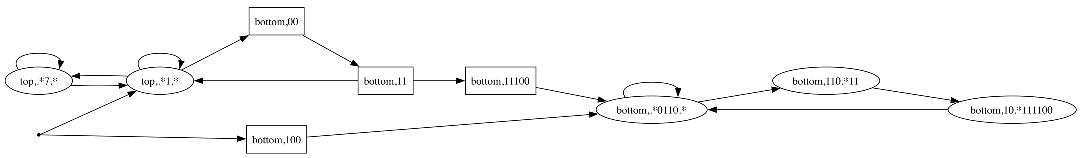
Manual Invariant Discovery Tool



Certificate Generation

Certificate Generation

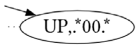
- We generated **Isabelle/HOL** code proving the correctness of Invariant and unsolvability for each specific PCP instance.



Given this graph…

- Define nodes: V_1, V_2, V_3
- Prove relations: $T(V_1) \subseteq V_2 \cup V_2 \cup V_3$ $\quad T(V_2) \subseteq V_1 \cup V_3$ $\quad T(V_3) \subseteq V_1 \cup V_3$
- Merge them and prove 3 conditions: $T(\cup V) \subseteq \cup V$ $\quad I \subseteq \cup V$ $\quad Bad \cap \cup V = \{\}$

- Define nodes



Tips: 状態/遷移関数の表し方は色々あるが, ビット列のパターンマッチでやると証明時のパフォーマンスが良かった

  - You first need to define NFA

datatype State = S_1 | S_2 | … | S_N requires O(N^2) lemmas

```
definition transition:: "alphabet ⇒ Num.num ⇒ Num.num list" where
"transition c s == case (c, s) of
(C1, (Num.Bit1 (Num.Bit1 Num.One))) => [(Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One)))]|
(C1, (Num.Bit1 (Num.Bit1 (Num.Bit1 Num.One)))) => [(Num.Bit0 (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One))))]
(C1, (Num.Bit0 (Num.Bit1 (Num.Bit1 Num.One)))) => [(Num.Bit1 (Num.Bit1 (Num.Bit1 Num.One)))]|
(C1, (Num.Bit0 Num.One)) => [(Num.Bit1 Num.One)]|
(C1, (Num.Bit1 (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One))))) => [(Num.Bit1 (Num.Bit1 Num.One)), (Num.Bit0
(C0, (Num.Bit1 (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One))))) => [(Num.Bit1 (Num.Bit1 Num.One)), (Num.Bit1
(C1, (Num.Bit1 Num.One)) => [(Num.Bit0 (Num.Bit0 Num.One))]|
(C1, (Num.Bit0 (Num.Bit0 Num.One))) => [(Num.Bit1 (Num.Bit0 Num.One))]|
(C1, (Num.Bit1 (Num.Bit1 (Num.Bit0 Num.One)))) => [(Num.Bit0 (Num.Bit0 (Num.Bit1 Num.One)))]|
(C1, (Num.Bit1 (Num.Bit0 (Num.Bit0 Num.One)))) => [(Num.Bit0 (Num.Bit1 (Num.Bit0 Num.One)))]|
(C1, (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One)))) => [(Num.Bit1 (Num.Bit0 (Num.Bit0 Num.One)))]|
(C1, Num.One) => [(Num.Bit0 Num.One)]|
(C1, (Num.Bit1 (Num.Bit0 Num.One))) => [(Num.Bit0 (Num.Bit1 Num.One))]|
(C1, (Num.Bit1 (Num.Bit0 (Num.Bit1 Num.One)))) => [(Num.Bit0 (Num.Bit1 (Num.Bit1 Num.One)))]|
(C1, (Num.Bit0 (Num.Bit1 (Num.Bit0 Num.One)))) => [(Num.Bit1 (Num.Bit1 (Num.Bit0 Num.One)))]|
(C1, (Num.Bit0 (Num.Bit1 Num.One))) => [(Num.Bit1 (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One)))), (Num.Bit1
(C1, (Num.Bit0 (Num.Bit0 (Num.Bit1 Num.One)))) => [(Num.Bit1 (Num.Bit0 (Num.Bit1 Num.One)))]|
( , ) => []
"

definition aut :: "(alphabet, Num.num) NA" where
"aut == NA [Num.One] transition [(Num.Bit0 (Num.Bit0 (Num.Bit0 (Num.Bit0 Num.One))))]"
```

- Define nodes

  - You first need to define NFA
  - A set of pairs of a direction and words is a node

```
datatype config = UP string | DN string

definition v_2:: "config set" where
  "v_2 ≡ PCPTrans.UP ` (lang nfa2.aut)"
```

- Prove inclusion relations
  - $T(V_1) \subseteq V_2 \cup V_2 \cup V_3$
  - Now, $V_1, V_2, \ldots$ were represented as NFAs
  - As we will check inclusion after V_1 is mapped by T, we would like to represent $T(V_1)$ as NFA as well. **How can we implement T ?**

Example. Consider applying [1001 / 00] to (UP, 001111010)

~~00~~1111010 [1001]
[00]

left-quotient and right-product operations on NFA !(basically)

Left-Quotient

```
fun pref_quotient::"('a, 's) NA ⇒ 'a list ⇒ ('a, 's) NA" where
  "pref_quotient A w = NA
    (steps_list A w (start_list A))
    (step A)
    (fin_list A)"
```

```
lemma pref_quotient_sanity_1:
  shows "{s | s. w@s ∈ lang A} ⊆ lang (pref_quotient A w)"
```

It is enough to show the inclusion of one side in this case

Right-Product

```
datatype 's AppendWordState = Orig 's | Appended nat

fun append_word::"('a, 's) NA ⇒ 'a list ⇒ ('a, 's AppendWordState) NA" where
  "append_word A [] = NA
    (map Orig (start_list A))
    (λ c. λ Orig s ⇒ map Orig (step A c s) | Appended n ⇒ [])
    (map Orig (fin_list A))
  " |
  "append_word A (w1#w) = NA
    (map Orig (start_list A))
    (λ c. λ Orig s ⇒ (let nexts = map Orig (step A c s) in
      if s ∈ fins A ∧ c = w1 then (Appended 0)#nexts
                             else nexts)|
            Appended n ⇒ (if n < length w ∧ c = w!n
                             then [Appended (n + 1)] else []))
    [Appended (length w)]
  "
```

If you try to do it with DFA, you need to double vertices every time you add a character (not scalable)

Transition Function

```
fun step_autconf_tile::"'s autconfig ⇒ PCP.tile ⇒
  ('s AppendWordState autconfig × (PCPTrans.config set))" where
  "step_autconf_tile (UP a) (up,dn) = (
    (UP (pref_quotient (append_word a up) dn)),
    PCPTrans.DN`((λ (s,p).  (drop (length up) p)) ` (Set.filter (λ(s,p
  )" |
```

- Now, let's check $T(V_1) \subseteq V_2 \cup V_2 \cup V_3$
- We implemented **antichain algorithm** (on-the-fly DFA construction and more)
  - **We didn't prove termination**
  - We defined step function and proved some invariant is hold when stepping.
  - The actual computation is done by transforming a subgoal.

state_invariant (state_1) ==> $T(V_1) \subseteq V_2 \cup V_2 \cup V_3$

use invariant holding lemma ↓

state_invariant (state_2) ==> $T(V_1) \subseteq V_2 \cup V_2 \cup V_3$

Finally

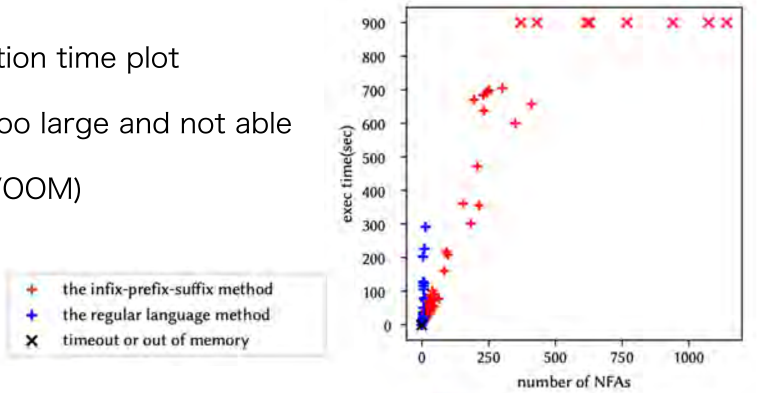
```
lemma no_solution_if_exists_invariant:
  fixes ts::pcp
  assumes "∃inv. is_invariant ts inv"
  shows "¬ have_solution ts"
proof (cases "have_solution ts")
```

```
theorem pcp_no_solution:
  "¬ have_solution pcp_instance.pcp_instance"
using this_is_invariant no_solution_if_exists_invariant by auto
```

Implementation Detail

- We generated Isabelle/HOL code using **templates** (like HTML template engine)
- `$ isabelle build` execution time plot
- some proofs are too large and not able

  to build (timeout/OOM)



Future Work

- Solve the remaining 2 problems in PCP[3,4]
- Formally prove results in [Zhao 2003]
- Improve scalability of proof execution
  - Some proofs require unreasonable time to execute "isabelle build"
- Formally prove the maximum length of shortest solution for each instance.
  - $\forall i \in PCP[3,4] \,.\, unsolvable(i) \lor len\_shortest\_sol(i) \leq 260$
  - If we proved this theorem on Isabelle/HOL, it supports we really solved all instances.

note: $unsolvable(i) \lor solvable(i)$ doesn't mean anything

Ling Zhao (2003): Tackling Post's Correspondence Problem. In: Computers and Games, Springer Berlin Heidelberg, pp. 326–344

重複を除去しつつ辞書順で最小のものを返す効率的なアルゴリズムのAgdaによる検証

法政大学大学院理工学研究科
城戸　道仁

Haskell のライブラリ関数　nubについて

・リストから重複を取り除く関数

例）nub “apple” = “aple”
nub [3,1,2,2,1] = [3,1,2]

・長さnのリストに対するnubの計算はΘ(n^2)ステップかかる

重複を除去しつつ辞書順で最小のものを返す関数

- Richard Bird(2010) は nubの定義を変更し、重複を除去しつつ辞書順で最小のものを返す関数へと変更したものを著書に示した。

変更前) nub “calculus” = “calus”

変更後) nub “calculus” = “aclus”

- この関数 nub は 計算量を**O(n logn)**ステップまで抑えることができるとBird は著書で述べており、証明も行なっている。

研究目的

Richard Bird(2010)は重複を除去しつつ辞書順で最小のものを計算量**O(n log n)**で返す関数nub のアルゴリズムを示した。

この計算量**O(n log n)**はリストの長さnに対して実行完了までに最悪の場合 **n log n** の計算ステップ数を必要とすることを示す。

このアルゴリズムの正当性を検証することを研究目的とする。

研究方法

Bird(2010)の示したアルゴリズムはHaskellによって示されている。

このアルゴリズムを定理証明支援系 Agda を用いて翻訳を行い、Agda上にて証明を行う。

翻訳について

- Bird(2010)の示したアルゴリズムが構成される過程で使用された関数の定義や導出された法則、命題をAgdaにて翻訳を行う。
- 翻訳の際、必要であれば補題を設け、補題の証明を行う。

nubの定義の運算

nub = minimum ・ longest ・ filter nodups ・ subseqs

nub　　　　　重複を除去しつつ辞書順で最小のものを返す関数

subseqs　与えられたリストのすべての部分列を計算

filter nodups 部分列のリストから重複を含まないもののみを取り出す

longest　最も長いものを計算(リストの長さ)

minimum　文字列順で最も小さいものを選ぶ

nubの定義の運算

nub = minimum ・ longest ・ filter nodups ・ subseqs
という仕様を元に運算から以下を導出できる

nub :
nub [] ≡ [] ×
∀ x xs → (x ∉ xs → nub (x ∷ xs) ≡ x ∷ nub xs) ×
(x ∈ xs → nub (x ∷ xs) ≡ (x ∷ nub (xs \\ [x])) min (nub xs))

xs \\ ys は リスト xs からリスト ysに含まれている要素を全て取り除いた
残りのリストである
A min B はAとBを比較して辞書順で小さい方を返す
しかし、これではAgdaでは停止性を認められない

関数 nub-helperの定義

```
nub-helper : List A → List A → List A
nub-helper [] _ = []
nub-helper (x ∷ xs) ys with x ∈? ys
... | yes _ = nub-helper xs ys
... | no _ with x ∈? xs
... | no _ = x ∷ nub-helper xs ys
... | yes _ = (x ∷ nub-helper xs (x ∷ ys)) min (nub-helper xs ys)

nub : List A → List A
nub xs = nub-helper xs []
```

nubの一般化に向けて

minの結合性と(x ∷) がminに対して分配できることを用いて、nubの一般化を行う。 nubの一般化をhubとするとhubは以下のように定義できる。

```
hub : List A → List A → List A
hub ws xs = minimum (map (λ is → is ++ nub (xs \\ is )) (inits ws))
```

また ws は厳密な増加順に並んでいるリストである。

hubを用いたnubの表記

nub-spec_3 :

nub [] ≡ [] ×

(∀ x y xs → (x < y → nub(x ∷ y ∷ xs) ≡

 hub (x ∷ y ∷ []) xs) ×

(¬(x < y) → nub(x ∷ y ∷ xs) ≡ hub [y] xs))

nubの一般化 hub に関して

hub : List A → List A → List A

hub ws xs = minimum (map (λ is → is ++

nub (xs \\ is)) (inits ws))

inits　リストのすべての先頭部分列を含むリストを返す

inits [] = [[]]

xs \\ [] = xs

nub xs = hub [] xs であるため、

hubはnubの一般化である

第２のnubの定義(Haskell)

```
nub            = hub []
hub ws []      = []
hub ws (x : xs) = case (x ∈ xs , x ∈ ws) of
                    (False , False) -> us ++ [x] ++
                                          hub[] (xs \\ us)
                    (False , True) -> us ++ [x] ++
                                          hub(tail vs) (xs \\ us)
                    (True , False) -> hub (us ++ [x]) xs
          (True , True)  -> hub ws xs
                    where (us , vs) = span (<x) ws
```

しかし、未だ全体の計算量は2乗オーダー

x ∈ xs , x ∉ wsにおける課題点

minimum (map (λ is → is ++ nub ((x ∷ xs) \\ is))
(inits us))

≡⟨ {! !} ⟩ -- x ∈ xs かつ x ∉ us だからnubの再帰的定義により
minimum (map (λ is →
is ++ (([x] ++ nub (xs \\ (is ++ [x]))) min
nub (xs \\ is))) (inits us))
isを複数回含むため、別に補題を設ける必要がある

x ∈ xs , x ∈ wsにおける課題点

minimum (map (λ is → is ++ nub (xs \\ is)) (inits (us ++ [x]))) min ((us ++ [x]) ++ minimum (map(λ is → is ++ nub(xs \\ (us ++ [x] ++ is))) (inits vs’)))

≡⟨ {! !} ⟩ -- initsとminの関係を用いて証明

minimum (map(λ is → is ++ nub(xs \\ is)) (inits (us ++ [x] ++ vs')))

今後の課題

- 証明できていない補題の着手
- 第２のnubの定義から最終版のnubの定義への論証の着手
→ 集合の導入が必要

参考文献

- Bird Richard 2010, 山下伸夫 2014 関数プログラミング珠玉のアルゴリズムデザイン(原題:Pearls of Functional Algorithm Design) Cambridge University Press.

Complete graphs and independence numbers

Kazunori Matsuda[1]   Takafumi Saikawa[2]   Yosuke Tsuji[2]

[1]Kitami Institute of Technology

[2]Nagoya University

TPP 2024

Graph

A graph is:

- V : type of vertices
- E : type of edges
- d : mapping from edges to sets of vertices
- axiom : $|d(x)| = 2$

Graph

```
(* undirected,loopless graph *)
Module LooplessUndirectedGraph.
Section def.
Record t := mk {
  V : finType;
  E : finType;
  boundary : E -> {fset V};
  _ : forall e : E, #|` boundary e | = 2;
}.
End def.
Module Exports.
Notation llugraph := t.
Notation "`d" := (boundary).
Notation "`E" := (E).
Notation "`V" := (V).
End Exports.
End LooplessUndirectedGraph.
```

3 / 18

Matching

For a graph (V, E, d) and $S \subset E$, S is a

matching	if no two edges share a vertex, and an
induced matching	if furthermore no two edges are connected by an edge

```
Definition is_matching :=
 [∀ e in S, [∀ f in S, (e != f) ==> [`d(e) ⊥ `d(f)]]].

Definition is_induced_matching :=
 [∀ e in S, [∀ f in S,
   (e != f) ==> [∀ g, [`d(e) ⊥ `d(g)] || [`d(f) ⊥ `d(g)]]]].

(* [ X ⊥ Y ] = X and Y are disjoint *)
```

4 / 18

Matching numbers

matching number Maximum size (cardinality) of a matching

induced matching number Maximum size of an induced matching

minimal matching number Minimum size of a maximal matching

```
Definition nmatch := \max_(S∈matching) #|` S |.
Definition nindmatch := \max_(S∈induced matching) #|` S |.

Definition is_maximal_matching :=
  (S ∈ matching) &&
  [∀ T : {fset `E}, (S ⊂ T) ==> (T ∉ matching)].
Definition nminmatch :=
  \big[min/nmaxmatch]_(S∈maximal matching) #|` S |.
```


Complete graphs

A graph (V, E, d) is complete if:

- d is injective (i.e., the graph is simple), and
- any two distinct vertices are connected by an edge

```
Definition is_complete_graph :=
  injective `d ∧
    ∀ v w : `V, v != w -> ∃ e : `E, `d(e) = {v,w}.
```

Number of edges

For a complete graph (V, E, d),

$$2|E| = |V|(|V| - 1)$$

```
Lemma card_Ecomplete :
  is_complete_graph -> (2 * #| `E | = #| `V | * (#| `V | - 1)).
```


Proof sketch for card_Ecomplete

Counting ones in the upper half of the adjacency matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

suffices, but..

Proof sketch for `card_Ecomplete`

Counting ones in the upper half of the adjacency matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

suffices, but..

“upper” requires an ordering among vertices

8 / 18

Proof sketch for `card_Ecomplete`

Counting ones in the upper half of the adjacency matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

suffices, but..

“upper” requires an ordering among vertices

NOT COMFORTABLE

8 / 18

Proof sketch for card_Ecomplete

Do it symmetrically:

$$\texttt{(edge } e \texttt{ with boundary } \{a, b\}\texttt{)} \longmapsto \{(a, b), (b, a)\}$$

```
Definition biboundary (e : `E) :=
  let: exist (a, b) _ := cardfs2_sig (boundary_card2 e) in
  {(a, b), (b, a)}.

Lemma card_Ecomplete_aux :
  is_complete_graph ->
  ⋃{biboundary e | e ∈ `E} =
    (fsetT `V × fsetT `V) ∖ fset_diag (fsetT `V).
```


Proof sketch for card_Ecomplete

Do it symmetrically:

$$\texttt{(edge } e \texttt{ with boundary } \{a, b\}\texttt{)} \longmapsto \{(a, b), (b, a)\}$$

```
Definition biboundary (e : `E) :=
  let: exist (a, b) _ := cardfs2_sig (boundary_card2 e) in
  {(a, b), (b, a)}.

Lemma card_Ecomplete_aux :
  is_complete_graph ->
  ⋃{biboundary e | e ∈ `E} =
    (fsetT `V × fsetT `V) ∖ fset_diag (fsetT `V).
```


Lots of fun with the mathcomp-finmap library

In our mathcomp_extra.v thing

```
Section finmap_ext.
Local Open Scope fset_scope.

Lemma fproperU1 [K : choiceType] (x : K) (A : {fset K}) :
  x \notin A -> A ⊊ x |` A.

Lemma fdisjointXX [K : choiceType] (A : {fset K}) (x : K) :
  x \in A -> ~ [A ⊥ A]%fset.

Lemma fdisjointXXP [K : choiceType] (A : {fset K}) :
  reflect (A = ∅) [A ⊥ A]%fset.

(* ... 300 lines ... *)

End finmap_ext.
```


Matching numbers of complete graphs

Matching numbers are determined for complete graphs:

- induced matching number is 1 (unless there are no edges)
- all maximal matchings have the same size $|V|/2$

```
Lemma nindmatch_complete :
  is_complete_graph ->
  nindmatch <= 1 (equality holds iff 2 <= #| `V |).

Lemma maximal_matching_complete (S : {fset `E}) :
  is_complete_graph ->
  S ∈ maximal_matching -> #|` S | = #| `V | ./2.
```

Proof sketch for nindmatch_complete

```
Lemma nindmatch_complete :
  is_complete_graph ->
  nindmatch <= 1 (equality holds iff 2 <= #| `V |).
```


Proof sketch for nindmatch_complete

```
Lemma nindmatch_complete :
  is_complete_graph ->
  nindmatch <= 1 (equality holds iff 2 <= #| `V |).
```

Towards a contradiction, assume ($1 <$ nindmatch). Then there are two edges in an induced matching that are however connected due to the completeness.

34 lines of code

Proof sketch for maximal_matching_complete

```
Lemma maximal_matching_complete (S : {fset `E}) :
  is_complete_graph ->
  S ∈ maximal_matching -> #|` S | = #| `V | ./2.
```


Proof sketch for maximal_matching_complete

```
Lemma maximal_matching_complete (S : {fset `E}) :
  is_complete_graph ->
  S ∈ maximal_matching -> #|` S | = #| `V | ./2.
```


Proof by contraposition. The case $(|V|/2 < |S|)$ is easy (7 lines (using other lemmas)). If $(|V|/2 > |S|)$, we can take two vertices from the complement of those in S. They are connected by the completeness and can be added to a matching, a contradiction to the maximality.

36 lines of code

Independent set (of vertices)

For a graph (V, E, d) and $S \subset V$, S is said to be an independent set if it does not contain both of the boundary vertices of any edge.

```
Definition is_independent_set :=
  [forall e : `E, ~~ (`d(e) ⊂ S)].
Definition independent_set :=
  [fset S : {fset `V} | is_independent_set S].

(* independence number;
   often denoted by α in the literature *)
Definition nindep := \max_(S∈independent_set) #|` S |.
```


Application: formalizing lemmas from [Hirano-Matsuda https://arxiv.org/abs/2001.10704]

```
Lemma nindmatch_leq_nindep (G : llugraph) :
  nindmatch G <= nindep G.

Lemma nmatch_minmatch_leq_nindep G :
  (nmatch G - nminmatch G).*2 <= nindep G.
```

Application: formalizing lemmas from [Hirano-Matsuda https://arxiv.org/abs/2001.10704]

```
Lemma nindmatch_leq_nindep (G : llugraph) :
  nindmatch G <= nindep G.

Lemma nmatch_minmatch_leq_nindep G :
  (nmatch G - nminmatch G).*2 <= nindep G.
```

- Pen-paper proof: 4+4 lines
- Coq proof: 6+7 lines

Conclusion

- MathComp-Algebra, -Finmap, and -Classical together work as a practical playground for graph theory
- Tension between concrete and axiomatic definitions
- TODO:
  - more graph invariants
  - connection to ring theory
  - use Hierarchy-Builder
  - bridge to Coq-Graph

Appendix: generating complete graphs

Complete graphs can be generated from given vertices

- V : type of vertices
- E = all unordered pairs of two distinct vertices
- $d(x)$ = type coercion
- the axiom $|d(x)| = 2$ is immediate

Generating complete graphs

```
Module CompleteGraph.
Section def.
Variable V : finType.
Definition E :=
 [fset {p₁,p₂} | p ∈ (fsetT V × fsetT V) ∖ fset_diag (fsetT V)].
Definition boundary (e : E) : {fset V} := val e.
Definition axiom : forall e : E, #|` `d(e) | = 2.
Proof. (* .. *) Qed.
Definition t := LooplessUndirectedGraph.mk axiom.
End def.
End CompleteGraph.

Notation "`K" := (CompleteGraph.t).
Notation "`K_ n" := (CompleteGraph.t 'I_n).
```

λ計算の代数と幾何
(Algebra and Geometry of the λ-calculus)

佐藤雅彦
京都大学情報学研究科

中野圭介
東北大学電気通信研究所

TPP 2024
九州大学
2024年11月26日

Summary

- この講演では，型のないλ計算の形式体系 $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}^*$, $\mathbf{V}$ を扱う．
- これらの体系のいずれも型体系として定式化する．
- これらの型について (α), (β), (ξ) の規則の扱い方の相違を中心に議論する．
- $\boldsymbol{\lambda}^*$ から $\mathbf{V}$を構築するための補助の型 $\boldsymbol{\Lambda}$ を用意する．
- Each expression in $\mathbf{V}$ is represented using the idea of de Bruijn level. Unlike de Bruijn indices, de Bruijn levels are counted from the root of a $\boldsymbol{\lambda}$-exp (represented as a tree).
- The most important idea in this talk, to regard sequece of variables as contexts in which expressions are evaluated.
- 今回は (η) は扱わない．

Summary (λ, λ^*, $\mathbf{V}$)

- 本講演で扱うλ計算の体系はすべて，自由変数，束縛変数ともに明示的な名前をもつ体系である．
- λはチャーチの体系をカリーが定式化したもの．
- 他は本講演で提案する体系
- λ はα同値という複雑な概念が必要．(β) は代入により計算される．変数の束縛はひとつの変数について行われる．
- λ^* は変数の束縛を変数の列について行う．これにより，自由モノイド $\mathbb{Z}^*$ が λ^* に左から作用する．λで 3 つ (var, app, abs) あった構成規則のうち abs が不要になる．
- $\mathbf{V}$ は λ^* の中からよい性質を持つものを取り出したものと見なせる．$\mathbf{V}$ には環 $\mathbb{Z}$ の加法群が右から作用する．$\mathbf{V}$ では (β) は代入でなく関数適用 (apply) として定義され，$\mathbf{V}$ 上の二項演算となる．また，(α) は不要になる．

Summary (Λ)

Λ は λ^* の中からよい性質を持つものを取り出し，$\mathbf{V}$を構築するために用意する型であるが，その位置付けは以下のように示すことができる．

$$\lambda \xrightarrow{\cong} \lambda^* \xrightarrow{\alpha} \mathbf{V} \xrightarrow{\text{incl}} \Lambda$$

ここで，incl は inclusion map である．

αは，各 $M:\lambda$ に対して，それを含む α-同値類 (equivalence class) の中から 0 を始点とする canonical な代表元 (representative) を返す関数である．

Λは，それ自身は計算体系ではないが，3 つのλ計算の体系λ, λ^*, $\mathbf{V}$を結びつける重要な場 (field) となっている．

Some notations

- 整数の型 $\mathbb{Z}$ の要素を i, j, k, x, y, z で表記する．x, y, z は整数を変数記号として扱うときに用いる．0 以上の整数の全体の型を $\mathbb{N}$ で表し，その要素を m, n で表記する．
- 整数列の型 $\mathbb{Z}^*$ の要素を u, v で表記する．$\mathbb{Z}^*$ の具体的な要素は $(1\ 0\ \ -2)$ のように表記する．また u の長さを $|u|$ で表す．例えば $|(3\ 1)| = 2$.

λ and their computation rules

型 λ を以下のように定義する

$$\frac{x : \mathbb{Z}}{x : \lambda}\ \text{var} \qquad \frac{K : \lambda \quad L : \lambda}{(K\ L) : \lambda}\ \text{app} \qquad \frac{x : \mathbb{Z} \quad K : \lambda}{\lambda x.K : \lambda}\ \text{abs}$$

λにおける計算は以下の規則を用いて行われる．

(e1)	$\Rightarrow K = K$
(e2)	$K = L \Rightarrow L = K$
(e3)	$K = K', K' = K'' \Rightarrow K = K''$
(c)	$K = K', L = L' \Rightarrow (K\ L) = (K'\ L')$
(ξ)	$K = L \Rightarrow \lambda x.K = \lambda x.L$
(α)	$K \equiv_\alpha L$ (α-equiv.) $\Rightarrow K = L$
(β)	$K[x := K'] \equiv L \Rightarrow (\lambda x.K\ K') = L$

The free monoid $\mathbb{Z}^*$ over $\mathbb{Z}$

モノイド $\mathbb{Z}^*$ とその上の**積演算**を以下のように定義する．以下では，u と v の積を $\boldsymbol{u \cdot v}$ と表記する．(1) と (2) で $\mathbb{Z}^*$ の要素を定義し，(3) と (4) で，$\mathbb{Z}^*$ の上の積演算を定義する．

$$\Rightarrow () : \mathbb{Z}^* \quad (1)$$

$$x : \mathbb{Z}, v : \mathbb{Z}^* \Rightarrow xv : \mathbb{Z}^* \quad (2)$$

$$() \cdot v := v \quad (3)$$

$$(xu) \cdot v := x(u \cdot v) \quad (4)$$

この定義により，$\mathbb{Z}^*$ が () を単位元とするモノイドになることがたしかめられる．

$\boldsymbol{\lambda}^*$ and $\mathbf{V}$

$\boldsymbol{\lambda}$計算を自然に展開できる空間としての型 $\mathbf{V}$ を定義したい．

そのため，まず，従来の$\boldsymbol{\lambda}$計算の場である $\boldsymbol{\lambda}$ と同型であるが，$\mathbb{Z}^*$ が自然に作用する**型体系 $\boldsymbol{\lambda}^*$** を定める．この結果 $\boldsymbol{\lambda}$ にあった abs の規則が不要になる．

次に最終目標である**型体系 $\mathbf{V}$** と $\alpha : \boldsymbol{\lambda}^* \to \mathbf{V}$ を

$$M \equiv_\alpha N \text{ in } \boldsymbol{\lambda}^* \Leftrightarrow \alpha(M) = \alpha(N) \text{ in } \mathbf{V}$$

が成り立つように定義する．これにより，$\mathbf{V}$では (α) の計算規則が不要になる．(ξ) も不要になるが，その理由は後に示す．

λ^* and context

λ^*

$$\frac{u : \mathbb{Z}^* \quad x : \mathbb{Z}}{ux : \lambda^*}\ \text{var} \qquad \frac{u : \mathbb{Z}^* \quad M : \lambda^* \quad P : \lambda^*}{u(M\ P) : \lambda^*}\ \text{app}$$

上の u のことを**文脈** (context) という．ux は**文脈 u における**変数 (variable)x を表し，$u(M\ P)$ は**文脈 u における**関数 M の P への適用 (application) を表している．

λ^*-exp M の**長さ** (length) $|M|$ を以下で定める．

$$|ux| := |u|$$
$$|u(M\ P)| := |u|$$

Comparison of λ and λ^*

λ

$$\frac{x : \mathbb{Z}}{x : \lambda}\ \text{var} \qquad \frac{K : \lambda \quad L : \lambda}{(K\ L) : \lambda}\ \text{app} \qquad \frac{x : \mathbb{Z} \quad K : \lambda}{\lambda x.K : \lambda}\ \text{abs}$$

λ^*

$$\frac{u : \mathbb{Z}^* \quad x : \mathbb{Z}}{ux : \lambda^*}\ \text{var} \qquad \frac{u : \mathbb{Z}^* \quad M : \lambda^* \quad P : \lambda^*}{u(M\ P) : \lambda^*}\ \text{app}$$

λの定義にあった abs の規則が λ^* ではなくなっている! この現象には，λでは一度にひとつの変数しか束縛できないのに対して，λ^* では (0 の場合を含めて) 任意個数の変数を束縛できることが関係している．

Action of $\mathbb{Z}^*$ on λ^*

$\mathbb{Z}^*$ は次のようにして λ^* へ作用する．以下 u の λ^* への作用を $u \cdot M$ と書く．

$$\begin{array}{lrcl} \text{(var)} & u \cdot (vx) & := & (uv)x \\ \text{(app)} & u \cdot (v(M\ P)) & := & (uv)(M\ P) \end{array}$$

この作用を用いて，次にλから λ^* への埋め込み (embedding) を定義する．

Embedding of λ into λ^*

λ-exp K を λ^*-exp $M = K^*$ に変換する関数

$$(\ \_\ )^* : \lambda \to \lambda^*$$

を次のように定める．

$$\begin{array}{lrl} \text{(var)} & (x)^* := & ()x \\ \text{(app)} & ((K\ L))^* := & ()((K)^*\ (L)^*) \\ \text{(abs)} & (\lambda x.K)^* := & (x) \cdot (K)^* \end{array}$$

例えば，

$$\begin{aligned} & (\lambda x.\lambda y.\lambda z.((x\ z)\ (y\ z)))^* \\ =& (x) \cdot (y) \cdot (z) \cdot (((x\ z)\ (y\ z)))^* \\ =& (x) \cdot (y) \cdot (z) \cdot ()(()(()x\ ()z)\ ()(()y\ ()z)) \\ =& (x) \cdot (y) \cdot (z)(()(()x\ ()z)\ ()(()y\ ()z)) \\ =& (x) \cdot (y\ z)(()(()x\ ()z)\ ()(()y\ ()z)) \\ =& (x\ y\ z)(()(()x\ ()z)\ ()(()y\ ()z)) \end{aligned}$$

Embedding of λ^* into λ

λ^*-exp M をλ-exp $K = M_*$ に変換する関数

$$(\_)_* : \lambda^* \to \lambda$$

を次のように定める．

$$
\begin{array}{lrcl}
\text{(var1)} & (()y)_* & := & y \\
\text{(var2)} & ((xu)y)_* & := & \lambda x.(uy)_* \\
\text{(app1)} & (()(M\ N))_* & := & ((M)_*\ (N)_*) \\
\text{(app2)} & ((xu)(M\ N))_* & := & \lambda x.(u(M\ N))_*
\end{array}
$$

$(\_)^*$ と $(\_)_*$ は互いに相手の逆関数 (inverse) になる．

Action of $\mathbb{Z}$ on $\mathbb{Z}^*$ and λ^*

X が整数列 (integer sequence) または λ^*-exp のとき，整数 k の X への右からの作用 X^k を以下で定義する．この作用は対象

X を k だけずらす (shift X by k)

作用であると考えられる．$\mathbb{Z}$ を実数直線の上の目盛 (integer scale) として見るとき，k が正のときは k だけ右に移動し，負のときは左，0 のときは元のままになる．

$$
\begin{aligned}
()^k &:= () \\
(xu)^k &:= (x+k)u^k
\end{aligned}
$$

$$
\begin{aligned}
(ux)^k &:= u^k(x+k) \\
(u(M\ P))^k &:= u^k(M^k\ P^k)
\end{aligned}
$$

これは $\mathbb{Z}$ の加法群 (additive group) が作用するものなので，$-k$ を作用させると元に戻る．

Λ

最終目標である型 $\mathbf{V}$ を定義する前にまず，それを含む型 $\mathbf{\Lambda}$ を以下のように定義する．

$$\frac{i:\mathbb{Z} \quad n:\mathbb{N} \quad x:\mathbb{Z}}{i^n x:\mathbf{\Lambda}}\ \mathsf{var} \qquad \frac{i:\mathbb{Z} \quad n:\mathbb{N} \quad M:\mathbf{\Lambda} \quad P:\mathbf{\Lambda}}{i^n(M\ P):\mathbf{\Lambda}}\ \mathsf{app}$$

ここで，i^n は直観的には, $i \leq j < i+n$ となる j 達からなる区間 (interval) を表している．

比較のため $\boldsymbol{\lambda}^*$ の定義を再掲する．

$$\frac{u:\mathbb{Z}^* \quad x:\mathbb{Z}}{ux:\boldsymbol{\lambda}^*}\ \mathsf{var} \qquad \frac{u:\mathbb{Z}^* \quad M:\boldsymbol{\lambda}^* \quad P:\boldsymbol{\lambda}^*}{u(M\ P):\boldsymbol{\lambda}^*}\ \mathsf{app}$$

両者の違いをよく見てほしい．

Start point, length and context of Λ-exp

すべての $\mathbf{\Lambda}$-exp M は，

$$i^n x$$

または

$$i^n(N\ P)$$

のいずれかの形をしている．

- i を M の始点 (start point) と呼び，$\underline{M}$ で表す．
- n を M の長さ (length) と呼び，$|M|$ で表す．
- i^n を M の文脈 (context) と呼ぶ．

$\mathbf{V}$

型 Λ の定義は以下のようであった．

$$\frac{}{i^n x : \Lambda}\ \text{var} \qquad \frac{M : \Lambda \quad P : \Lambda}{i^n (M\ P) : \Lambda}\ \text{app}$$

型 $\mathbf{V}$ をもつ exp を，条件付 Λ-exp として以下のように定義する．

$$\frac{x < i + n}{i^n x : \mathbf{V}}\ \text{var} \qquad \frac{M : \mathbf{V} \quad P : \mathbf{V} \quad \underline{M} = \underline{P} = i + n}{i^n (M\ P) : \mathbf{V}}\ \text{app}$$

規則 (var) で得られる $i^n x : \mathbf{V}$ は, $x < i$ のとき**自由変数** (free variable) であり，$i \le x$ のとき**束縛変数** (bound variable) である．

Geometric structure of $\mathbf{V}$

Λ-exp を始点ごとに分類し，

$$\mathbf{V}_i := \{M : \mathbf{V} \mid \underline{M} = i\}$$
$$(\mathbf{V}_i)^k := \{M^k : \mathbf{V} \mid M : \mathbf{V}_i\}$$

と定めると，

$$\mathbf{V}_{i+k} = (\mathbf{V}_i)^k$$

が成り立つ．

ただし，上の M^k は以下のように定める．

$$(i^n x)^k := (i+k)^n (x+k)$$
$$(i^n (N\ P))^k := (i+k)^n (N^k\ P^k)$$

これは $\mathbf{V}$-exp を整数 k だけずらす作用である．

Concatenation operation on $\mathbf{V}$

整数 i, n $(n \geq 0)$ と $M : \mathbf{V}_{i+n}$ が与えられたとき，

$$i^n \odot M : \mathbf{V}_i$$

を以下で定義する．

$$i^n \odot (i+n)^m x := i^{n+m} x$$
$$i^n \odot (i+n)^m (N\ P) := i^{n+m} (N\ P)$$

これは，$\mathbf{V}_{i+n}$-exp の前に i^n を連結 (concatenate) し，$\mathbf{V}_i$-exp を得る操作と見ることができる．

Function application on $\mathbf{V}$

$\mathbf{V}$-exp $i^n(M\ P)$ は $|M| > 0$ であるとき β-redex となる．このとき $x = i + n$ とすれば，M, P ともに $\mathbf{V}_x$-exp である．そこで，$\mathbf{V}$-exp における 関数適用 (function application) —@— を

$$(x^{n+1} y)@P := \mathsf{sub}(P, x, 0, (x+1)^n y)$$
$$(x^{n+1}(M\ N))@P := \mathsf{sub}(P, x, 0, (x+1)^n (M\ N))$$

と定義する．ここで用いた補助関数 sub の定義は次のスライドで示すが，$\mathsf{sub}(P, x, 0, M)$ は M 中の x に P を代入する関数であり，Curry の記法では $[P/x]M$, Barendregt の記法では $M[x := P]$ に相当する．

Substitution on V

代入関数 sub を以下のように定義する．

$$\mathsf{sub}(P, x, n, j^m y) := \begin{cases} (j-1)^m y & (y < x \text{ のとき}) \\ (j-1)^m \odot P_x^{[n+m]} & (y = x \text{ のとき}) \\ (j-1)^m (y-1) & (y > x \text{ のとき}) \end{cases}$$

$$\mathsf{sub}(P, x, n, j^m (M\ N)) := (j-1)^m (\mathsf{sub}(P, x, n+m, M)\ \mathsf{sub}(P, x, n+m, N))$$

上で用いた記法 $P_x^{[n]}$ は以下のように定める．

$$(i^m y)_x^{[n]} := \begin{cases} (i+n)^m y & (y < x \text{ のとき}) \\ (i+n)^m (y+n) & (x \leq y \text{ のとき}) \end{cases}$$

$$(i^m (P\ Q))_x^{[n]} := (i+n)^m (P_x^{[n]}\ Q_x^{[n]})$$

Example of function application on V

λ-exp $\lambda xyz.xz(yz)$ と $\lambda xy.x$ は V-exp としてそれぞれ

$$S := 0^3(3^0(3^0 0\ 3^0 2)\ 3^0(3^0 1\ 3^0 2)) \qquad K := 0^2 0$$

と表せる．このとき，$S@K$ は以下のように計算される．

$$\begin{aligned} S@K &= \mathsf{sub}(K, 0, 0, 1^2(3^0(3^0 0\ 3^0 2)\ 3^0(3^0 1\ 3^0 2))) \\ &= 0^2(2^0(\mathsf{sub}(K, 0, 2, 3^0 0)\ \mathsf{sub}(K, 0, 2, 3^0 2)) \\ &\qquad 2^0(\mathsf{sub}(K, 0, 2, 3^0 1)\ \mathsf{sub}(K, 0, 2, 3^0 2))) \\ &= 0^2(2^0(2^0 \odot K_0^{[2]}\ 2^0 1)\ 2^0(2^0 0\ 2^0 1)) \\ &= 0^2(2^0(2^2 2\ 2^0 1)\ 2^0(2^0 0\ 2^0 1)) \end{aligned}$$

となり，これは λ-exp $\lambda xy.((\lambda zw.z)y)(xy)$ に対応する．

Computation rules of $\mathbf{V}$-exp

$\mathbf{V}$-exp の計算は以下の規則を用いて行われる．α, ξは不要.

$$
\begin{array}{ll}
\text{(e1)} & M = M \\
\text{(e2)} & M = N \Rightarrow N = M \\
\text{(e3)} & M = N, N = P \Rightarrow M = P \\
\text{(c)} & M = M', N = N' \Rightarrow i^n(M\ N) = i^n(M'\ N') \\
(\beta) & |M| > 0 \Rightarrow i^n(M\ P) = i^n \odot (M@P)
\end{array}
$$

上の規則の下で $M = N$ となるとき，$\underline{M} = \underline{N}$ となる．

ξが不要なのは，(c) と (β) により，(ξ) を使うことなく，(β)-redex に対して関数適用ができるからである．

αが不要な理由は次のスライドで示す．

Relationship between λ^* and $\mathbf{V}$

λ^*-exp M, N に対し，

$$M \equiv_\alpha N \iff \alpha_i(M) = \alpha_i(N)$$

となる関数

$$\alpha_i : \lambda^* \to \mathbf{V}_i$$

を Coq で定義した．このとき，α_i は各 $M : \lambda^*$ に対して，M の属する α 同値類 (α-equiv. class) の中から i を始点とする canonical な代表元 (representative) を与える．また，

$$(\alpha_i(M))^j = \alpha_{i+j}(M)$$

となる．

Conclusion

By imposing the convention that (1) the free variables of a $\boldsymbol{\lambda}$-exp must be negative and (2) the bound variables of a $\boldsymbol{\lambda}$-exp must be nonnegative, we have the following diagram. In the diagram below, inj is a naturally defined injection.

$$\boldsymbol{\lambda} \xrightarrow{(-)^*} \boldsymbol{\lambda}^* \xrightarrow{\alpha_0} \mathbf{V} \xrightarrow{\text{incl}} \boldsymbol{\Lambda} \xrightarrow{\text{inj}} \boldsymbol{\lambda}^* \xrightarrow{(-)_*} \boldsymbol{\lambda}$$

Writing $\boldsymbol{\alpha}$ for the composition of the 5 maps above, we have $\boldsymbol{\alpha} : \boldsymbol{\lambda} \to \boldsymbol{\lambda}$. Here, given $M : \boldsymbol{\lambda}$, $\alpha(M)$ enjoys the property that $M \equiv_\alpha \alpha(M)$.

We have implemented the above diagram in Coq.

¬ AC の相対的無矛盾性証明の Isabelle/ZF による形式化

東北大学 大学院情報科学研究科
住井・松田研究室 M2
舟根大喜

November 22, 2024

概要

やったこと

¬AC の ZF 上の相対的無矛盾性証明を Isabelle/ZF で形式化

動機

- 公理的集合論の重要な技法である強制法 (後述) を用いた議論の形式化は限られている
- AC の相対的無矛盾性証明は形式化されており合わせて AC の独立性が形式化されたことになる

1/20

用語

- **公理的集合論**：
  集合がみたすべき性質を公理として定めた集合論

- **ZF**：Zermelo-Fraenkel 公理系。一階述語論理上で形式化されたよく採用される公理系

- **AC**：選択公理 (Axiom of Choice)
  任意の非空集合の族からそれぞれ 1 つの元を選ぶ関数が存在するという公理
  - 整列可能定理、Zorn の補題など同値な重要な命題がある

2/20

用語

- *Extensionality:* $\forall x \forall y (\forall z (z \in x \leftrightarrow z \in y) \to x = y)$
- *Pairing:* $\forall x \forall y \exists z \forall w (w \in z \leftrightarrow w = x \lor w = y)$
- *Union:* $\forall x \exists y \forall z (z \in y \leftrightarrow \exists w (z \in w \land w \in x))$
- *Power set:* $\forall x \exists y \forall z (z \in y \leftrightarrow z \subseteq x)$
- *Infinity:* $\exists x (\emptyset \in x \land \forall y (y \in x \to y \cup \{y\} \in x))$
- *Regularity:* $\forall x (x \neq \emptyset \to \exists y (y \in x \land y \cap x = \emptyset))$
- *Infinity:* $\forall x (x \neq \emptyset \to \exists y (y \in x \land \forall z (z \in x \to z \notin y)))$
- *Separation:* $\forall p \forall x \exists y \forall z (z \in y \leftrightarrow z \in x \land \phi(z, p))$
- *Replacement:* $\forall p (\forall x \forall y \forall z (\phi(x, y, p) \land \phi(x, z, p) \to y = z) \to \forall X \exists Y \forall y (y \in Y \leftrightarrow \exists x (x \in X \land \phi(x, y, p))))$
- *Choice:* $\forall x \exists f$("f *is a function on* x" $\land \forall y (y \in x \to f(y) \in y))$

3/20

用語

- 命題 φ が公理系 T 上で**相対的無矛盾**：
  T が無矛盾ならば公理系 $T+\varphi$ も無矛盾なこと
- φ が T から**独立**：
  T から φ も $\neg\varphi$ も証明できないこと

▶ φ と $\neg\varphi$ の T 上の相対的無矛盾性を示すことで
φ が T から独立であることが証明できる
(T が無矛盾と仮定すれば)

4/20

用語

- **強制法**：
  集合論のモデルを拡張し新しいモデルを作る技法
  - M を強制法で拡張したモデルを
    M の **generic extension** と呼び $M[G]$ と書く
  - $M[G]$ は poset $\mathbb{P}$ と generic filter G に依存しており、$\mathbb{P}$ をうまく選ぶことで、ある程度 $M[G]$ で成り立つことを制御できる
  - $\mathbb{P}$ を決めたうえで、強制関係 $\Vdash$ によって、$M[G]$ で成り立つことを確認できる

※モデルが存在すれば無矛盾であるため、強制法でモデルを構成することで無矛盾性を証明できる

5/20

Isabelle/ZF

Isabelle/ZFにおける先行研究

- CHのZFC上の独立性証明 [Gunther et al. 20,22]
  - 強制法の形式化 (13K行)
  - CHの独立性証明 (16K行) ※ Lean3にもある
- ACのZF上の相対的無矛盾性証明 [Paulson 02]
  - 構成可能宇宙を形式化 (12K行)

▶ ¬ACの相対的無矛盾性証明は
形式化されていなかったので挑戦

6/20

本研究における Isabelle/ZF の利点

- 集合論に関する補題・糖衣構文が豊富
- 強制法の形式化 [Gunther et al. 20] がすでにある
  - ZF の c.t.m. の存在を仮定している
    - c.t.m. : countable transitive model

※ Lean3 にも強制法の形式化がある

7/20

注意

本研究では Isabelle/ZF 上でさらに形式化された ZF を扱う

- 今後出てくる「ZF のモデル」とは、Isabelle/ZF 上で形式化された ZF を満たす集合のこと
- [Paulson 02] の形式化を利用

8/20

[Paulson 02] の形式化

```
text‹De Bruijn representation.
  Unbound variables get their denotations from an environment.›

consts   formula :: i
datatype
  "formula" = Member ("x ∈ nat", "y ∈ nat")
            | Equal  ("x ∈ nat", "y ∈ nat")
            | Nand ("p ∈ formula", "q ∈ formula")
            | Forall ("p ∈ formula")
```

```
consts   satisfies :: "[i,i]⇒i"
primrec (*explicit lambda is required because the environment varies*)
  "satisfies(A,Member(x,y)) =
      (λenv ∈ list(A). bool_of_o (nth(x,env) ∈ nth(y,env)))"

  "satisfies(A,Equal(x,y)) =
      (λenv ∈ list(A). bool_of_o (nth(x,env) = nth(y,env)))"

  "satisfies(A,Nand(p,q)) =
      (λenv ∈ list(A). not ((satisfies(A,p)`env) and (satisfies(A,q)`env)))"

  "satisfies(A,Forall(p)) =
      (λenv ∈ list(A). bool_of_o (∀x∈A. satisfies(A,p) ` (Cons(x,env)) = 1))"
```

9/20

証明概略

証明概略 [Karagila 23]

ZF の c.t.m. M から出発して
ZF+¬AC のモデル N を構成する

- N は First Cohen Model と呼ばれるモデル
  - generic extension の部分モデルである symmetric extension のひとつ
- N は ZF を満たすが、整列可能定理を満たさない
  - $N \models$「単射 $\omega \to A$ がない無限集合 A が存在」
    - N ではこの A が整列できない

10/20

Isabelle/ZF による形式化

成果

本研究で形式証明した命題

```
theorem ZF_notAC_main_theorem :
  fixes M
  assumes "nat ≈ M" "M ⊨ ZF" "Transset(M)"
  shows "∃N. nat ≈ N ∧ N ⊨ ZF ∧ Transset(N)
          ∧ ¬(∀A ∈ N. ∃r ∈ N. wellordered(##N, A, r))"
```

意味

M を ZF の c.t.m. とする。このとき

ある ZF の c.t.m. N があって

N は整列可能定理を満たさない

11/20

作業工程

以下の工程に分けられる

1. symmetric extension の構成法の定義
2. ZF のモデルであることの証明
3. First Cohen Model の構成
4. それが ¬AC を満たすことの証明

12/20

作業量

約1万5千行のコード　　補題など(3K行)

1. symmetric extensionの定義(3K行)
2. ZFのモデルであることの証明(5K行)
3. First Cohen Modelの構成(2K行)
4. それが¬ACを満たすことの証明(2K行)

※Isabelle/ZFでの集合論の形式化の各先行研究と同じくらい

13/20

面倒だった点　自明なことの確認が大変

- クラスが本当にクラスであること
  - 実際に論理式を構成する必要がある
- 定義した関数が本当に関数であること
  - 特に「帰納的に定義された M 内の関数」
    - 仮定とZFからちゃんと構成できるか？
    - このような関数を定義するための補題が2千行以上

14/20

困難だった点 N が ZF をみたすことの証明

Isabelle/ZF で構成した N が $M[G]$ において
クラスであることが証明できなかった

- 書き下すのが面倒なだけでなく、非自明？
- これは [Karagila 23] で用いられている
  次の命題の証明に必要

命題

N が推移的かつ almost universal なクラスで
Δ_0-separation を満たすならば N は ZF の内部モデルである

15/20

解決策

- HS に相対化した強制関係 $\Vdash_{\mathbf{HS}}$ を形式化
  - 参考資料 [Karagila 23] に書かれている概念
  - 強制関係の定義の量化の動く範囲を HS に制限
  - $\Vdash_{\mathbf{HS}}$ は、symmetric extension に対し、
    generic extension に対する $\Vdash$ のように振舞う
- $\Vdash_{\mathbf{HS}}$ を用いて ZF のモデルであることを証明

```
lemma HS_truth_lemma:
  assumes
    "φ∈formula" "M_generic(G)"
  shows
    "⋀env. env∈list(HS) ⟹ arity(φ)≤length(env) ⟹
     (∃p∈G. p ⊩HS φ env) ⟷ SymExt(G), map(val(G),env) ⊨ φ"
```

```
lemma definition_of_forcing_HS:
  assumes
    "p∈P" "φ∈formula" "env∈list(HS)" "arity(φ)≤length(env)"
  shows
    "(p ⊩HS φ env) ⟷
     (∀G. M_generic(G) ∧ p∈G ⟶ SymExt(G), map(val(G),env) ⊨ φ)"
```

16/20

考察

考察 (1) c.t.m. アプローチについて

- 本研究で形式化したのは、「ZF の c.t.m. が存在 $\to$ ZF+¬AC の c.t.m. が存在」
  - ZF の c.t.m. の存在は、強制法の形式化 [Gunther et al. 20] を使うため仮定
- 証明したいのは Con(ZF)→Con(ZF+¬AC) だが、ZF の c.t.m. の存在は、Con(ZF) から証明できない
  - このギャップを埋めることができるが、通常 c.t.m. アプローチの紙の上での証明では大まかな議論のみで省略される
    - この部分の形式化できていない (本当は形式化したい)

17/20

形式化できていない部分

以下の形式化ができれば、
ZF の c.t.m. の存在の仮定をなくせる

- 任意の ZF の有限部分 Δ に対し、ZF の有限部分 Γ があって Γ の c.t.m. が存在すれば $\Delta + \neg$AC のモデルが存在する
  - 今回の形式化を修正すれば可能
- ZF の有限部分 Γ に対し、Γ の c.t.m. が存在する
  - ZF モデルの中で ZF のモデルを考える必要がある？
  - 形式化が難しい？

18/20

考察 (2) メタ/対象レベル

ZF のモデルの中の性質の証明が大変だった

- コード化された論理式を扱う必要があった
- メタレベルで成り立つことを
もう一度証明しなければいけなかった

▶ メタ/対象レベルの証明を同時に書ける or
他方に変換できるような機能があると便利

▶ 今回のテーマに限らず
数学基礎論の形式化でも有用

19/20

まとめ

¬AC の相対無矛盾性証明を Isabelle/ZF で形式化

- ZF の c.t.m. から出発し、
  ZF+¬AC をみたす symmetric extension を構成
- 紙の上では省略される c.t.m. に関する議論の形式化が残っている
- 参考資料の通りにいかず試行錯誤した部分も
- メタ/対象レベルの形式的証明を「つなげる」機能がほしい

20/20

参考文献 (1)

- K. Kunen, Set Theory An Introduction To Independence Proofs, North-Holland, 1980
  日本語訳: 藤田 博司 訳, 集合論: 独立性証明への案内, 日本評論社, 2008
- T. Jech, Set Theory: The Third Millennium Edition, Springer, 2002
- T. Jech, The Axiom of Choice, Dover Publications, 2008
- A. Karagila, Lecture Notes: Forcing & Symmetric Extensions, 2023

参考文献 (2)

- G. Klein et al., seL4: Formal Verification of an OS Kernel, 2014
- LC. Paulson, The Relative Consistency of the Axiom of Choice Mechanized Using Isabelle/ZF, 2003
- E. Gunther et al., Formalization of Forcing in Isabelle/ZF, 2020
- E. Gunther et al., The Independence of the Continuum Hypothesis in Isabelle/ZF, 2022

Lean4 を用いた Gödel の第一/第二不完全性定理の形式化

齋藤彰悟

東北大学大学院理学研究科数学専攻

0.1 FFL

形式論理学の形式化プロジェクト： `github.com/FormalizedFormalLogic`

- 古典命題論理 `LO.Propositional`
- 直観主義命題論理 `LO.IntProp`
- 古典命題様相論理 `LO.Modal`
- **古典一階述語論理 `LO.FirstOrder`**
- 直観主義一階述語論理 `LO.IntFO`

0.2 不完全性定理

理論 T を計算可能な一階算術の理論とする．

定理 0.2.1 (不完全性定理):
(G1) T が基礎的な算術を扱える程度に強く，まともならば，T から証明も反証もできない論理式が存在する．
(G2) T が十分強く，無矛盾ならば，T の無矛盾性を表す文は証明できない．

1986(!), Shanker Nqthm による G1 の形式化.
2004, O'Connor Coq による G1 の形式化.
2004, Harrison HOL Light による G1 の形式化.
2021, Paulson Isabelle/HOL による G1 と G2 の形式化 [1]． 算術ではなく遺伝的有限集合の理論 HF (ペアノ算術 PA と同等)の上で証明している．

- 私が知る限り唯一の第二不完全性定理の形式化.

0.2 不完全性定理

私は次の不完全性定理の一つのバリエーションを形式化した[1]：

定理 0.2.2:
(G1) T が R_0 より強く Σ_1-健全ならば，T から証明も反証もできない論理式が存在する．
(G2) T が $\mathsf{I}\Sigma_1$ より強く無矛盾ならば，T の無矛盾性を表す文は証明できない．

- Σ_1-健全：T から証明可能な Σ_1文は標準モデルの上で真.
- R_0 : Cobham の最弱の算術.
- $\mathsf{I}\Sigma_1$: ペアノ算術の断片理論.

[1]後述するように完全性定理を用いているため非構成的.

0.2 不完全性定理

以下の 5 つのステップを踏んで証明を行う．

1. 一階述語論理の形式化.
   - 項と論理式，代入操作などの形式化.
   - 証明可能性，充足性の形式化.
2. 完全性定理.
   - 証明探索
3. $\mathsf{I}\Sigma_1$ の内部で算術を展開する．
   - 指数関数が定義可能であることを証明する．
   - 有限集合，有限列といった基礎概念をコード化する．
4. メタ数学の算術化.
   - 項，論理式，証明可能性などをコード化する．
5. Hilbert-Bernays-Löb の可証性条件

1. 一階述語論理の形式化.

1.1 項・論理式の形式化

論理式（擬論理式）は常に否定標準形を取るように定義する[1]． すなわち，

$$\varphi, \psi ::= \top \mid \bot \mid R(\vec{v}) \mid \neg R(\vec{v}) \mid \varphi \land \psi \mid \varphi \lor \psi \mid \forall \varphi \mid \exists \varphi$$

```
inductive Semiformula (L : Language) (ξ : Type*) : ℕ → Type _ where
  | verum  {n} : Semiformula L ξ n
  | falsum {n} : Semiformula L ξ n
  | rel    {n} : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
  | nrel   {n} : {arity : ℕ} → L.Rel arity → (Fin arity → Semiterm L ξ n) → Semiformula L ξ n
  | and    {n} : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
  | or     {n} : Semiformula L ξ n → Semiformula L ξ n → Semiformula L ξ n
  | all    {n} : Semiformula L ξ (n + 1) → Semiformula L ξ n
  | ex     {n} : Semiformula L ξ (n + 1) → Semiformula L ξ n
```

論理式	糖衣構文
$\overline{98} + \overline{6748} = \overline{6846}$	“98 + 6748 = 6846”
$(\forall x)(\forall y)[a \cdot (x + y) = a \cdot x + a \cdot y]$	“a | ∀ x y, a * (x + y) = a * x + a * y”
$(\forall x)\left[x < \overline{n} \leftrightarrow \bigvee_{i<n} x = \overline{i}\right]$	“∀ x, x < ↑n ↔ ⋁ i < n, x = ↑i”

[1]否定 ¬ は論理式の関数として定義する． また $\varphi \to \psi := \neg\varphi \lor \psi$ とする．

1.2 証明可能性の形式化

証明体系には Tait 計算を採用する．

$$\dfrac{}{R(\vec{v}), \neg R(\vec{v}), \Delta}\,\texttt{axL} \quad \dfrac{}{\top, \Delta}\,\texttt{verum} \quad \dfrac{\varphi, \psi, \Delta}{\varphi \lor \psi, \Delta}\,\texttt{or} \quad \dfrac{\varphi, \Delta \qquad \psi, \Delta}{\varphi \land \psi, \Delta}\,\texttt{and} \quad \dfrac{\varphi^{+}(\&0), \Delta^{+}}{\forall \varphi, \Delta}\,\texttt{all}^{1} \quad \dfrac{\varphi(t), \Delta}{\exists \varphi, \Delta}\,\texttt{ex} \quad \underset{(\text{if } \Delta \subseteq \Gamma)}{\dfrac{\Delta}{\Gamma}\,\texttt{wk}} \quad \dfrac{\varphi, \Delta \qquad \neg\varphi, \Delta}{\Delta}\,\texttt{cut} \quad \underset{(\text{if } \varphi \in T)}{\dfrac{}{\varphi}\,\texttt{root}}$$

```
inductive Derivation (T : Theory L) : Sequent L → Type _
  | axL (Δ) {k} (r : L.Rel k) (v) : Derivation T (rel r v :: nrel r v :: Δ)
  | verum (Δ)    : Derivation T (⊤ :: Δ)
  | or {Δ p q}   : Derivation T (p :: q :: Δ) → Derivation T (p ⋎ q :: Δ)
  | and {Δ p q}  : Derivation T (p :: Δ) → Derivation T (q :: Δ) → Derivation T (p ⋏ q :: Δ)
  | all {Δ p}    : Derivation T (Rew.free.hom p :: Δ⁺) → Derivation T ((∀' p) :: Δ)
  | ex {Δ p} (t) : Derivation T (p/[t] :: Δ) → Derivation T ((∃' p) :: Δ)
  | wk {Δ Γ}     : Derivation T Δ → Δ ⊆ Γ → Derivation T Γ
  | cut {Δ p}    : Derivation T (p :: Δ) → Derivation T (~p :: Δ) → Derivation T Δ
  | root {p}     : p ∈ T → Derivation T [p]
```

Tait 計算は推論規則が少ないため扱いやすく，LK などの他の推論規則に比べ様々な証明が簡易になる（こともある）．

[1]ここでは自由変数を $\&0, \&1, \ldots$ と表記する． また φ^{+}, Γ^{+} はそれぞれに含まれる自由変数をインクリメントしたもの

2. 完全性定理

2.1 完全性定理

定理 2.1.1 (完全性定理): すべての 論理式 φ について,

$$T \vdash \varphi \Longleftrightarrow T \vDash \varphi$$

$\Longrightarrow$ 方向（健全性定理）は証明に関する帰納法により従う. $\Longleftarrow$ を示すには以下を証明すれば良い.

補題 2.1.1: 推件 Γ について, 次のいずれかが成立する.

- $T \vdash_{\mathsf{T}} \Gamma$
- すべての $\varphi \in \Gamma$ を充足しないような T のモデルが存在する.

3. $\mathsf{I}\Sigma_1$ の内部で算術を展開する

3.1 （形式的）証明のルーチン

何らかの算術の体系 T からある論理式を証明できること を証明したい.

$$\mathsf{Lean} \vdash \text{“}T \vdash \varphi\text{”}$$

しかし，そもそも Lean が形式化された数学であり，その内部でさらに形式化された証明を証明するのはかなり煩雑になる（さらに後には「形式化された形式化された形式化された証明」のようなものさえ扱うことになる).

これは大変なので，完全性定理を用いて代わりに意味論帰結を証明する.

$$\mathsf{Lean} \vdash \text{“}T \vDash \varphi\text{”}$$

意味論的な議論では，Lean のライブラリに用意された代数学の種々の補題やメタプログラミングや自動証明が利用できるため，より簡単に作業を行うことができる.

3.1 （形式的）証明のルーチン

まず理論の任意のモデル V を固定する.

```
variable {V : Type*} [ORingStruc V] [V ⊧ₘ* Iopen]
```

- **ORingStruc V:** V が言語 $\mathcal{L}_{\mathrm{OR}}$ の構造であることを主張する typeclass.
- **V ⊧ₘ* Iopen:** V が理論 Iopen を満たすことを主張する typeclass.

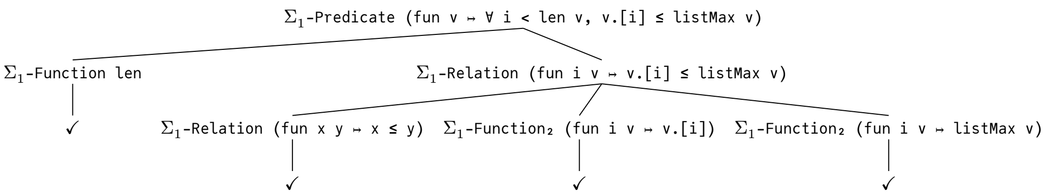
この仮定のもとで証明を行う. 関数を追加したいならば選択関数を用いる.

```
lemma sqrt_exists_unique (a : V) :
    ∃! x, x * x ≤ a ∧ a < (x + 1) * (x + 1) := by ...
def sqrt (a : V) : V := Classical.choose! (sqrt_exists_unique a)
prefix:75 "√" => sqrt
@[simp] lemma sqrt_mul_self (a : V) : √(a * a) = a := by ...
```

ただし，帰納法を適用するために，定義した関数や関係を含む述語がある算術的階層に含まれることが示せなければならない.

3.2 definability tactic

definability tactic は Lean の述語がある算術的階層に含まれることを（可能なら）自動証明する.



typeclass に登録された関数や関係は自動証明の末尾を示すために使用される.

```
instance exp_definable : Σ₀-Function₁ (Exp.exp : V → V) := by ...
instance length_definable : Σ₀-Function₁ (‖·‖ : V → V) := by ...
instance dvd_definable : Σ₀-Relation (fun a b : V ↦ a ∣ b) := by ...
instance Language.isSemiterm_definable : Δ₁-Relation L.IsSemiterm := by ...
```

3.3 指数関数

指数関数のグラフ $\mathrm{Exp}(x, y) \Leftrightarrow 2^x = y$ は Σ_0 論理式で定義可能 [2], [3].

$$\mathrm{Exp}(a_0, a_1) \equiv [[a_0 = 0] \land [a_1 = 1]] \lor [(\exists x_0 < ((((a_1 \cdot a_1) \cdot a_1) \cdot a_1) + 1))[(\exists x_1 < ((((a_1 \cdot a_1) \cdot a_1) \cdot a_1) + 1))[[[(\exists x_2 < (x_0 + 1))[[[[0 \nless (((1 + 1) + 1) + 1)] \lor [(\exists x_3 < (((1 + 1) + 1) + 1))[x_0 = (((((1+1)+1)+1) \cdot x_2)+x_3)]]] \land [[(((1+1)+1)+1) \neq 0] \lor [x_2 = 0]]] \land [(\exists x_3 < (x_2+1))[[[[0 \nless (((1+1)+1)+1)] \lor [(\exists x_4 < (((1+1)+1)+1))[x_2 = (((((1+1)+1)+1) \cdot x_3)+x_4)]]] \land [[(((1+1)+1)+1) \neq 0] \lor [x_3 = 0]]] \land [[[[((((1+1)+1)+1) \cdot x_3) \neq x_2] \land [((((1+1)+1)+1) \cdot x_3) \nless x_2]] \lor [x_2 = (((((1+1)+1)+1) \cdot x_3)+1)]] \land [[x_2 \nless ((((1+1)+1)+1) \cdot x_3)] \lor [1 = 0]]]]]]] \land [(\exists x_2 < (x_1+1))[[[[0 \nless (((1+1)+1)+1)] \lor [(\exists x_3 < (((1+1)+1)+1))[x_1 = (((((1+1)+1)+1) \cdot x_2)+x_3)]]] \land [[(((1+1)+1)+1) \neq 0] \lor [x_2 = 0]]] \land [(\exists x_3 < (x_2+1))[[[[0 \nless (((1+1)+1)+1)] \lor [(\exists x_4 < (((1+1)+1)+1))[x_2 = (((((1+1)+1)+1) \cdot x_3)+x_4)]]] \land [[(((1+1)+1)+1) \neq 0] \lor [x_3 = 0]]] \land [[[[((((1+1)+1)+1) \cdot x_3) \neq x_2] \land [((((1+1)+1)+1) \cdot x_3) \nless x_2]] \lor [x_2 = (((((1+1)+1)+1) \cdot x_3)+(1+1))]] \land [[x_2 \nless ((((1+1)+1)+1) \cdot x_3)] \lor [(1+1) = 0]]]]]]] \land [[(\forall x_2 < (a_1+1))[[x_2 = (1+1)] \lor [[[[0 \nless x_2] \lor [(\exists x_3 < (x_2+1))[[1 < x_3] \land [[(\exists x_4 < (x_2+1))[x_2 = (x_3 \cdot x_4)]] \land [(\forall x_4 < (x_3+1))[x_3 \neq ((1+1) \cdot x_4)]]]]]] \lor [(\forall x_3 < ((1+1) \cdot x_2))[[[(\exists x_4 < (x_3+1))[[[[0 \nless 1] \lor [(\exists x_5 < 1)[x_3 = ((1 \cdot x_4)+x_5)]]] \land [[1 \neq 0] \lor [x_4 = 0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1) \cdot x_5)]]]] \lor [[(\forall x_4 < (x_3+1))[[[[0 < (1+1)] \land [(\forall x_5 < (1+1))[x_3 \neq (((1+1) \cdot x_4)+x_5)]]] \lor [[(1+1) = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 = ((1+1) \cdot x_5)]]]] \lor [(\exists x_4 < (x_3+1))[[[0 < x_4] \land [(\forall x_5 < (x_4+1))[[1 \nless x_5] \lor [[(\forall x_6 < (x_4+1))[x_4 \neq (x_5 \cdot x_6)]] \lor [(\exists x_6 < (x_5+1))[x_5 = ((1+1) \cdot x_6)]]]]]] \land [[(1+1) < x_4] \land [[[(\exists x_5 < (x_3+1))[[[[0 \nless x_4] \lor [(\exists x_6 < x_4)[x_3 = ((x_4 \cdot x_5)+x_6)]]] \land [[x_4 \neq 0] \lor [x_5 = 0]]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1) \cdot x_6)]]]] \land [(\forall x_5 < (x_4+1))[[[[(x_5 \cdot x_5) \neq x_4] \land [(x_5 \cdot x_5) \nless x_4]] \lor [x_4 \nless ((x_5+1) \cdot (x_5+1))]] \lor [[(x_5 \cdot x_5) \neq x_4] \lor [(\forall x_6 < (x_3+1))[[[[0 < x_5] \land [(\forall x_7 < x_5)[x_3 \neq ((x_5 \cdot x_6)+x_7)]]] \lor [[x_5 = 0] \land [x_6 \neq 0]]] \lor [(\exists x_7 < (x_6 + 1))[x_6 = ((1 + 1) \cdot x_7)]]]]]]]] \lor [[(\exists x_5 < (x_4 + 1))[[[[(x_5 \cdot x_5) = x_4] \lor [(x_5 \cdot x_5) < x_4]] \land [x_4 < ((x_5 + 1) \cdot (x_5 + 1))]] \land [[(x_5 \cdot x_5) = x_4] \land [(\exists x_6 < (x_3 + 1))[[[[0 \nless x_5] \lor [(\exists x_7 < x_5)[x_3 = ((x_5 \cdot x_6)+x_7)]]] \land [[x_5 \neq 0] \lor [x_6 = 0]]] \land [(\forall x_7 < (x_6+1))[x_6 \neq ((1+1) \cdot x_7)]]]]]]] \land [(\forall x_5 < (x_3+1))[[[[0 < x_4] \land [(\forall x_6 < x_4)[x_3 \neq ((x_4 \cdot x_5)+x_6)]]] \lor [[x_4 = 0] \land [x_5 \neq 0]]] \lor [(\exists x_6 < (x_5+1))[x_5 = ((1+1) \cdot x_6)]]]]]]]]]] \lor [(\forall x_4 < (x_3+1))[[[[0 < x_2] \land [(\forall x_5 < x_2)[x_3 \neq ((x_2 \cdot x_4)+x_5)]]] \lor [[x_2 = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 = ((1+1) \cdot x_5)]]]]]]] \lor [[[(\exists x_3 < (x_0+1))[[(\exists x_4 < (x_0+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_0 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2 \cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2 \cdot x_5) \neq x_4] \land [(x_2 \cdot x_5) \nless x_4]] \lor [x_4 = ((x_2 \cdot x_5)+x_3)]] \land [[x_4 \nless (x_2 \cdot x_5)] \lor [x_3 = 0]]]]]] \land [(\exists x_4 < (x_0+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_5 < (x_2 \cdot x_2))[x_0 = (((x_2 \cdot x_2) \cdot x_4)+x_5)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_6 < (x_2 \cdot x_2))[x_4 = (((x_2 \cdot x_2) \cdot x_5)+x_6)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2 \cdot x_2) \cdot x_5) \neq x_4] \land [((x_2 \cdot x_2) \cdot x_5) \nless x_4]] \lor [x_4 = (((x_2 \cdot x_2) \cdot x_5)+((1+1) \cdot x_3))]] \land [[x_4 \nless ((x_2 \cdot x_2) \cdot x_5)] \lor [((1+1) \cdot x_3) = 0]]]]]]]] \land [(\exists x_3 < (x_1+1))[[(\exists x_4 < (x_1+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_1 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2 \cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2 \cdot x_5) \neq x_4] \land [(x_2 \cdot x_5) \nless x_4]] \lor [x_4 = ((x_2 \cdot x_5)+x_3)]] \land [[x_4 \nless (x_2 \cdot x_5)] \lor [x_3 = 0]]]]]] \land [(\exists x_4 < (x_1+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_5 < (x_2 \cdot x_2))[x_1 = (((x_2 \cdot x_2) \cdot x_4)+x_5)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_6 < (x_2 \cdot x_2))[x_4 = (((x_2 \cdot x_2) \cdot x_5)+x_6)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2 \cdot x_2) \cdot x_5) \neq x_4] \land [((x_2 \cdot x_2) \cdot x_5) \nless x_4]] \lor [x_4 = (((x_2 \cdot x_2) \cdot x_5)+(x_3 \cdot x_3))]] \land [[x_4 \nless ((x_2 \cdot x_2) \cdot x_5)] \lor [(x_3 \cdot x_3) = 0]]]]]]]]] \lor [[(\exists x_3 < (x_0+1))[[(\exists x_4 < (x_0+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_0 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2 \cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2 \cdot x_5) \neq x_4] \land [(x_2 \cdot x_5) \nless x_4]] \lor [x_4 = ((x_2 \cdot x_5)+x_3)]] \land [[x_4 \nless (x_2 \cdot x_5)] \lor [x_3 = 0]]]]]] \land [(\exists x_4 < (x_0+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_5 < (x_2 \cdot x_2))[x_0 = (((x_2 \cdot x_2) \cdot x_4)+x_5)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_6 < (x_2 \cdot x_2))[x_4 = (((x_2 \cdot x_2) \cdot x_5)+x_6)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2 \cdot x_2) \cdot x_5) \neq x_4] \land [((x_2 \cdot x_2) \cdot x_5) \nless x_4]] \lor [x_4 = (((x_2 \cdot x_2) \cdot x_5)+(((1+1) \cdot x_3)+1))]] \land [[x_4 \nless ((x_2 \cdot x_2) \cdot x_5)] \lor [(((1+1) \cdot x_3)+1) = 0]]]]]]]] \land [(\exists x_3 < (x_1+1))[[(\exists x_4 < (x_1+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_1 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless x_2] \lor [(\exists x_6 < x_2)[x_4 = ((x_2 \cdot x_5)+x_6)]]] \land [[x_2 \neq 0] \lor [x_5 = 0]]] \land [[[[(x_2 \cdot x_5) \neq x_4] \land [(x_2 \cdot x_5) \nless x_4]] \lor [x_4 = ((x_2 \cdot x_5)+x_3)]] \land [[x_4 \nless (x_2 \cdot x_5)] \lor [x_3 = 0]]]]]] \land [(\exists x_4 < (x_1+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_5 < (x_2 \cdot x_2))[x_1 = (((x_2 \cdot x_2) \cdot x_4)+x_5)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_4 = 0]]] \land [(\exists x_5 < (x_4+1))[[[[0 \nless (x_2 \cdot x_2)] \lor [(\exists x_6 < (x_2 \cdot x_2))[x_4 = (((x_2 \cdot x_2) \cdot x_5)+x_6)]]] \land [[(x_2 \cdot x_2) \neq 0] \lor [x_5 = 0]]] \land [[[[((x_2 \cdot x_2) \cdot x_5) \neq x_4] \land [((x_2 \cdot x_2) \cdot x_5) \nless x_4]] \lor [x_4 = (((x_2 \cdot x_2) \cdot x_5) + ((1+1) \cdot (x_3 \cdot x_3)))]] \land [[x_4 \nless ((x_2 \cdot x_2) \cdot x_5)] \lor [((1+1) \cdot (x_3 \cdot x_3)) = 0]]]]]]]]]]]]] \land [(\exists x_2 < ((a_1 \cdot a_1)+1))[[x_2 \neq (1+1)] \land [[[[0 < x_2] \land [(\forall x_3 < (x_2+1))[[1 \nless x_3] \lor [[(\forall x_4 < (x_2+1))[x_2 \neq (x_3 \cdot x_4)]] \lor [(\exists x_4 < (x_3+1))[x_3 = ((1+1) \cdot x_4)]]]]]] \land [(\exists x_3 < ((1+1) \cdot x_2))[[[(\forall x_4 < (x_3+1))[[[[0 < 1] \land [(\forall x_5 < 1)[x_3 \neq ((1 \cdot x_4)+x_5)]]] \lor [[1 = 0] \land [x_4 \neq 0]]] \lor [(\exists x_5 < (x_4+1))[x_4 = ((1+1) \cdot x_5)]]]] \land [[(\exists x_4 < (x_3+1))[[[[0 \nless (1+1)] \lor [(\exists x_5 < (1+1))[x_3 = (((1+1) \cdot x_4)+x_5)]]] \land [[(1+1) \neq 0] \lor [x_4 = 0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1) \cdot x_5)]]]] \land [(\forall x_4 < (x_3+1))[[[0 \nless x_4] \lor [(\exists x_5 < (x_4+1))[[1 < x_5] \land [[(\exists x_6 < (x_4+1))[x_4 = (x_5 \cdot x_6)]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1) \cdot x_6)]]]]]] \lor [[(1+1) \nless x_4] \lor [[[(\forall x_5 < (x_3+1))[[[[0 < x_4] \land [(\forall x_6 < x_4)[x_3 \neq ((x_4 \cdot x_5)+x_6)]]] \lor [[x_4 = 0] \land [x_5 \neq 0]]] \lor [(\exists x_6 < (x_5+1))[x_5 = ((1+1) \cdot x_6)]]]] \lor [(\exists x_5 < (x_4+1))[[[[(x_5 \cdot x_5) = x_4] \lor [(x_5 \cdot x_5) < x_4]] \land [x_4 < ((x_5+1) \cdot (x_5+1))]] \land [[(x_5 \cdot x_5) = x_4] \land [(\exists x_6 < (x_3+1))[[[[0 \nless x_5] \lor [(\exists x_7 < x_5)[x_3 = ((x_5 \cdot x_6)+x_7)]]] \land [[x_5 \neq 0] \lor [x_6 = 0]]] \land [(\forall x_7 < (x_6+1))[x_6 \neq ((1+1) \cdot x_7)]]]]]]]] \land [[(\forall x_5 < (x_4+1))[[[[(x_5 \cdot x_5) \neq x_4] \land [(x_5 \cdot x_5) \nless x_4]] \lor [x_4 \nless ((x_5+1) \cdot (x_5+1))]] \lor [[(x_5 \cdot x_5) \neq x_4] \lor [(\forall x_6 < (x_3+1))[[[[0 < x_5] \land [(\forall x_7 < x_5)[x_3 \neq ((x_5 \cdot x_6)+x_7)]]] \lor [[x_5 = 0] \land [x_6 \neq 0]]] \lor [(\exists x_7 < (x_6+1))[x_6 = ((1+1) \cdot x_7)]]]]]]] \lor [(\exists x_5 < (x_3+1))[[[[0 \nless x_4] \lor [(\exists x_6 < x_4)[x_3 = ((x_4 \cdot x_5)+x_6)]]] \land [[x_4 \neq 0] \lor [x_5 = 0]]] \land [(\forall x_6 < (x_5+1))[x_5 \neq ((1+1) \cdot x_6)]]]]]]]]]]] \land [(\exists x_4 < (x_3+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_3 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [(\forall x_5 < (x_4+1))[x_4 \neq ((1+1) \cdot x_5)]]]]]]] \land [[(\exists x_3 < (x_0+1))[[[[0 \nless x_2] \lor [(\exists x_4 < x_2)[x_0 = ((x_2 \cdot x_3)+x_4)]]] \land [[x_2 \neq 0] \lor [x_3 = 0]]] \land [(\exists x_4 < (x_3+1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_3 = ((x_2 \cdot x_4)+x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [[[[(x_2 \cdot x_4) \neq x_3] \land [(x_2 \cdot x_4) \nless x_3]] \lor [x_3 = ((x_2 \cdot x_4)+a_0)]] \land [[x_3 \nless (x_2 \cdot x_4)] \lor [a_0 = 0]]]]]] \land [(\exists x_3 < (x_1+1))[[[[0 \nless x_2] \lor [(\exists x_4 < x_2)[x_1 = ((x_2 \cdot x_3)+x_4)]]] \land [[x_2 \neq 0] \lor [x_3 = 0]]] \land [(\exists x_4 < (x_3 + 1))[[[[0 \nless x_2] \lor [(\exists x_5 < x_2)[x_3 = ((x_2 \cdot x_4) + x_5)]]] \land [[x_2 \neq 0] \lor [x_4 = 0]]] \land [[[[(x_2 \cdot x_4) \neq x_3] \land [(x_2 \cdot x_4) \nless x_3]] \lor [x_3 = ((x_2 \cdot x_4) + a_1)]] \land [[x_3 \nless (x_2 \cdot x_4)] \lor [a_1 = 0]]]]]]]]]]]]]]]$$

3.4 遺伝的有限集合

「x は y の要素」 $\Longleftrightarrow$ 「y を二進数展開したとき x 桁目が 1」 と定義する.

$$x \in y \Longleftrightarrow \mathrm{Bit}(x, y) \Longleftrightarrow \lfloor y/2^x \rfloor \bmod 2 = 1$$

Ackermann coding によって遺伝的有限集合 V_ω が扱える. $I\Sigma_1$ のもとで基礎的な集合論が展開できる.

```
theorem finset_comprehension₁ {Γ} {P : V → Prop} (hP : Γ-[1]-Predicate P) (a : V) :
    ∃ s < exp a, ∀ i < a, i ∈ s ↔ P i -- (制限された)内包公理図式
theorem sUnion_exists_unique (s : V) :
    ∃! u, ∀ x, (x ∈ u ↔ ∃ t ∈ s, x ∈ t) -- 和集合公理
theorem sigma₁_replacement {f : V → V} (hf : 𝚺₁-Function₁ f) (s : V) :
    ∃! t, ∀ y, (y ∈ t ↔ ∃ x ∈ s, y = f x) -- (制限された)置換公理図式
```

また有限の関数/有限列も扱える.

```
def IsMapping (m : V) : Prop := ∀ x ∈ domain m, ∃! y, ⟪x, y⟫ ∈ m
def Seq (s : V) : Prop := IsMapping s ∧ ∃ l, domain s = under l
```

3.5 再帰的定義

定理 3.5.1: $\Phi_{\boldsymbol{C}}(\vec{v}, x)$ を***クラス*** $\boldsymbol{C}$ をパラメータとして取る述語だとする． Φ が以下を満たすならば，

1. 述語 $P(c, \vec{v}, x) \coloneqq \Phi_{\{z \mid z \in c\}}(\vec{v}, x)$ は Δ_1 定義可能.
2. 単調： $\boldsymbol{C} \subseteq \boldsymbol{C}'$ ならば， $\Phi_{\boldsymbol{C}}(\vec{v}, x) \Longrightarrow \Phi_{\boldsymbol{C}'}(\vec{v}, x)$
3. 有限： $\Phi_{\boldsymbol{C}}(\vec{v}, x)$ ならば，ある m が存在して $\Phi_{\{z \in \boldsymbol{C} \mid z < m\}}(\vec{v}, x)$

次を満たす Σ_1 定義可能な述語 $\mathrm{Fix}_{\Phi}(\vec{v}, x)$ が存在する．

$$\mathrm{Fix}_{\Phi}(\vec{v}, x) \Longleftrightarrow \Phi_{\{x \mid \mathrm{Fix}_{\Phi}(\vec{v}, x)\}}(\vec{v}, x)$$

さらに次を満たすならば，$\mathrm{Fix}_{\Phi}(\vec{v}, x)$ は Δ_1 定義可能でその構造帰納法が証明できる．

4. 強有限： $\Phi_{\boldsymbol{C}}(\vec{v}, x) \Longrightarrow \Phi_{\{y \in \boldsymbol{C} \mid y < x\}}(\vec{v}, x)$

4. メタ数学の算術化

4.1 項・論理式のコーディング

Δ_1述語 T_C を以下のように定める．

$$
\begin{aligned}
u \in T_C \iff & (\exists z)\left[u = \widehat{\#z}\right] \vee (\exists x)\left[u = \widehat{\&x}\right] \vee \\
& (\exists k, f, v)\left[\mathrm{Func}(k, f) \wedge \mathrm{len}(v) = k \wedge (\forall i < k)[\mathrm{nth}(v, i) \in \boldsymbol{C}] \wedge u = \widehat{f^k(v)}\right]
\end{aligned}
$$

$$
\begin{aligned}
\text{bounded variable:} \quad & \widehat{\#z} := \langle 0,\, z\rangle + 1 \\
\text{free variable:} \quad & \widehat{\&z} := \langle 1,\, z\rangle + 1 \\
\text{function:} \quad & \widehat{f^k(v)} := \langle 2,\, f,\, k,\, v\rangle + 1
\end{aligned}
$$

`L` を形式化された言語だとする． fixpoint construction により T_C の不動点 `L.IsUTerm : V → Prop` が得られる． また， その構造帰納法から束縛変数の最大値+1 を返す Σ_1関数 `L.termBV : V → V` が定義でき， `t ∈ V` がコード化された（`n` 個の束縛変数を持つ）擬項であることを表す Δ_1述語 `L.IsSemiterm n t : Prop` が定義される． 同様に形式化された擬論理式を指す Δ_1述語 `IsSemiformula n p : Prop` が定義される．

```
def Language.IsSemiterm (n t : V) : Prop := L.IsUTerm t ∧ L.termBV t ≤ n
```


4.2 Tait 計算のコーディング

T を Δ_1 定義可能な理論とする． Δ_1述語 D^T_C を項や論理式と同様に定義し，

$$
\begin{aligned}
d \in D^T_C \iff & (\forall p \in \mathrm{sqt}(d))[\mathrm{Semiformula}(0, p)] \wedge \\
& \Big[(\exists s, p)[d = \mathrm{AXL}(s, p) \wedge p \in s \wedge \hat{\neg} p \in s] \\
& (\exists s)\left[d = \top\text{-INTRO}(s) \wedge \hat{\top} \in s\right] \vee \\
& (\exists s, p, q)\left(\exists d_p, d_q \in \boldsymbol{C}\right)\left[d = \wedge\text{-INTRO}\left(s, p, q, d_p, d_q\right) \wedge \widehat{p \wedge q} \in s \wedge \mathrm{sqt}\left(d_p\right) = s \cup \{p\} \wedge \mathrm{sqt}\left(d_q\right) = s \cup \{q\}\right] \vee \\
& (\exists s, p, q)(\exists d \in \boldsymbol{C})\left[d = \vee\text{-INTRO}(s, p, q, d) \wedge \widehat{p \vee q} \in s \wedge \mathrm{sqt}(d) = s \cup \{p, q\}\right] \vee \\
& (\exists s, p)(\exists d \in \boldsymbol{C})\left[d = \forall\text{-INTRO}(s, p, d) \wedge \widehat{\forall} p \in s \wedge \mathrm{sqt}(d) = s^+ \cup \left\{p^+\left(\widehat{\&0}\right)\right\}\right] \vee \\
& (\exists s, p, t)(\exists d \in \boldsymbol{C})\left[d = \exists\text{-INTRO}(s, p, t, d) \wedge \widehat{\exists} p \in s \wedge \mathrm{sqt}(d) = s \cup \{p(t)\}\right] \vee \\
& (\exists s)(\exists d' \in \boldsymbol{C})[d = \mathrm{WK}(s, d') \wedge s \supseteq \mathrm{sqt}(d')] \vee \\
& (\exists s)(\exists d' \in \boldsymbol{C})\left[d = \mathrm{SHIFT}(s, d') \wedge s = \mathrm{sqt}(d')^+\right] \vee \\
& (\exists s, p)(\exists d_1, d_2 \in \boldsymbol{C})[d = \mathrm{CUT}(s, p, d_1, d_2) \wedge \mathrm{sqt}(d_1) = s \cup \{p\} \wedge \mathrm{sqt}(d_2) = s \cup \{\hat{\neg} p\}] \vee \\
& (\exists s, p)[d = \mathrm{ROOT}(s, p) \wedge p \in s \wedge p \in T]\Big]
\end{aligned}
$$

$$\mathrm{sqt}(d) \coloneqq \pi_1(d-1)$$
$$\mathrm{AXL}(s,p) \coloneqq \langle s,\ 0,\ p\rangle + 1$$

$$\top\text{-INTRO}(s) \coloneqq \langle s,\ 1,\ 0\rangle + 1$$
$$\wedge\text{-INTRO}(s,p,q,d_p,d_q) \coloneqq \langle s,\ 2,\ p,\ q,\ d_p,\ d_q\rangle + 1$$
$$\vee\text{-INTRO}(s,p,q,d) \coloneqq \langle s,\ 3,\ p,\ q,\ d\rangle + 1$$
$$\forall\text{-INTRO}(s,p,d) \coloneqq \langle s,\ 4,\ p,\ d\rangle + 1$$
$$\exists\text{-INTRO}(s,p,t,d) \coloneqq \langle s,\ 5,\ p,\ t,\ d\rangle + 1$$

$$\mathrm{WK}(s,d) \coloneqq \langle s,\ 6,\ d\rangle + 1$$
$$\mathrm{SHIFT}(s,d) \coloneqq \langle s,\ 7,\ d\rangle + 1$$
$$\mathrm{CUT}(s,p,d_1,d_2) \coloneqq \langle s,\ 8,\ p,\ d_1 d_2\rangle + 1$$
$$\mathrm{ROOT}(s,p) \coloneqq \langle s,\ 9,\ p\rangle + 1$$

D^T_C の不動点を取って `T.Derivation` とする．以下のように定める．

```
def Language.Theory.Derivable (T) (s : V) : Prop := ∃ d, T.DerivationOf d s
def Language.Theory.Provable (T) (p : V) : Prop := T.Derivable {p}
```

定義より `T.Derivable` や `T.Provable` は Σ_1.

5. Hilbert-Bernays-Löb の可証性条件

5.1 可証性条件

補題 5.1.1: T を $\mathsf{I}\Sigma_1$ より強い理論，U を R_0 より強い Δ_1 定義可能な理論だとする． このとき，

D1 $U \vdash \sigma \Longrightarrow T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner)$

D2 $T \vdash \mathrm{Provable}_U(\ulcorner\sigma \to \tau\urcorner) \to \mathrm{Provable}_U(\ulcorner\sigma\urcorner) \to \mathrm{Provable}_U(\ulcorner\tau\urcorner)$

D3 $T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner) \to \mathrm{Provable}_U(\ulcorner\mathrm{Provable}_U(\ulcorner\sigma\urcorner)\urcorner)$

$\mathbb{N} \models T$ ならば[1] 更に次が成り立つ．

D1′ $U \vdash \sigma \Longleftrightarrow T \vdash \mathrm{Provable}_U(\ulcorner\sigma\urcorner)$

D1 及び D2, D1′ は形式化された証明の性質を地道に証明すれば示せる．

[1]実際には Σ_1 健全性で十分．

5.2 形式化された Σ_1-完全性

D3 は直接示すのは難しいが，次の補題より従う．

補題 5.2.1 (形式化された Σ_1-完全性): T を $\mathsf{I}\Sigma_1$ より強い理論，U を R_0 より強い Δ_1 定義可能な理論だとする． 文 σ が Σ_1 論理式ならば，次が証明可能．

$$T \vdash \sigma \to \mathrm{Provable}_U(\ulcorner\sigma\urcorner)$$

証明 T のモデル V の内部で作業する．

次を示せばよい： すべての Σ_1 論理式 $\varphi(x_1, ..., x_k)$, $a_1, ..., a_k \in V$ について，$V \models \varphi[a_1, ..., a_k] \Longrightarrow \mathrm{Provable}_U(\ulcorner\varphi\urcorner(\overline{a_1}, ..., \overline{a_k}))$ これは φ に関する帰納法で示せる． □

```
theorem provableₐ_sigma₁_complete
    {σ : Sentence ℒₒᵣ} (hσ : Hierarchy Σ 1 σ) :
    T ⊢! σ → U.bewₐ σ
```

6. 不完全性定理

6.1 第一不完全性定理

G1 は 1, 2, 3, 4 までの結果で証明できる.

定理 6.1.1 (G1): T が Δ_1 定義可能で R_0 より強く Σ_1-健全ならば, T から証明も反証もできない論理式が存在する.

証明 $D \coloneqq \{\ulcorner\varphi\urcorner \mid \varphi : 1$ 変数の論理式$, T \vdash \neg\varphi(\ulcorner\varphi\urcorner)\}$ と定義する. $T \vdash \pi \Leftrightarrow \mathbb{N} \models \mathrm{Provable}_T(\ulcorner\pi\urcorner)$, また $\mathrm{Provable}_T$ は Σ_1 定義可能なので, D は r.e.集合.

表現定理より次を満たす θ が存在する: すべての $n \in \mathbb{N}$ について $n \in D \Longleftrightarrow T \vdash \theta(\overline{n})$.

$n = \ulcorner\theta\urcorner$ と置くと,

$$T \vdash \theta(\ulcorner\theta\urcorner) \Longleftrightarrow \ulcorner\theta\urcorner \in D \Longleftrightarrow T \vdash \neg\theta(\ulcorner\theta\urcorner)$$

よって T が完全だと仮定すると矛盾する. □

6.2 第二不完全性定理

補題 6.2.1 (不動点補題): 1 変数の論理式 θ について，次を満たす 文 $\mathrm{fixpoint}_\theta$ が存在する．

$$\mathsf{I}\Sigma_1 \vdash \mathrm{fixpoint}_\theta \leftrightarrow \theta(\ulcorner \mathrm{fixpoint}_\theta \urcorner)$$

以降 T を $\mathsf{I}\Sigma_1$ より強い理論とする． Gödel 文 G_T, T の無矛盾性を表す文 Con_T を定義する．

$$\mathrm{G}_T := \mathrm{fixpoint}_{\neg \mathrm{Provable}_T}$$
$$\mathrm{Con}_T := \neg \mathrm{Provable}_T(\ulcorner \bot \urcorner)$$

6.2 第二不完全性定理

適当な条件のもと G_T は T から独立であり， また Con_T と同値であることが証明できる．

補題 6.2.2:
1. T が無矛盾ならば $T \nvdash \mathrm{G}_T$,
2. $\mathbb{N} \models T$ ならば $T \nvdash \neg \mathrm{G}_T$.
3. $T \vdash \mathrm{Con}_T \leftrightarrow \mathrm{G}_T$

従って，

定理 6.2.1 (G2): T が無矛盾ならば $T \nvdash \mathrm{Con}_T$. $\mathbb{N} \models T$ ならば $T \nvdash \neg \mathrm{Con}_T$.

6.2 第二不完全性定理

よって以下が証明できる.

```
theorem goedel_second_incompleteness
    [IΣ₁ ≼ T] [T.Delta1Definable] [LO.System.Consistent T] :
    T ⊬ ↑Con
theorem inconsistent_undecidable
    [IΣ₁ ≼ T] [T.Delta1Definable] [ℕ ⊧ₘ* T] :
    System.Undecidable T ↑Con
```

ただし，上の証明が本質的に依存しているのは D1, D2, D3 と述語の健全性のみ.

6.3 証明可能性論理

以下の内容は私（齋藤）ではなく，主に神戸大の野口氏の形式化した結果である.

型クラスを用いて証明可能性述語や Hilbert-Bernays の可証性条件を抽象的に定義でき，これらの仮定の上で一般的に G1 や G2 が証明できる.

第一不完全性定理：

```
theorem goedel_independent
    [T ≼ U] [Diagonalization T] [LO.System.Consistent U]
    (𝔅 : ProvabilityPredicate T U) [𝔅.GoedelSound] :
    System.Undecidable U (goedel 𝔅)
```

第二不完全性定理：

```
theorem unprovable_consistency
    [T ≼ U] [Diagonalization T] [System.Consistent U]
    (𝔅 : ProvabilityPredicate T U) [𝔅.HBL] :
    U ⊬ con 𝔅
```

6.4 今後

- 算術的完全性定理.
- Paris-Harrington の定理等の独立命題に関する結果.
- 集合論.
- 二階算術.
- 直観主義一階述語論理, 特に Heyting arithmetic.
- Büchi arithmetic や S2S の決定性.
- 自動証明.

⋮

Bibliography

[1] L. C. Paulson, “A mechanised proof of Gödel’s incompleteness theorems using Nominal Isabelle,” Journal of Automated Reasoning, vol. 55, pp. 1–37, 2015.

[2] P. Hájek and P. Pudlák, Metamathematics of first-order arithmetic, vol. 3. Cambridge University Press, 2017.

[3] S. Cook and P. Nguyen, Logical foundations of proof complexity, vol. 11. Cambridge University Press Cambridge, 2010.

Sponsor

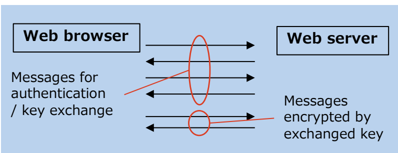
`FormalizedFormalLogic` is supported by Proxima Technology Proxima

Utilizing LLM Chatbots for Formal Descriptions of Cryptographic Protocols

Hideki Sakurada（NTT Communication Science Laboratories）
Kouichi Sakurai（Kyushu University）

Background：Cryptographic Protocols

Protocol Specification (Example)



Protocol Implementations

Web browsers
- Chrome
- FireFox
- Edge
- Safari

Web servers
- Apache
- Nginx

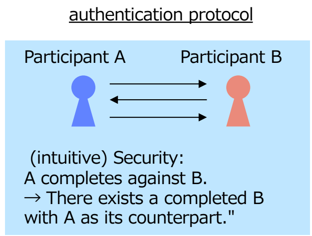
- Achieve security such as confidentiality, authentication, etc.
- By combining cryptographic techniques (digital signatures, public-key cryptography etc.)
- Even if cryptography is secure, the **combination in the wrong way can make it insecure.**

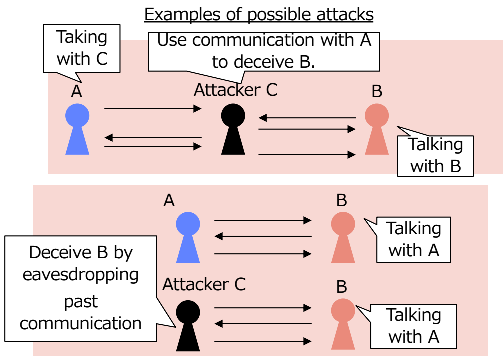
- Insecure if the protocol specification is insecure.

Background: Cryptographic Protocol Security Difficulties

NTT

Cryptographic protocols are used repeatedly by various participants, and attacks combining them are expected





Need to **consider all actions of the attacker**, **all combinations of participants**, and **arbitrary number of repetitive executions**

2

Background: Formal Verification

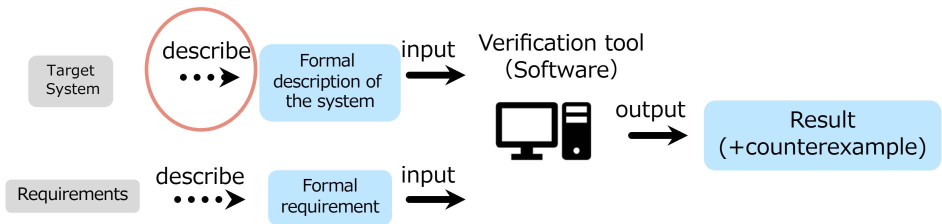
NTT

formal specification

Describe expected properties of a system in a logically defined language.

formal verification

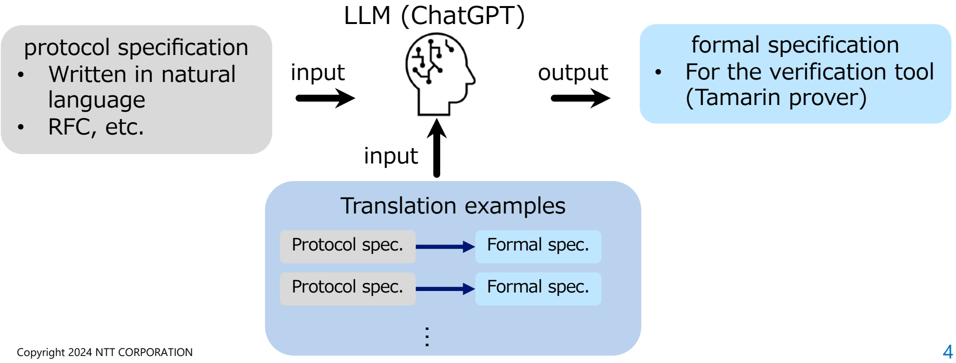
Rigorously verify that the developed system meets specifications.



3

Purpose of this work

Using large-language models (LLMs), how well can we generate formal specification of security protocols from protocol specification written in natural language?



 4

Input example① (Yahalom protocol)

(Protocol description in Wikipedia)

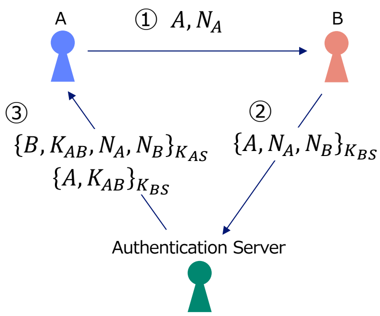
Yahalom is an authentication and secure key-sharing protocol designed for use on an insecure network such as the Internet. …

Protocol description
If Alice (A) initiates the communication to Bob (B) with S is a server trusted by both parties, the protocol can be specified as follows using security protocol notation:

- A and B are identities of Alice and Bob respectively
- Kas is a symmetric key known only to A and S

…

A->B: A, Na
 Alice sends a message to Bob requesting communication.
B->S: {A, Na, Nb}Kbs
 Bob sends a message to the Server encrypted under Kbs
S->A: {B, Kab, Na, Nb}Kas, {A, Kab}Kbs
 The Server sends to Alice a message containing the generated session key Kab and a message to be forwarded to Bob. …



 5

Expected Output Example① (Yahalom protocol)

NTT

```
theory Yahalom
begin

builtins: symmetric-encryption

rule Init:
  [ Fr(~kxs)]
  --[KeyGen($X)]->
  [ !LongtermKey(~kxs,$X) ]
…

rule A_1:
[Fr(~na)]  --[]->
  [Out(<$A, ~na>), StateA1($A, $B, ~na)]

rule B_1:
  let msg = senc(<'1', yA, yna, ~nb>,ykbs) in
  [In(<yA, yna>), !LongtermKey(ykbs,yB),
   Fr(~nb)]
  --[RunningB(yB,yA,<yna,~nb>)]->
  [Out(<yB, msg>), StateB1(yB, yA, yna, ~nb)]
…
```

Set of multiset rewriting rules

Common part for protocols using symmetric-key encryption

Protocol specific part: describing state transition and message exchanges of participants

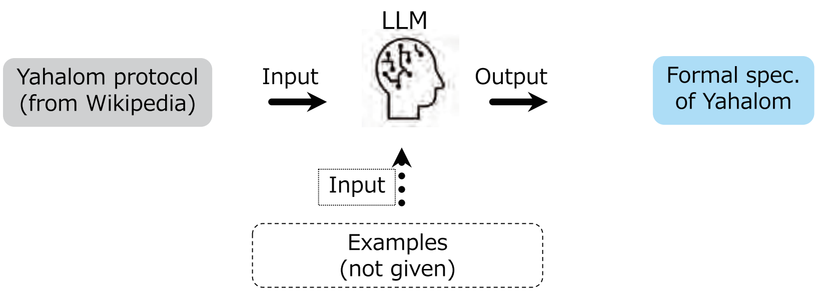
6

Experiment 1-1: Yahalom protocol

NTT

No translation examples are given, just the protocol description.

Prompt (query) for LLM (Chat GPT, GPT-4o) :
The following is an explanation of the Yahalom protocol. Write this protocol as a Tamarin Prover script



7

Output of Experiment 1-1

```
theory Yahalom begin

// Definitions of free variables
free variables
  A B S : agent // Alice, Bob, and the Server
  Kas Kbs Kab : symmetric_key // Symmetric keys
  Na Nb : nonce // Nonces

// Function symbols
Functions
  // Encryption and decryption
  senc/2: pubkey * message -> message
  sdec/2: pubkey * message -> message
  // Hashing (or other processing) functions can be defined as needed

// Rules for protocol steps
// Step 1: A -> B : A, Na
rule Alice_Send_Message:
  [ Fr(~Na) ]
  --[ Send(A, B, Na) ]-> [ State_A(A, B, ~Na) ]
...
```

Mixture of formal spec. for Proverif and Tamarin tools

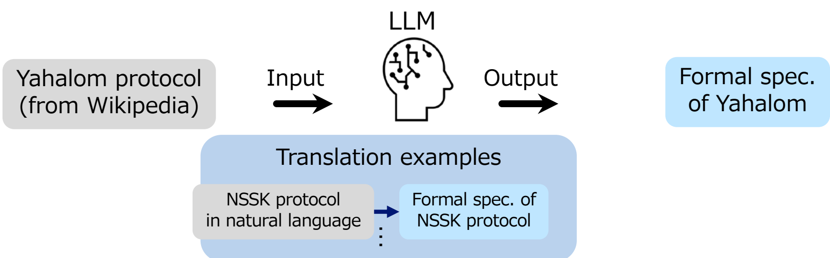
⬇

Not usable for verification, but suggesting GPT-4o has some knowledge for the tools' language

Experiment 1-2: Yahalom, with example

Translation of NSSK protocol given as an example

Prompt :
The following are the descriptions of **the Yahalom protocol, the Needham-Schroeder symmetric-key authentication protocol** written in natural language, and **the script for the Needham-Schroeder symmetric-key authentication protocol** for Tamarin Prover . Write a script for the Tamarin Prover for the Yahalom protocol, …



Output of Experiment 1-2

NTT

```
theory Yahalom
Begin
builtins: symmetric-encryption
…
// A and B already share a long-term symmetric key with S
rule Init_A:
  [ Fr(~kas) ] --[ KeyGen($A) ]-> [ !LongtermKey(~kas,$A) ]
…

// 1. A -> B : A, Na
rule A_to_B:
  [ Fr(~na) ] --[]-> [ Out(<'1', $A, ~na>) , StateA1($A, ~na) ]

// 2. B -> S : {A, Na, Nb}Kbs
rule B_to_S:
  let msg = senc(<'2', $A, ~na, ~nb>, ~kbs) in
  [ In(<'1', $A, ~na>) , !LongtermKey(~kbs, $B) , Fr(~nb) ]
  --[]->
  [ Out(msg) , StateB1($B, $A, ~kbs, ~na, ~nb) ]
…
```

- Grammatically almost correct.
- With some wellformedness error (Decrypting a cipher that cannot be decrypted)
- Semantically, the messages sent and received are correct, but the description of the participant's states is incorrect.

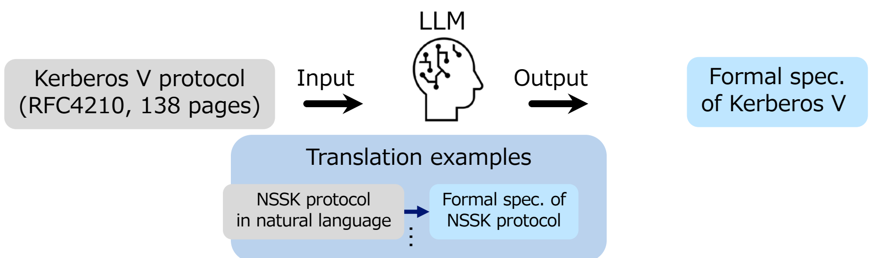
⬇

Although containing errors, grammatical errors are correctable with additional instructions.
Can it be used as a starting point for a description?

Experment 2: Kerberos V protocol

NTT

Prompt :
Below, I provide both the description of the Needham-Schroeder secret-key authentication protocol and the Tamarin Prover script for the Needham-Schroeder secret-key authentication protocol. Based on the Tamarin Prover script written from the description of the Needham-Schroeder secret-key authentication protocol, write a Tamarin Prover script for the Kerberos V protocol, as described in the attached text file.



Output of Experiment 2

- Grammar errors and wellformedness errors are the same as in Experiments 1-2.
- Messages generally followed the Kerberos V protocol, but some parts that should have been sent in plain text were encrypted; in the RFC, the description of this part is ambiguous in the natural language description, and the description of messages in the latter half of the RFC (in ASN.1) is plain text ⬇
- It is thought that GPT-4o attempts to convert faithfully to the input to some extent (?). but, (of course) the correctness of the result needs to be scrutinized.

Future Work

- Since protocols in this work are well-known (many references,) GPT-4o has knowledge about the target protocol
  ➡ What about a completely unknown protocol?
- How can I get more accurate output?
  ➡ rather than directly outputting Tamarin Prover scripts, better to output to an intermediate language suitable for conversion in LLM?
- How to check the correctness of the output?
  Grammar errors can be detected by Tamarin Prover.
  Based on the output, give instructions to correct it?

A Formalization of Prokhorov's Theorem in Isabelle/HOL

Michikazu Hirata

Institute of Science Tokyo

TPP2024,
November 26, 2024

Today's Talk

A Formalization of the Lévy-Prokhorov Metric in Isabelle/HOL, ITP2024.

The Lévy-Prokhorov Metric
= A metric between finite measures on a metric space.

- Lévy-Prokhorov metric
- Prokhorov's theorem
- The space of all finite measures is a Polish and standard Borel space

Today's Talk
Prokhorov's Theorem
(I did not talk the detail in the presentation at ITP2024)

What is Prokhorov's theorem?

Measure

A measure μ on X

$$\boxed{\mu : \Sigma_X \to [0, \infty]}$$

X : set, $\Sigma_X \subseteq 2^X$: σ-algebra on X.

Intuitively, $\mu(A) =$ the *size* of A
(finite measure $\iff \mu(X) < \infty$)

Ex.

The Lebesgue measure ν on $\mathbb{R}^n$	$\nu\left((a_i, b_i]^n\right) = (b_i - a_i)^n$ infinite measure ($\nu(\mathbb{R}^n) = \infty$)
A probability measure	$P(E) =$ probability E happens finite measure (P(sample space) $= 1$)

Weak Convergence

Let $\mathcal{P}(X) = \{$all finite measures on $X\}$.
$\neq$ power set

Topology of Weak Convergence

$\mathcal{O}_{\mathrm{WC}(X)}$ on $\mathcal{P}(X)$
$\mathcal{O}_{\mathrm{WC}(X)}$ = the coarsest topology making $(\lambda\mu.\ \int f \mathrm{d}\mu)$ continuous $\forall f \in \mathrm{C_b}(X)$.

$\mathrm{C_b}(X) = \{f : X \to \mathbb{R},\ f$ is bounded continuous$\}$

$$\begin{array}{rcccc} (\lambda\mu.\int f\mathrm{d}\mu) : & \mathcal{P}(X) & \to & \mathbb{R} \\ & \Cup & & \Cup \\ & \mu & \mapsto & \int f\mathrm{d}\mu \end{array}$$

Fact. X is separable metrizable $\Longrightarrow$ So is $\mathcal{P}(X)$
X is Polish $\Longrightarrow$ So is $\mathcal{P}(X)$

Weak Convergence

Weak Convergence

$$\mu_n \Rightarrow_{\mathrm{wc}} \mu \overset{\text{def}}{\iff} \mu_n \longrightarrow \mu \text{ in } \mathcal{P}(X)$$

$$\mu_n \Rightarrow_{\mathrm{wc}} \mu \iff \forall f \in \mathrm{C_b}(X).\ \int f\mathrm{d}\mu_n \longrightarrow \int f\mathrm{d}\mu$$

Ex. The central limit theorem

$X_n \cdots$ i.i.d. samples,
$P_n \cdots$ the distribution of normalized sample mean of $X_0, \ldots, X_n$,

Under appropriate conditions, $P_n \Rightarrow_{\mathrm{wc}} \mathrm{Normal}(0, 1)$

Prokhorov's Theorem

$\mathcal{P}_r(X) = \mathcal{P}(X) \cap \{\mu.\ \mu(X) \leq r\}$ for some $r < \infty$

Prokhorov's Theorem

X: a Polish space,
$\Gamma \subseteq \mathcal{P}_r(X)$

$\overline{\Gamma}$ is compact in $\mathcal{P}(X) \iff \Gamma$ is tight: i.e.,

$\forall \varepsilon > 0.\ \exists K$: compact in X, s.t.$\forall \mu \in \Gamma.\ \mu(X - K) \leq \varepsilon$

Fact For a metrizable space X and $A \subseteq X$,

$$A \text{ is compact} \iff \forall \{x_n\}_{n\in\mathbb{N}} \subseteq A.\ \exists n_k.\ \exists x \in A.\ \text{s.t. } x_{n_k} \longrightarrow x$$

Obtain a weak converging subsequence!

Corollary

If X is separable and metrizable, $\{\mu_n\}_{n\in\mathbb{N}} \subseteq \mathcal{P}_r(X)$,
and $\{\mu_n\}_{n\in\mathbb{N}}$ is tight.

Then, $\exists \{\mu_{n_k}\}_{k\in\mathbb{N}}$: subsequence and μ s.t. $\mu_{n_k} \Rightarrow_{\mathrm{wc}} \mu$.

Prokhorov's Theorem

$\mathcal{P}_r(X) = \mathcal{P}(X) \cap \{\mu.\ \mu(X) \leq r\}$ for some $r < \infty$

Prokhorov's Theorem

X: a Polish space,
$\Gamma \subseteq \mathcal{P}_r(X)$

$\overline{\Gamma}$ is compact in $\mathcal{P}(X) \iff \Gamma$ is tight: i.e.,

$\forall \varepsilon > 0.\ \exists K$: compact in X, s.t.$\forall \mu \in \Gamma.\ \mu(X - K) \leq \varepsilon$

Used for

- Completeness of the Lévy-Prokhorov metric
- Central limit theorem
- Sanov's theorem in large deviation theory
- Existence of optimal coupling in transportation theory

depending on

- Riesz representation theorem
- Alaoglu's theorem

Summary

Formalization in Isabelle/HOL

- Prokhorov's theorem
- Riesz representation theorem (Tough! 2.1k– LOC)
- (A special case of) Alaoglu's theorem

Archive of formal proofs

- *The Lévy-Prokhorov Metric*, June 2024 (6.6K lines)
  - the special case of Alaoglu's theorem
  - Prokhorov's theorem
- *The Riesz representation theorem*, June 2024 (4.4K lines)

Prokhorov's Theorem

A key lemma for the proof of Prokhorov's Theorem

If X is a compact metric space, then $\mathcal{P}_r(X)$ is compact.

Idea: $\mathcal{P}_r(X) \cong \langle$a compact space$\rangle$

$$\begin{array}{ccc} T : \mathcal{P}_r(X) & \to & \mathbb{R}^{C(X)} \cap \{\varphi.\ \varphi \text{ is positive linear } \wedge \varphi(1) \leq r\} \quad (=: \Phi) \\ \Psi & & \Psi \\ \mu & \mapsto & (\lambda f.\ \int f \mathrm{d}\mu) \end{array}$$

Linearity $T(\mu)(f+g) = \int f + g \, \mathrm{d}\mu = \int f \mathrm{d}\mu + \int g \mathrm{d}\mu = T(\mu)(f) + T(\mu)(g)$

- Inverse function?
  $\Longrightarrow$ The Riesz representation theorem
- Compactness of Φ?
  $\Longrightarrow$ Alaoglu's theorem

Riesz Representation Theorem

The $\begin{cases} \text{Riesz} \\ \text{Riesz-Markov} \\ \text{Riesz-Markov-Kakutani} \end{cases}$ representation theorem.

Riesz Representation Theorem

X, a locally compact Hausdorff space
$\mathrm{C_C}(X)$, the set of continuous functions which have closed compact supports

$\varphi : \mathrm{C_C}(X) \to \mathbb{R}$, a positive linear functional
i.e., $\varphi(\alpha f + \beta g) = \alpha\varphi(f) + \beta\varphi(g)$ and $\varphi(f) \geq 0$ if $f \geq 0$.

Then, there exists $\mathcal{M} \supseteq \sigma[\mathcal{O}_X]$ and a unique measure μ on $\mathcal{M}$ s.t.

$$\forall f \in \mathrm{C_C}(X).\ \varphi(f) = \int f \mathrm{d}\mu \quad + 5 \text{ conditions}$$

Rudin's book

- 9 pages including lemmas (e.g. Urysohns' lemma)

Formal proof

- 2.1k+ lines

Set-Based Vector Space

$\varphi : \mathrm{C_C}(X) \to \mathbb{R}$, a positive linear functional

$\mathrm{C_C}(X)$: a vector space

Vector space in Isabelle/HOL

- Type-based (**class**) by [Hölzl+, ITP2013]
  - carrier sets must be *UNIV* (all elements of the type)
    Ex. Vector spaces on
    ☺ $\mathbb{C}$
    ☹ $\mathrm{C_C}(X)$
- Set-based (**locale**, **typedef**)
  - any carrier sets
  - has only basic definitions ([Lee, AFP2014])

My choice

- do not use vector space library
- write down conditions directly

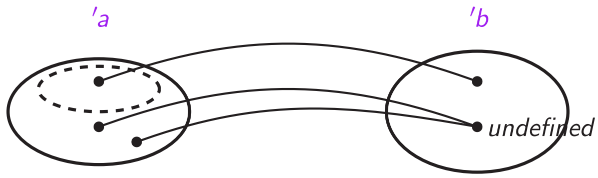
Riesz Representation Theorem in Isabelle/HOL

$\varphi : C_C(X) \to \mathbb{R}$, a positive linear functional

definition *positive-linear-functional-on-CX* ::
'a topology $\Rightarrow$ ((*'a* $\Rightarrow$ *'b* :: {*ring*, *order*, *topological-space*}) $\Rightarrow$ *'b*) $\Rightarrow$ *bool*

where *positive-linear-functional-on-CX X* $\varphi \equiv$
($\forall$ *f. continuous-map X euclidean f* $\longrightarrow$ *f has-compact-support-on X*
$\longrightarrow$ ($\forall$ *x*$\in$*topspace X. f x* $\geq$ *0*) $\longrightarrow$ φ (λ*x*$\in$*topspace X. f x*) $\geq$ *0*) $\wedge$
+ *linearity*

- *'b* :: {*ring*, *order*, *topological-space*}
  for *real* and *complex*
- ($\lambda x \in$ *topspace X. f x*) $y = \begin{cases} f\ y & \text{if } y \in \textit{topspace } X \\ \textit{undefined} & \text{o.w.} \end{cases}$



Alaoglu's Theorem

Y : a normed vector space
Y^* : the dual space
weak* topology on Y^* : the coarsest topology making all $(\lambda f.\, f(y)) : Y^* \to \mathbb{R}$ continuous

Alaoglu's Theorem

Let $B^* = \{\varphi \in Y^* \mid \|\varphi\| \leq r\}$, then B^* is compact in Y^* w.r.t. weak* topology.

Set-based vector spaces have neither dual space nor norm.

My choice Prove the spacial case for Prokhorov's theorem

Special Case of Alaoglu's Theorem

If X is compact, then $\mathbb{R}^{C(X)} \cap \{\varphi.\, \varphi \text{ is positive linear } \wedge \varphi(1) \leq r\}$ is compact.

$\|\varphi\| = \varphi(1)$ if φ is positive linear.

Alaoglu's Theorem in Isabelle/HOL

Special Case of Alaoglu's Theorem

If X is compact, then $\mathbb{R}^{C(X)} \cap \{\varphi.\ \varphi$ is positive linear $\wedge\ \varphi(1) \leq r\}$ is compact.

theorem *Alaoglu-theorem-real-functional*:
  fixes X :: *'a topology* **and** r :: *real*
  defines *prod-space* $\equiv \mathbb{R}^{C(X)}$
  defines $B \equiv \{\varphi \in$ *topspace prod-space*.
    φ $(\lambda x \in$ *topspace X. 1*$) \leq r \wedge$ *positive-linear-functional-on-CX X* $\varphi\}$
  assumes *compact-space X* **and** *topspace X* $\neq \{\}$
  shows *compactin prod-space B*

Related Works

In Isabelle/HOL by Avigad et al. (2017)

A special case of Prokhorov's theorem for the central limit theorem

- The special case
  $\Longrightarrow$ a simpler proof
- General case
  $\Longrightarrow$ needs Riesz representation, Alaoglu's theorem

In Lean

`RieszMarkovKakutani.lean`

> *This file will prove different versions of the Riesz-Markov-Kakutani representation theorem. ...*

I could not find the final statements. It seems still ongoing.

Conclusion

Formalization in Isabelle/HOL

- Prokhorov's theorem
- The Riesz representation theorem
- A special case of Alaoglu's theorem

Reference

- Prokhorov's theorem

  Onno van Gaans, *Probability measures on metric spaces*,
  https://www.math.leidenuniv.nl/~vangaans/jancol1.pdf

- The Riesz representation theorem

  Rudin, Walter, *Real and Complex Analysis, 3rd Ed*, 1987

- Alaoglu's theorem

  Christopher E. Heil, *Alaoglu's Theorem*,
  https://heil.math.gatech.edu/6338/summer08/section9f.pdf

Monadic equational reasoning for general recursive function

Ryuji Kawakami(Nagoya University)

Implementation of Delay monad in monae

Example

Combination with other monads via monad transformer

Related work

Implementation of Delay monad in monae

Monad and side effect

- Monad
  - The type class which has operators return and bind.
  - Monads are used to express side effect, such as I/O, exception, stateful computation and so on.

Exception monad

- Monadic structure of Exception monad.

```
A B : Type
Definition option := fun A : Type => unit + A.
Instance Exception  : Monad option :={
   ret := inr ;
   >>= := fun (m : option  A) (f : A -> option  B) => match m with
                                                           |inl z => inl z
                                                           |inr b => f b end}.
```

- We can define "division with error".

```
Definition safedivide(n m: nat): option nat := match m with
                                                 |O => inl tt (* error *)
                                                 |S m' => ret (n./m)
                                               end.
```

- The monadic structure allows us to combine functions which might get an error.

```
Definition safedivide3 (n m1 m2 m3: nat): option nat :=
    (safedivide n m1) >>= (fun x => safedivide x m2) >>= (fun x => safedivide x m3).
```

2

Monae: Monadic equational reasoning in COQ [ANS19]

- Monadic equational reasoning [GH11]:Equational theory for monadic programs.
- Interfaces for equational reasoning and models for soundness.
- It already supports state monad,probability monad,typed-store monad and so on.

interface for exception monad

```
A B: Type
```

- Operators for exception monad.

```
fail : M A
catch : M A -> M A -> M A
```

- Laws for catch and fail.

```
catchfailm : forall h, catch fail h = h
catchmfail : forall m, catch m fail = fail
catchmfail : forall m, catch m fail = fail
catchA : forall m h k, catch m (catch h k)  = (catch (catch m h) k)
catchret : forall x, catch (ret x) h = ret x
```

3

Monadic equational reasoning

- Equational reasoning for fast product.

```
Let fail A : M A:= inl tt.
Let catch A (m m': M A):M A := m >>= (fun _ => m').
Let work A (l: list nat) : M nat := if 0 in l then fail else ret (product l).
Let fastproduct A (l: list nat) : M nat := catch (work l) (ret 0).
```

```
 fastproduct s
《 definition of fastproduct 》
 = catch (work s) (ret 0)
《 definition of work 》
 = catch (if 0 in s then fail else ret (product s))
《 equations for if then else 》
= if 0 in s then catch fail (ret 0) else catch (ret (product s)) (ret 0)
《 catchfailm 》
 = if 0 in s then ret 0 else catch (ret (product s)) (ret 0)
《 catchret 》
 = if 0 in s then ret 0 else ret (product s)
《 0 in s => product s  = 0 》
 = if 0 in s then ret (product s) else ret (product s)
《 equations for if then else 》
 = ret (product s)
```

4

課題

- Limitation of Coq:
  To protect consistency, we cannot define a function without proof of the termination.

```
Fixpoint collatz (n:nat): bool :=
  if n == 1 then true
  else if (n %%2 == 0)
       then collatz (n./2)
       else collatz (3*n + 1).
```

$\Rightarrow$

```
Error:
Cannot guess decreasing argument of fix.
```

- Supporting general recursive functions in essential to make monae practical.
- General recursive functions are expressed by Delay Monad [Cap05].

Goal

- Implementing the Delay monad in monae.
- Reasoning the properties of general recursive functions based on the equational theory.

5

Delay Monad

- Delay monad consists of Delay: (A:Type) $\rightarrow$ (cofixpoint of X = A + X).

```
CoInductive Delay (A : Type) : Type :=
  |DNow : A -> Delay A
  |DLater : Delay A -> Delay A.
```

- The data of Delay A consisits of
  - DNow a
  - DLater (DLater(DLater ⋯ (DNow a)))
  - ⋯ (DLater (DNow a)))) ⋯

- The DLater constructor expresses a step of computation.

- Monadic structure

```
Let ret (a:A) := DNow a
CoFixpoint bind (m: Delay A) (f: A -> Delay B ) :=
  match m with
  |DNow a => f a
  |DLater d => DLater (bind d f)
  end.
```

6

Weak bisimilarity

```
CoFixpoint while (body: A -> M(B + A)) :A -> M B :=
  fun a =>  (body a >>= (fun ab => match ab with
                                  |inr a => DLater (while body a)
                                  |inl b => DNow b end) ).
Let countup := while (fun n => if n < 5
                                then DNow (inr (n+1))
                                else DNow (inl n)).
```

- We want to judge the equality of computations.
- This relation is a weak bisimulation ≈ on Delay A using Coinductive type.

```
Error :countup 0  = DLater(DLater(DLater(DLater(DLater(DNow 5))))) ≠  DNow 5

Ok :countup 0  ≈  DNow 5
```

7

interface for delay monad

```
Definition sum_rect (f: A -> C) (g: B -> C) :=
  fun ab => match ab with |inl a => f a |inr b => g b end.
```

- Operators for delay monad

```
while : (A -> M (B + A)) -> A ->M B;
```

- weak bisimulation is a equivalence relation

```
≈: M A -> M A -> bool;
wBisim_refl:forall a, a ≈ a;
wBisim_sym: forall (a1 a2: A) a1 ≈ a2 -> a2 ≈ a1;
wBisim_trans: forall (a1 a2 a3: A) a1 ≈ a2 -> a2 ≈ a3 -> a1 ≈ a3;
```

- Laws for operator while, analogy to equations of Complete elgot monad [AMV10]

```
fixpointE: forall (f: A -> M (B + A)) (a: A),
while f a ≈ (f a) >>= (sum_rect (@ret M B) (while f))

naturalityE: forall (f: A -> M (B + A)) (g: B -> M C) (a: A),
(while f a) >>= g
≈ while (fun y => (f y) >>= (sum_rect (M # inl o g) (M # inr o (@ret M A )) ) )

codiagonalE:forall (f: A -> M ((B + A) + A)) (a: A),
while ((M # ((sum_rect idfun inr))) o f ) a ≈ while (while f)

bindmwB: (f: A -> M B)(d1 d2: M A), wBisim d1 d2 -> wBisim (d1 >>= f) (d2 >>= f)

bindfwB: (f g: A -> M B)(d: M A),
(forall a, wBisim (f a) (g a)) -> wBisim (d >>= f) (d >>= g)

whilewB: forall (A B : UU0) (f g: A -> M ((B + A))) (a: A),
 (forall a, wBisim (f a) (g a)) -> wBisim (while f a) (while g a);
```

8

Example

McCarthy 91 function

- McCarthy 91 returns 91 for all integer argument $n \leq 101$, which is defined as follows.

```
let rec mc91 n = if 100 < n then n-10 else mc91 (mc91 (n+11))
```

- The complex recursion causes it difficult to define the function in Coq.

Calculation of mc91

```
mc91(98)
= mc91(mc91(109))
= mc91(99)
= mc91(mc91(110))
= mc91(100)
= mc91(mc91(111))
= mc91(101)
= 91
```

- we defined the function using while operator and proved $n \leq 101 \implies mc91(n) = 91$.

9

Definition of mc91 in Monae

- n : the depth of the continuation, m:value

```
int mc91 (int n ,int m) {
    while (n != 0) {
        if (m > 100) {
            n -= 1;
            m -= 10;
          } else {
            n += 1;
            m += 11;
          }
    }
    return m;}
```

$\Longrightarrow$

```
Let mc91_body nm :=
    match nm with (n, m) =>
    if n==0
    then ret (inl m)
    else if m > 100
         then ret (inr(n.-1,m-10))
         else ret (inr(n.+1,m+11))
    end.

Let mc91 n m := while mc91_body (n.+1,m).
```


The proof in Monae

- we proved if $90 \leq m \leq 100$, then `mc91 m = mc91 (m+1)`.
- `mc91 n = 91` is proven from the lemma, `mc91 101 = 91`, and induction on $k = 90 - m$.

```
Goal: forall n m, 90 <= m < 101 -> mc91 n m ≈ mc91 n (m.+1).
mc91 n m
《 definition of mc91 》
≈ while (fun nm : nat * nat =>
let (n0, m0) := nm in if n0 == 0 then Ret (inl m0)
                                  else if 100 < m0
                                       then Ret (inr (n0.-1, m0 - 10))
                                       else Ret (inr (n0.+1, m0 + 11))) (n.+1, m)
《 fixpointE 》
≈ (if 100 < m then Ret (inr (n, m - 10))
            else Ret (inr (n.+2, m + 11))) >>= sum_rect (fun=> M nat) Ret (while mc91_body)
《 m < 101 》
≈ Ret (inr (n.+2, m + 11)) >>= sum_rect (fun=> M nat) Ret (while mc91_body)
《 Monad law: Ret a >>= f = f a 》
≈ while mc91_body (n.+2, m + 1)
《 definition of mc91,fixpointE 》
≈ (if 100 < m + 11
   then Ret (inr (n.+1, m + 11 - 10))
   else Ret (inr (n.+3, m + 11 + 11))) >>= sum_rect (fun=> M nat) Ret (while mc91_body)
《 90 ≤ m ⟹ 100 < m + 11, Monad law 》
≈ while mc91_body (n.+1, m + 11 - 10) = mc91 n (m+1)
```

Combination with other monads via monad transformer

Monad transformer

Monad transformer MT

MT consists of 4 transformer set $(\mathsf{T}, \mathrm{return}^{\mathrm{T}}, \mathrm{bind}^{\mathrm{T}}, \mathrm{lift}^{\mathrm{T}})$ such that for any monad M

- $(\mathsf{TM}, \mathrm{return}^{\mathrm{T}}(\mathrm{ret}, \mathrm{bind}), \mathrm{bind}^{\mathrm{T}}(\mathrm{return}, \mathrm{bind}))$ is a monad
- $\mathsf{lift}^{\mathsf{T}}(\mathrm{return}, \mathrm{bind})$ is a monad morphism from M to TM

- We can create new monad keeping original monad structure.
- Monae already supports monad transformers [AN21].

Combination via state monad transformer stateT

stateT

- It combines side effects about state.

```
Definition MS := fun M: Monad => fun A : Type => S -> M (A * S).
Instance stateM : Monad (MS M):={
   retS := fun a => Ret (a, s) ;
   bindS :=
     fun (m : MS A) (f : A -> MS B) =>
       fun (s: S) => m s >>= (fun (a: A) => f a s) }
```

- I proves the following definition is a instance of Delay monad in Coq based on [PG13].

```
M:delayMonad
Definition dist1 {X Y} (s:S*(Y + X)) :(S*Y) + (S*X) :=
let (yx, s) := s in match yx with |inl y => inl (y,s) | inr x => inr (x,s) end.
Definition DS := MS M
Definition whileDS {X Y} (body: X -> DS (Y + X)) :=
  curry (while (M # dist1 o uncurry body)).

Definition wBisimDS {A} (ds1 ds2:DS A): Prop :=
  forall s:S, wBisim (ds1 s) (ds2 s).
```

13

Combination via except monad transformer exceptT

exceptT

- It combines side effects about exception.

```
Definition MX := fun M:Monad => fun X : Type => M (unit + X).
Instance exceptM  : Monad (MX M) :={
   retX := fun x => Ret (inr x) ;
   bindX :=
     fun (t : MX X) (f: X -> MX Y) =>
     t >>= fun c => match c with inl z => Ret (inl z) | inr x => f x end}
```

- We prove the following definition is a instance of Delay monad in Coq.

```
M: delayMonad
Definition DE := MX unit M.
Definition DEA {A B} :DE (A + B) -> M ((unit + A) + B ) :=
M # (fun uab => match uab with
|inl u => inl (inl u)
|inr ab => match ab with inl a => inl (inr a) |inr b => inr b end
end).
Definition whileDE {A B} (body: A -> DE (B + A)): DE B := while (DEA o body)
Definition wBisimDE {A} (d1 d2: DE A) := wBisim d1 d2.
```

14

Combination via typed store monad transformer

- Monae supports it in [AGS23] to express ML reference.
- For example, it can express linked list.

typed store monad transformer

- It is defined by composition of stateT and exceptT.
- List of records which have type and value, seq binding, expresses typed store.
- It has basic operators cnew, cget and cput for reference.

```
Record binding :=
mkbind { bind_type : ml_type; bind_val : coq_type bind_type }.

Definition DTS : UU0 -> UU0 := MS (seq binding) (MX unit M0).
```

- It keeps the delay monad structure from the results of exceptT and stateT.

15

Example

- Programs which have iterations and references are expressed using delaytypedStoreMonad.

```
let fact n =
let r = ref 1 in
let l = ref 1 in
while !l <= n do
  r := !r * !l;
  l := !l + 1;
done;
!r
```

$\Longrightarrow$

```
Definition factdts (n: nat):=
do r <- cnew ml_int 1;
do l <- cnew ml_int 1;
do _ <-
     while (fun ( _:unit) =>
       do v <- cget r;
       do i <- cget l;
       if i <= n then do _ <- cput r (i*v);
                      do _ <- cput l (i.+1);
                      Ret (inr tt )
                 else Ret (inl v)) tt;
do v <- cget r; Ret v.
```

16

Related work

Related work

- Interaction tree [XZH+19]is also constructed to model general recursive function using coinductive type
  - It also treats various effects as events.
  - The equivalence of interaction trees is defined by generalized form of bisimulation.
- Piróg and Gibbon [PG13], and Sergey, Christoph, and Lutz [GSRJ18] shows the class of Elgot monads is stable under some transformations.
  - [PG13] shows Elgot monad structure is preserved under guarded coinductive generalized resumption transformer.
  - [GSRJ18] removed the guard condition of [PG13]
- Simpson and Plotkin [SP00] have studied complete laws for recursion in algebraic theory.
  - [SP00] have proved the completeness of axioms for iteration theory in [BÉ93].

17

Reference i

[AGS23] Reynald Affeldt, Jacques Garrigue, and Takafumi Saikawa.
A practical formalization of monadic equational reasoning in dependent-type theory, 2023.

[AMV10] Jiří Adámek, Stefan Milius, and Jiří Velebil.
Equational properties of iterative monads.
Information and Computation, 208(12):1306–1348, 2010.
Special Issue: International Workshop on Coalgebraic Methods in Computer Science (CMCS 2008).

[AN21] Reynald Affeldt and David Nowak.
Extending Equational Monadic Reasoning with Monad Transformers.
In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, 26th International Conference on Types for Proofs and Programs (TYPES 2020), volume 188 of Leibniz International Proceedings in Informatics (LIPIcs), pages 2:1–2:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

18

Reference ii

[ANS19] Reynald Affeldt, David Nowak, and Takafumi Saikawa.
A hierarchy of monadic effects for program verification using equational reasoning.
In Graham Hutton, editor, Mathematics of Program Construction, pages 226–254, Cham, 2019. Springer International Publishing.

[BÉ93] Stephen L. Bloom and Zoltán Ésik.
Iteration Theories, pages 159–213.
Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[Cap05] Venanzio Capretta.
General Recursion via Coinductive Types.
Logical Methods in Computer Science, Volume 1, Issue 2, July 2005.

[GH11] Jeremy Gibbons and Ralf Hinze.
Just do it: simple monadic equational reasoning.
In Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming, ICFP '11, page 2–14, New York, NY, USA, 2011. Association for Computing Machinery.

19

Reference iii

[GSRJ18] Sergey Goncharov, Lutz Schröder, Christoph Rauch, and Julian Jakob.
Unguarded Recursion on Coinductive Resumptions.
Logical Methods in Computer Science, Volume 14, Issue 3, August 2018.

[PG13] Maciej Piróg and Jeremy Gibbons.
Monads for behaviour.
Electronic Notes in Theoretical Computer Science, 298:309–324, 2013.
Proceedings of the Twenty-ninth Conference on the Mathematical Foundations of Programming Semantics, MFPS XXIX.

[SP00] Alex Simpson and Gordon Plotkin.
Complete axioms for categorical fixed-point operators.
In Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science, LICS '00, page 30, USA, 2000. IEEE Computer Society.

[XZH+19] Li-yao Xia, Yannick Zakowski, Paul He, Chung-Kil Hur, Gregory Malecha, Benjamin C. Pierce, and Steve Zdancewic.
Interaction trees: representing recursive and impure programs in coq.
Proc. ACM Program. Lang., 4(POPL), December 2019.

ON REPRESENTABILITY OF MULTIPLE-VALUED FUNCTIONS BY LINEAR LAMBDA TERMS TYPED WITH SECOND-ORDER POLYMORPHIC TYPE SYSTEM

SATOSHI MATSUOKA

Abstract. We show that any multiple-valued function can be represented by a linear lambda term that is typed in a second-order polymorphic type system.

1. Introduction

In [Mat16] besides the main result of the paper, the author showed that any two-valued function with any number of arguments can be represented by a linear lambda term that is typed with second polymorphic type system, where the base type is $\mathtt{T}_2 = \forall\mathtt{'a.('a->'a)->('a->'a)->('a->'a)}$. In this article, extending that result, we show that any multiple-valued function with any number of arguments can be represented, where the base type is

$$\mathtt{T}_\mathtt{r} = \forall\mathtt{'a.}\, \overbrace{\mathtt{('a->'a)->}\cdots\mathtt{->('a->'a)}}^{r}\mathtt{->('a->'a)}$$

when r is the number of arguments of a given multiple-valued function.

2. Second-order Polymorphic Linear Type System

In this section we present a second-order polymorphic linear type system. This system is the linear type system in [Mat16] augmented with second-order quantifier.

Untyped terms we consider are as follows:

$$\mathtt{t} ::= \mathtt{x} \mid \mathtt{t\ t} \mid \mathtt{fn\ x\ =>\ t} \mid \mathtt{(t,t)} \mid \mathtt{let\ val\ (x,y) = t\ in\ t\ end}$$

Such a term is not necessarily a linear term which we want to discuss: we only consider linear terms that are typed in the second-order polymorphic linear type system described in the following.

The definition of types is as follows:

$$\mathtt{A} ::= \mathtt{'a} \mid \mathtt{A*A} \mid \mathtt{A->A} \mid \forall\mathtt{'a.A->A}$$

where `'a` is an atomic type.

A typing environment denoted by Γ or Δ is a list of pairs of a term variable and a type, $\mathtt{x1} : \mathtt{A1}, \ldots, \mathtt{xn} : \mathtt{An}$, where $n \geq 0$.

A typing judgement is a triple of typing environment, term `t` and type `A` denoted by $\Gamma \vdash \mathtt{t} : \mathtt{A}$.

The set of free type variables in type `A` denoted by FTV(`A`) is defined inductively:

$$\begin{aligned} \mathrm{FTV}(\mathtt{'a}) &= \{\mathtt{'a}\} \\ \mathrm{FTV}(\mathtt{A1*A2}) &= \mathrm{FTV}(\mathtt{A1->A2}) = \mathrm{FTV}(\mathtt{A1}) \cup \mathrm{FTV}(\mathtt{A2}) \\ \mathrm{FTV}(\forall\mathtt{'a.A}) &= \mathrm{FTV}(\mathtt{A}) \backslash \{\mathtt{'a}\} \end{aligned}$$

The set of free type variables in typing environment Γ is also defined:

$$\begin{aligned} \mathrm{FTV}() &= \emptyset \\ \mathrm{FTV}(\mathtt{x} : \mathtt{A}, \Gamma) &= \mathrm{FTV}(\mathtt{A}) \cup \mathrm{FTV}(\Gamma) \end{aligned}$$

Our second-order linear type system is as follows:

$$\frac{}{\mathtt{x:A} \vdash \mathtt{x:A}} \qquad \frac{\Gamma, \mathtt{x:A}, \mathtt{y:B}, \Delta \vdash \mathtt{t:C}}{\Gamma, \mathtt{y:B}, \mathtt{x:A}, \Delta \vdash \mathtt{t:C}}$$

$$\frac{\mathtt{x:A}, \Gamma \vdash \mathtt{t:B}}{\Gamma \vdash \mathtt{fn\ x=>t:A->B}} \qquad \frac{\Gamma \vdash \mathtt{t : A->B} \quad \Delta \vdash \mathtt{s:A}}{\Gamma, \Delta \vdash \mathtt{t\,s:B}}$$

$$\frac{\Gamma \vdash \mathtt{s:A} \quad \Delta \vdash \mathtt{t:B}}{\Gamma, \Delta \vdash \mathtt{(s,t):A*B}} \qquad \frac{\Gamma \vdash \mathtt{s:A*B} \quad \mathtt{x:A}, \mathtt{y:B}, \Delta \vdash \mathtt{t:C}}{\Gamma, \Delta \vdash \mathtt{let\ val\ (x,y)=s\ in\ t\ end:C}}$$

$$\frac{\Gamma \vdash \mathtt{s:\forall 'a.A}}{\Gamma \vdash \mathtt{s:A[B/'a]}} \qquad \frac{\Gamma \vdash \mathtt{s:A}}{\Gamma \vdash \mathtt{s:\forall 'a.A}} \ (\mathtt{'a} \notin \mathrm{FTV}(\Gamma))$$

In the next section, we only consider linear terms s where typed with the form

$$\vdash \mathtt{s:A}$$

in our type system. Such a term is usually called a *combinator*.

Since we only consider of typability of linear lambda terms in this paper, we omit equality and reduction rules on terms.

3. Representation of r-valued Functions with Any Number of Arguments

In the following, the notation

$$\mathtt{fun\ f\ x_1\ \cdots\ x_n = g}$$

means a term

$$\mathtt{fn\ x_1=> \cdots => fn\ x_n=>\ g}$$

named by `f`.

The base type for our representation is the following polymorphic type:

$$\mathtt{T_r} = \forall \mathtt{'a.}\ \overbrace{\mathtt{('a->'a)->\cdots->('a->'a)}}^{r}\mathtt{->('a->'a)}$$

The number of normal terms of $\mathtt{T_r}$ is $r!$. But any choice of r normal terms from $\mathtt{T_r}$ may fail for a correct representation. Although other choices are possible, we choose the following r terms which are considered as r powers of a cyclic permutation with length r:

$$\begin{aligned} \mathtt{fun\ v\_0\ f_{r-1}\ f_{r-2} \cdots f_0\ x} &= \mathtt{f_0(f_1(\cdots(f_{r-1}\ x)\cdots))} \\ \mathtt{fun\ v\_1\ f_{r-1}\ f_{r-2} \cdots f_0\ x} &= \mathtt{f_1(f_2(\cdots(f_0\ x)\cdots))} \\ &\cdots \\ \mathtt{fun\ v\_r-1\ f_{r-1}\ f_{r-2} \cdots f_0\ x} &= \mathtt{f_{r-1}(f_0(\cdots(f_{r-2}\ x)\cdots))} \end{aligned}$$

We need a few auxiliary terms. The following term is the standard `I` combinator:

$$\mathtt{fun\ I\ x = x}$$

The following term is a specialized version of the identity term:

$$\mathtt{fun\ id\ h\ f_{r-1}\ f_{r-2} \cdots f_0\ x = h\ f_{r-1}\ f_{r-2} \cdots f_0\ x}$$

The following term represents a one variable constant function:

$$\mathtt{fun\ const\_i\ h\ f_{r-1}\ f_{r-2} \cdots f_0\ x = f_i(f_{i+1}(\cdots(f_{i-1}\ (h\ \overbrace{I \cdots I}^{r}\ x))\cdots))}$$

The following term represents a higher order function that receives function $F : \{0,\ldots,r-1\} \to \{0,\ldots,r-1\}$ and a value $v \in \{0,\ldots,r-1\}$ and always returns a fixed value $i \in \{0,\ldots,r-1\}$:

$$\texttt{fun const\_f\_i F h } \mathtt{f}_{k-1}\ \mathtt{f}_{k-2} \cdots \mathtt{f}_0\ \mathtt{x} = \mathtt{f_i}(\mathtt{f_{i+1}}(\cdots(\mathtt{f_{i-1}}\ (\mathtt{h}\overbrace{\mathtt{I}\cdots\mathtt{I}}^{k}(\mathtt{F\ v\_0}\ \overbrace{\mathtt{I}\cdots\mathtt{I}}^{k}\ \mathtt{x})))\cdots))$$

Using some of these r higher-order functions we can represent any one variable function $g : \{0,1,\ldots,r-1\} \to \{0,1,\ldots,r-1\}$:

$$\texttt{fun g h} = \texttt{h const\_f\_g}_{k-1}\ \texttt{const\_f\_g}_{k-2}\ \cdots\ \texttt{const\_f\_g}_0\ \texttt{id v\_0}$$

This term g can be typed with $\mathtt{T_r}$->$\mathtt{T_r}$.

Thus for each $i, p\ (0 \le i, p \le r-1)$, the folloiwng one variable function $C_i^p : \{0,1,\ldots,r-1\} \to \{0,1,\ldots,r-1\}$ can be represented:

$$C_i^p(x) = \begin{cases} p & (x = i) \\ 0 & (x \ne i) \end{cases}$$

Let the corresponding combinator with type $\mathtt{T_r}$->$\mathtt{T_r}$ be $\mathtt{C}_i^p$.

Next we consider representation of two variable functions $M : \{0,1,\ldots,r-1\}^2 \to \{0,1,\ldots,r-1\}$, which can be considered as a square matrix with order r.

For the preparation, we define the following auxiliary term, which represents the higher-order function that receives a function $F : \{0,\ldots,r-1\} \to \{0,\ldots,r-1\}$ and a value $j \in \{0,\ldots,r-1\}$ and returns the value $M(i,j) \in \{0,\ldots,r-1\}$:

$$\texttt{fun row\_i F h} = \texttt{h const\_f\_M}_{i,k-1}\texttt{const\_f\_M}_{i,k-2}\cdots\texttt{const\_f\_M}_{i,0}\ \texttt{id (F v\_0)}$$

Then we can realize M as follows:

$$\texttt{fun M h} = \texttt{h row\_k}-1\ \texttt{row\_k}-2\ \cdots\ \texttt{row\_0 id}$$

The combinator M can be typed with $\mathtt{B_r}$->$\mathtt{B_r}$->$\mathtt{B_r}$.

Thus we can represent the following r-valued conjunction $x\&y$ and disjunction $x \sqcup y$ as combinators:

$$\begin{aligned} x\&y &= \min\{x,y\} \\ x \sqcup y &= \max\{x,y\} \end{aligned}$$

We can easily see that n variable versions of conjunction and disjunction can be realized as combinators. Let the generalized conjunction and disjunction with type $\overbrace{\mathtt{T_r}\text{->}\cdots\text{->}\mathtt{T_r}}^{n}\text{->}\mathtt{T_r}$ be $\&_n$ and $\sqcup_n$ respectively.

In our type system, a duplication function like $x \mapsto (x,x)$ can not be represented without any constraint. Nevertheless, such a duplication function can be represented in a restricted manner.

For this purpose, we need a term for function application of tensor products:

```
fun tp_app h z = let val (f,g) = h in let val (x,y) = z in (f x,g y) end end
```

Then we can represent the function returning (v_i, v_i) when receiving v_i for each $i\,(0 \le i \le r-1)$:

```
fun copy v =
    let val (x, y) =
      v (tp_app (const_0, const_0)) ··· (tp_app (const_k − 1, const_k − 1))
                                                              (v_0, v_0)
     in (x, y) end
```

The combinator copy can be typed with $\mathtt{T_r}$->($\mathtt{T_r}$, $\mathtt{T_r}$). Moreover we can easily define the generaized version of copy with type $\mathtt{T_r}$->$(\overbrace{\mathtt{T_r}, \cdots, \mathtt{T_r}}^{n})$, which we call $\mathtt{copy}_n$.

Based on the preparations, using the method described in[Eps93], which is a generalization of disjunctive normal form of boolean functions, we can represent any n-variable r-valued function as a combinator with type $\overbrace{\mathtt{T_r}\text{->} \cdots \text{->}\mathtt{T_r}}^{n}\text{->}\mathtt{T_r}$.

Concretely we suppose we are given a n variable r-valued function $f : \{0, 1, \ldots, r-1\}^n \to \{0, 1, \ldots, r-1\}$. For each $(u_1, \ldots, u_n) \in \{0, 1, \ldots, r-1\}^n$, when we have $v = f(u_1, \ldots, u_n)$, we construct the following monomial term:

$$\&_n (\mathtt{C}^{v}_{u_1} \mathtt{x}_1) \cdots (\mathtt{C}^{v}_{u_n} \mathtt{x}_n)$$

Next we combine these r^n monomials with $\sqcup_{r^n}$. Then using $\mathtt{copy}_{r^n}$, we can unify the same variable occurrences. We can easily see that the resulting combinator has type $\overbrace{\mathtt{T_r}\text{->} \cdots \text{->}\mathtt{T_r}}^{n}\text{->}\mathtt{T_r}$.

References

[Eps93] George Epstein. *Multiple-valued logic design: an introduction*. IOP Publishing Ltd., GBR, 1993.

[Mat16] Satoshi Matsuoka. Strong typed Böhm theorem and functional completeness on the linear lambda calculus. In *Proceedings of 6th Workshop on Mathematically Structured Functional Programming, MSFP 2016*, pages 1–22, 2016.

Current address: Research Institute for Engineering Measurement,, National Metrology Institute of Japan

Email address: s-matsuoka@aist.go.jp

ペトリネットにおける
到達可能性問題の変種間の
還元可能性の形式化

手塚 凜 （千葉大学大学院融合理工学府 M2）
山本 光晴 （千葉大学大学院理学研究院）

1．ペトリネットとは

2．4種の到達可能性問題と
その間の還元可能性

3．証明の形式化手順

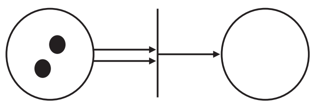
4．まとめと今後の課題

ペトリネットとは

・**プレース**と**トランジション**をノードに持つ、有限有向2部グラフ。

・プレースには有限個の**トークン**が置かれ、
　この配置のことを**マーキング**といい、ペトリネットの状態を表す。

（ペトリネットの例）



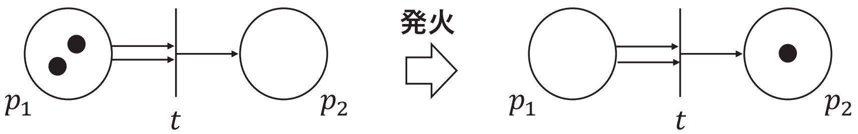
このときのマーキングを $(2, 0)$ と書く

図記号	名称
○	プレース
｜	トランジション
●	トークン

発火

トランジション上の操作「**発火**」は、マーキングの**遷移**を表す。

(ex) トランジション t で発火



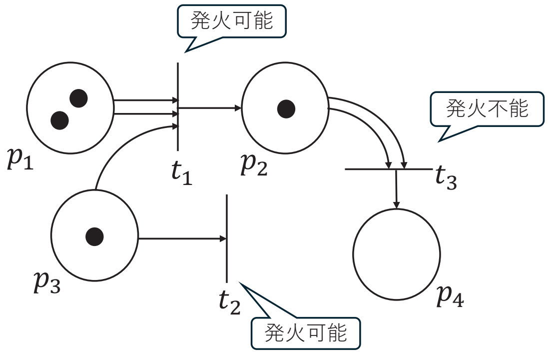
このとき、マーキングの遷移を次のように表す。

$$(2, 0) \xrightarrow{t} (0, 1)$$

発火可能条件

あるトランジションに伸びる矢印の本数が、その矢印で繋がっている
プレースのトークン数以下の時にのみ、そのトランジションは発火可能。

(ex)



1. ペトリネットとは

2. 4種の到達可能性問題と
その間の還元可能性

3. 証明の形式化手順

4. まとめと今後の課題

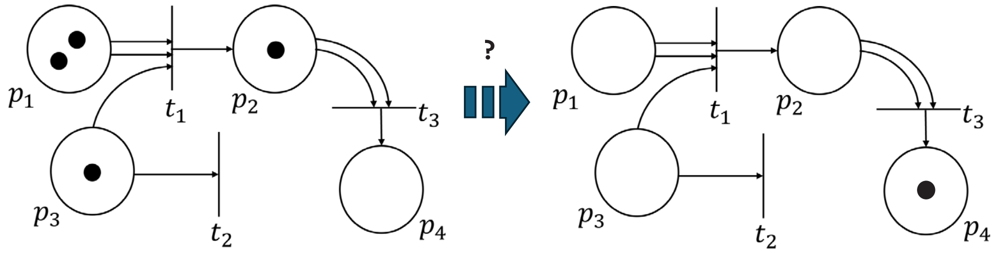
4種類の到達可能性問題

RP

1．**到達可能性問題** (The Reachability Problem)

… 初期マーキングm_0からマーキングmに到達可能なトランジションの有限列が存在するか？

(ex) $m_0 = (2, 1, 1, 0), m = (0, 0, 0, 1)$のとき



4種類の到達可能性問題

SRP

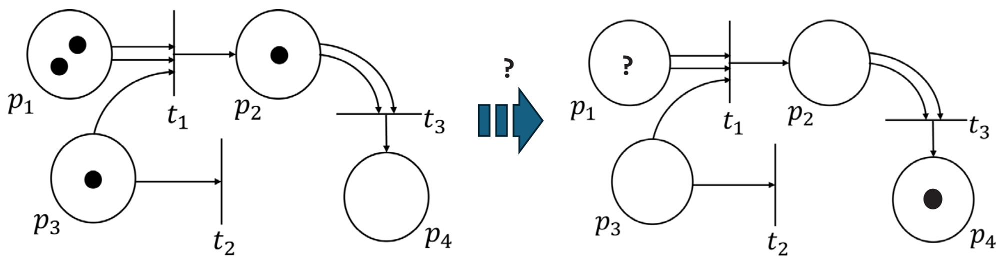
2．**サブマーキング到達可能性問題** (The Submarking Reachability Problem)

… 初期マーキングm_0から到達可能で、かつサブマーキングmcと合致するマーキングmが存在するか？

【サブマーキングとは？**】**

ペトリネット中の一部のプレースのみトークン数が指定されているマーキング。

(ex) $m_0 = (2, 1, 1, 0), mc = (-, 0, 0, 1)$のとき



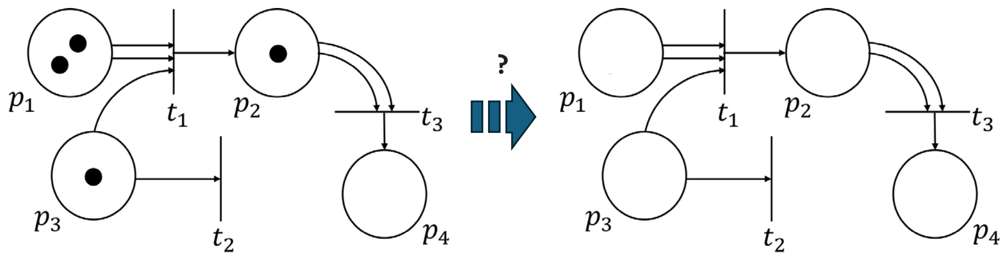
4種類の到達可能性問題

ZRP

3．ゼロ到達可能性問題 (The Zero Reachability Problem)

… 初期マーキングm_0から、全てのプレースのトークン数が0であるゼロマーキングに到達可能か？

(ex) $m_0 = (2, 1, 1, 0)$のとき



4種類の到達可能性問題

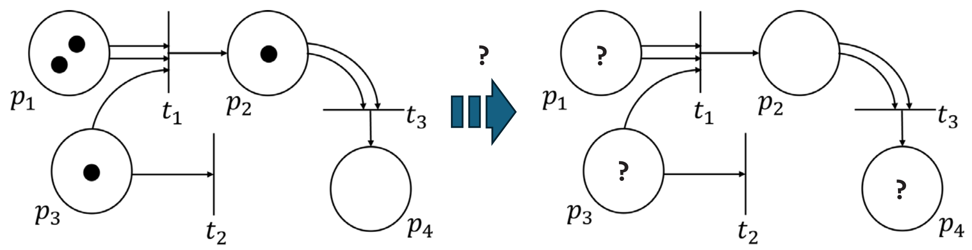
SPZRP

4．Single-Placeゼロ到達可能性問題

(The Single-Place Zero Reachability Problem)

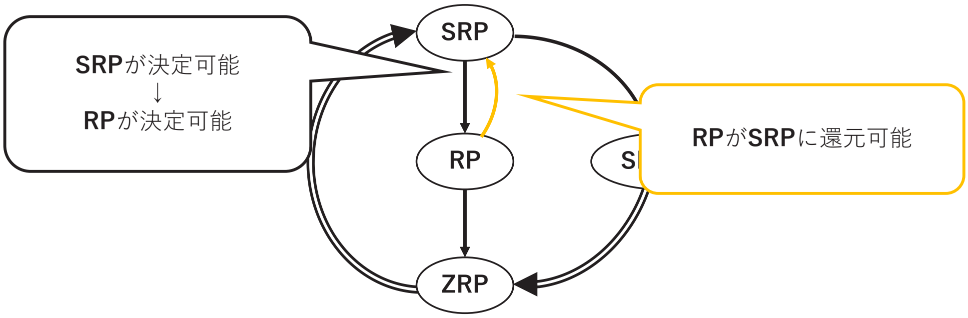
… 与えられたプレースpのトークン数が0であるようなマーキングmに、初期マーキングm_0から到達可能か？

(ex) $m_0 = (2, 1, 1, 0)$, $p = p_2$のとき



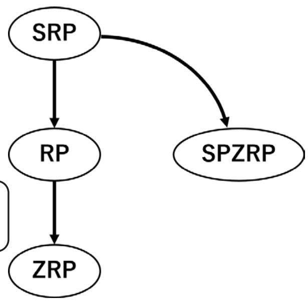
4 種類の到達可能性問題の間の関係

Theorem 4.1 [Hack 1976]
RP, SRP, ZRP, SPZRPは互いに還元可能(reducible)。



4 種類の到達可能性問題の間の関係

Theorem 4.1　RP, SRP, ZRP, SPZRPは互いに還元可能(reducible)。



RPとSPZRPはSRPに、ZRPはRPに還元可能。

還元される問題は還元先の問題の
インスタンスになっている。

(ex) ZRPはRPのインスタンス

mがゼロマーキングなら
ZRPの定義そのもの

RP： m_0からmに到達可能か
ZRP： m_0からゼロマーキングに到達可能か

4種類の到達可能性を形式化

2017年 山本らにより既に形式化済

1．到達可能性

```
Definition reachable (m0 m : marking) : Prop :=  ∃ s, m0 ={ s }> m.
```

2．サブマーキング到達可能性

```
Definition submreachable (m0 : marking) (mc : markingc) : Prop :=
  ∃ m, reachable m0 m ∧ ∀ p, if mc p is Some n then n = m p else True.
```

3．ゼロ到達可能性

```
Definition zeroreachable (m0 : marking) : Prop := reachable m0 [ffun=> 0].
```

4．Single-Placeゼロ到達可能性

```
Definition spzreachable (m0 : marking) (p : place) : Prop :=
  ∃ m, reachable m0 m ∧ m p = 0
```

証明に多対一還元を用いる

Theorem 4.1 RP, SRP, ZRP, SPZRPは互いに還元可能(reducible)。

集合$A, B \subseteq \mathbb{N}$に対して、 AがBに**多対一還元(many-one reduction)**されるとは、ある計算可能関数 $f : \mathbb{N} \to \mathbb{N}$が存在して、

$$n \in A \iff f(n) \in B$$

を満たすことである。

[? $f(n) \in B$]が決定可能
→ [? $n \in A$]が決定可能

⇩ **形式化**

```
Definition mreducible {T1 T2 : Type}
                      (P : T1 → Prop) (Q : T2 → Prop) : Type :=
  {f | ∀ x : T1, P x ↔ Q (f x)}.
```

補題を作る

Theorem 4.1 RP, SRP, ZRP, SPZRPは互いに還元可能(reducible)。

```
Lemma SPZRP_red_SRP :
 mreducible (fun '(m0, p) =:
             (fun '(m0, mc) :
```

4種類の到達可能性の定義は
「reachable m0 m ∧ P x m」の形で表せるので
その差分だけで議論ができるようになっている。

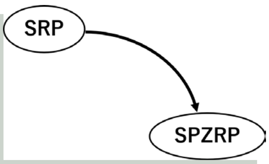
x ： 問題固有のパラメタ

証明には次の補題を用いる

```
Lemma mreducible_Preachable (T1 T2 : Type) (f : T1 → T2)
                   (P : T1 → marking → Prop) (Q : T2 → marking     op) :
 (∀ (x : T1) (m : marking), P x m ↔ Q (f x) m)
 → mreducible (fun '(m0, x) => ∃ m : marking, reachable m0 m ∧ P x m)
              (fun '(m0, y) => ∃ m : marking, reachable m0 m ∧ Q y m).
```

SPZRPはSRPに還元可能である。

Theorem 4.1 RP, SRP, ZRP, SPZRPは互いに還元可能(reducible)。




```
Lemma SPZRP_red_SRP :
 mreducible (fun '(m0, p) => spzreachable m0 p)
            (fun '(m0, mc) => submreachable m0 mc).
Proof.
apply: (@mreducible_Preachable _ _
        (fun p : place => [ffun p' => if p == p' then Some 0 else None])).
move=> p m; split; last by move/(_ p); rewrite ffunE eq_refl.
by move=> h p'; rewrite ffunE; case: ifP => //; move/eqP=> <-.
Qed.
```

1． ペトリネットとは

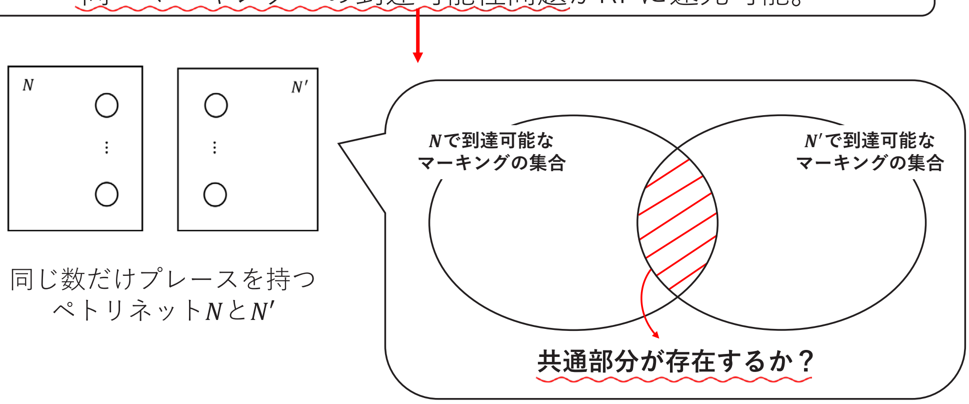
2． 4種の到達可能性問題と
その間の還元可能性

3． 証明の形式化手順

4． まとめと今後の課題

還元可能性に関する証明形式化手順

プレースの集合が共通している２つのペトリネットの
同一マーキングへの到達可能性問題がRPに還元可能。

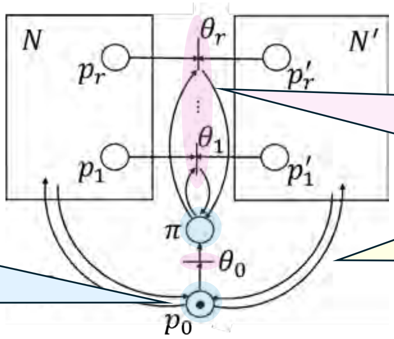


同じ数だけプレースを持つ
ペトリネットNとN'

還元可能性に関する証明形式化手順

プレースの集合が共通している２つのペトリネットの
同一マーキングへの到達可能性問題がRPに還元可能。

[新しくペトリネットN2を構成]



トランジションは
(Nのプレース数)+1個
追加する。

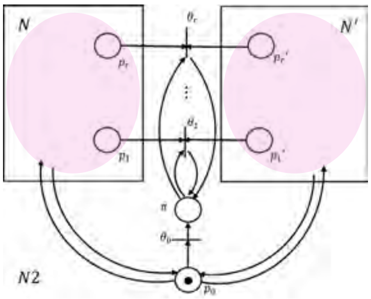
プレースは
2個追加する。

p_0はN ,N'のトランジションと
πはθ_i $(i \neq 0)$とセルフループ

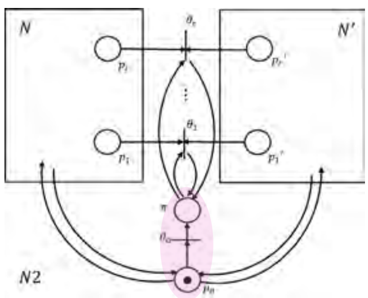
プレースの集合が共通している２つのペトリネットの
同一マーキングへの到達可能性問題がRPに還元可能。

[証明アイデア]

① N 、 N'でそれぞれ
有限回発火する。



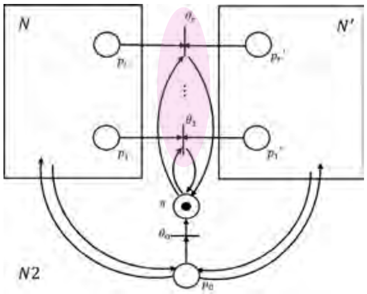
②トランジションθ_0で発火すると
p_0のトークンが πに移動する。



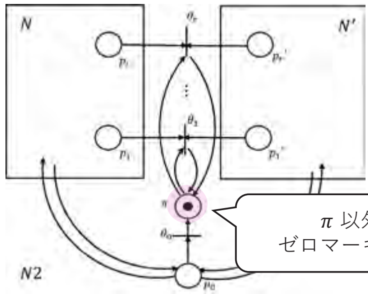
プレースの集合が共通している２つのペトリネットの
同一マーキングへの到達可能性問題がRPに還元可能。

[証明アイデア]

③トランジション$\theta_i (i \neq 0)$で
可能な限り発火する。



④ N, N'が同一マーキング到達可能
⇅
$N2$は $(0, \ldots, 0, 1)$に到達可能



π 以外
ゼロマーキング

プレースの集合が共通している２つのペトリネットの
同一マーキングへの到達可能性問題がRPに還元可能。

[形式化手順]

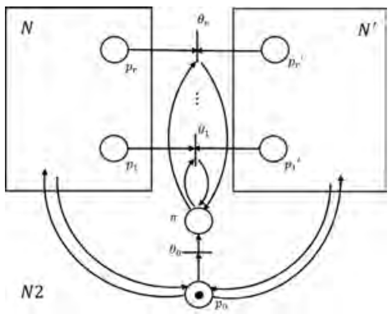
①新しいペトリネットとそのマーキングを定義

②発火可能条件に関する補題を証明

③1ステップ遷移を特徴づける述語を定義

④制御状態ごとの不変条件を考える

⑤定理を証明

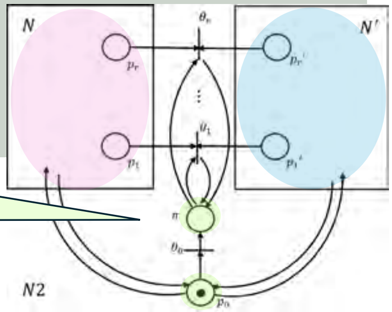


手順①新しいペトリネットのマーキングを定義

先に新しいペトリネットのマーキングを表す関数を定義する。

```
Definition N2_m' (m : marking pn) (m' : marking pn') (b : bool)
                  : {ffun (place pn + place pn' + bool) → nat} :=
  [ffun p => match p with
            | inl (inl p
            | inl (inr p
            | inr true => b
            | inr false => ~~ b
            end].
```

$N2$のプレースの型

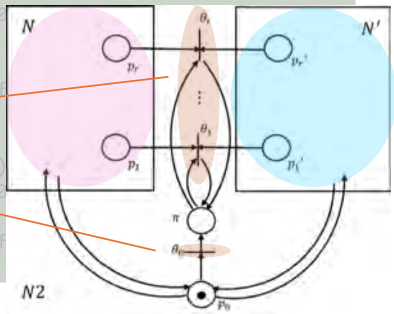


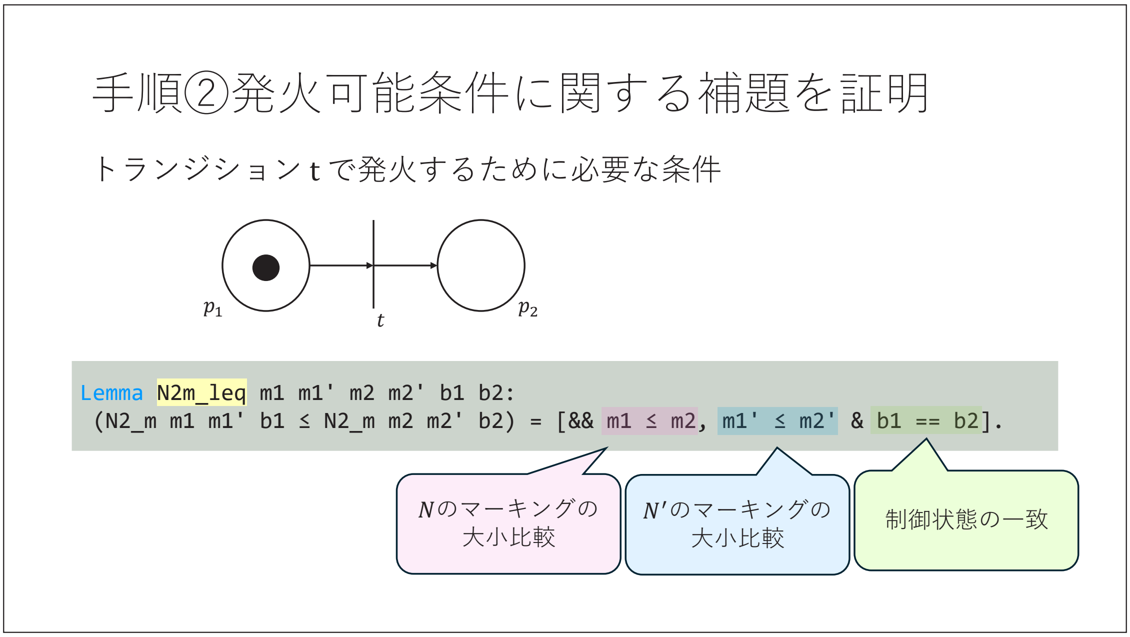
p_0をtrue， πをfalseとする。

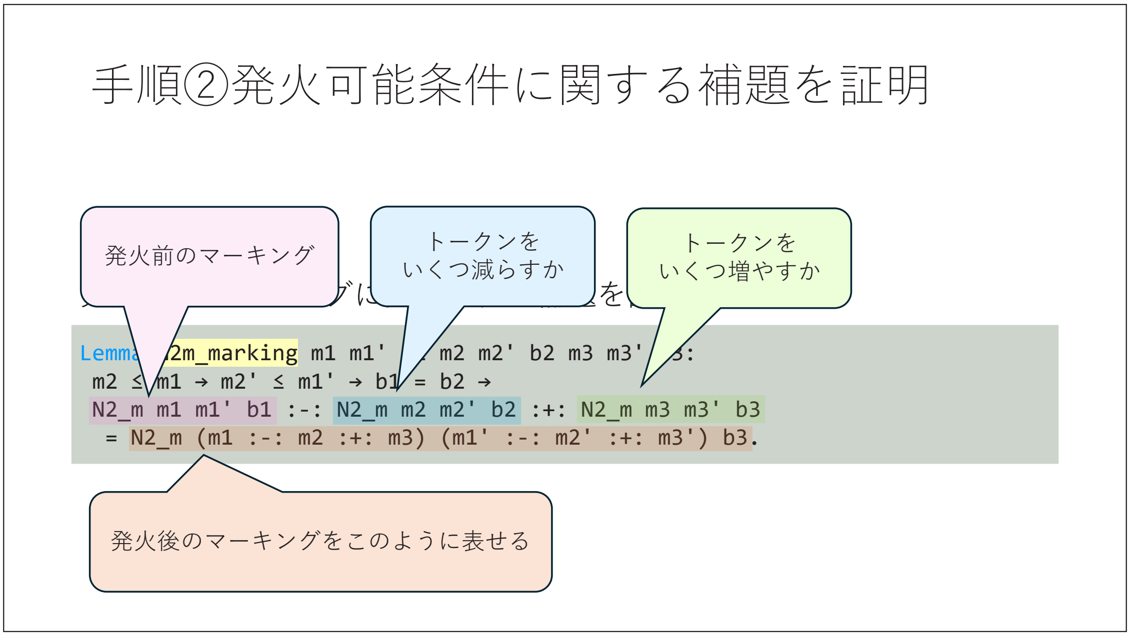
手順①新しいペトリネットを定義

```
Definition N2 : petri_net :=
 @PetriNet (place pn + place pn' + bool)%type
           (transition pn + transition pn' + option (place pn))%type
           (fun t => match t with
                     | inl (inl t') => N2_m' (  e t') [ffun=> 0] true
                     | inl (inr t') => N  [ffun=> 0
                                      m' [ffun p
                                         [ffun p
                                      ffun=> 0] [f
                                      m' (post t')
                                      2_m' [ffun=> 0
                                 ) => N2_m' [ffun=>
                     | inr None => N2_m' [ffun=> 0] [f
                     end).
```

新しく追加された
トランジションθ_iを
$\begin{cases} \text{Some p} & (i \neq 0) \\ \text{None} & (i = 0) \end{cases}$
で表す。




手順②発火可能条件に関する補題を証明
トランジション t で発火するために必要な条件
p1
t
p2
Lemma N2m_leq m1 m1' m2 m2' b1 b2:
(N2_m m1 m1' b1 ≤ N2_m m2 m2' b2) = [&& m1 ≤ m2, m1' ≤ m2' & b1 == b2].
Nのマーキングの大小比較
N′のマーキングの大小比較
制御状態の一致


手順②発火可能条件に関する補題を証明
発火前のマーキング
トークンをいくつ減らすか
トークンをいくつ増やすか
m2 ≤ m1 → m2' ≤ m1' → b1 = b2 →
N2_m m1 m1' b1 :-: N2_m m2 m2' b2 :+: N2_m m3 m3' b3
= N2_m (m1 :-: m2 :+: m3) (m1' :-: m2' :+: m3') b3.
発火後のマーキングをこのように表せる

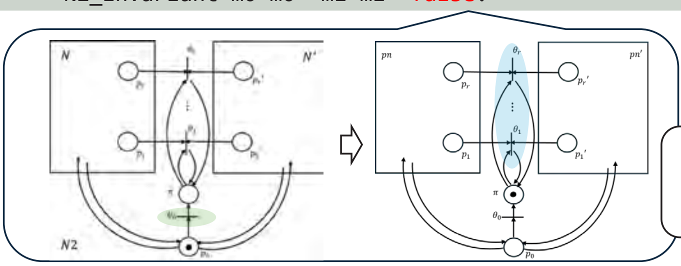
手順③1ステップ遷移を特徴づける述語

```
Variant N2_step : marking pn → marking pn' → bool → marking pn
                  → marking pn' → bool → transition N2 → Prop :=
| N2_pn m0 m0' t m1 :
    m0 =1{ t }> m1 →
    N2_step m0 m0' true m1 m0' true (inl (inl t))
| N2_pn' m0 m0' t m1' :
    m0' =1{ t }> m1' →
    N2_step m0 m0' true m0 m1' true (inl (inr t))
| N2_b1 m0 m0' :
    N2_step m0 m0' true m0 m0' false (inr None)
| N2_b2 t (m1 : marking pn) (m1' : marking pn') :
    N2_step (m1 :+: [ffun p => t == p])
              (m1' :+: [ffun p => f t == p]) false
     m1 m1' false (inr (Some t)).
```

手順④制御状態ごとの不変条件を定義

```
Inductive N2_invariant (m0 : marking pn) (m0' : marking pn') :
                            marking pn → marking pn' → bool → Prop :=
  N2_inv_T m m' :
     reachable m0 m → reachable m0' m'
      → N2_invariant m0 m0' m m' true
| N2_inv_F m1 m1' (md : marking pn') :
     N2_invariant m0 m0'
     (m1 :+: [ffun p => md (f p)]) (m1' :+: md) true
     → N2_invariant m0 m0' m1 m1' false.
```

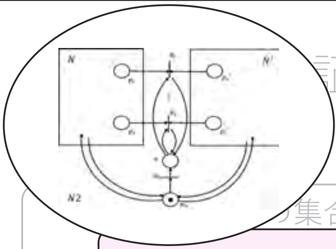
トランジションθ_iの多重集合

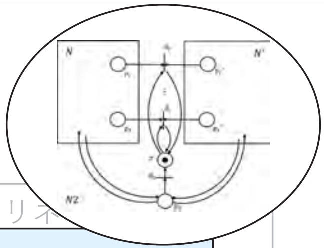


制御状態 p_0, πの
マーキングが
falseのとき

証明

…集合が共通している２つのペトリネ…
…到達可能性問…





初期マーキング

マーキング$(0, \ldots, 0, 1)$

形式化

```
Lemma CMP_iff_RP (m0 : marking pn) (m0' : marking pn') :
 reachable (N2_m0 m0 m0') (N2_m [ffun=> 0] [ffun=> 0] false)
  ↔ ∃ m : marking pn',
       reachable m0' m ∧ reachable m0 [ffun p => m (f p)].
```

N, N'が同一マーキングに到達可能

1. ペトリネットとは
2. 4種の到達可能性問題とその間の還元可能性
3. 証明の形式化手順
4. まとめと今後の課題

まとめ

1976年のHackの論文の4章で証明されている、
いくつかの到達可能性問題の変種間の還元可能性を形式化した。

・多対一還元の考え方を用いて証明した。
・自明な証明では、それぞれの定義の差分を使った。
・非自明な証明では、アドホックに証明するのではなく、
　系統的に証明するための共通手順を作成し、それに従って証明した。

同じ4章で証明されている、「半線形集合が与えられたとき、
その中の要素への到達可能性がRPにチューリング還元可能」という
定理も形式化。

今後の課題

・Hackの論文の5章以降の証明
・5章：Liveness (任意のトランジションtに対して、初期マーキングから到達可能な
　すべてのマーキングがtで潜在的に発火可能)
　Persistence(任意のトランジションtに対して、初期マーキングから到達可能な
　すべてのマーキングで他のトランジションの発火により
　tが発火できなくなることがない)
　これらの問題がRPに還元可能

・去年のTPPで脇坂らが示したVASSからVASへの変換

⇒これらも同じ共通手順で形式化できるか

Axiomatic real numbers for verified exact real-number computation

Sewon Park (Kyoto University)
j.w.w. Holger Thies (Kyoto U.) and Michal Konečný (Aston U.)

TPP 2024: 20th Theorem Proving and Provers Meeting
November 25-26, 2024
Kyushu University, IMI, Kyushu, Japan

Overview

Axiomatic real numbers for verified exact real-number computation

- **Part I:** (1) Introduce exact real-number computation (2) and axiomatic reals in constructive type theory whose interpretation corresponds to *exact real-number computation* (cAERN in Coq: `https://github.com/holgerthies/coq-aern`)

  Michal Konečný, SP, Holger Thies : **Extracting efficient exact real number computation from proofs in constructive type theory (2024)**, *JLC*

- **Part II:** Recent progress in formalization of (1) continuity principle, (2) various subsets of general spaces and *polish* spaces, and (3) hyperspace computations over those

  ·: **Formalizing Hyperspaces for Extracting Efficient Exact Real Computation**. Mathematical Foundations of Computer Science (**MFCS 2023**)

Part I

Computable Analysis - Exact Real Computation

- Infinite representations for real numbers
  E.g., rationals $q_1, q_2, \cdots$ expresses $x \Leftrightarrow \forall i.\ |x - q_i| \leq 2^{-i}$
  exact computations by type-2 machines
  E.g., $x + y$ is realized by $(p_i)_i, (q_i)_i \mapsto (p_{i+1} + q_{i+1})_i$
- Hide representation-specific details
  $\rightsquigarrow$ ***Abstract data type*** *for exact real numbers*:

```
>>> print(pi, 10) # print 2^{-10} approximation of π
3.14159 ± 2^{-10}
>>> print(pi, 100)
3.14159265358979323846··· ± 2^{-100} # for high-precision result
>>> pi + sqrt(2) # evaluates exactly to π + √2
>>> print(pi + sqrt(2), p) # prints 2^{-p} approx. to π + √2
```

- **Expectation**: intuitive reasoning with reals as in textbooks
- Semantics of an arithmetical expression e is simply e, without a rounding error

Naïve Reals in Constructive Type Theory

- Constructive dependent type theory:

$$A + B \text{ is valid } \cong \text{deciding } A \text{ or } B \text{ is computable}$$

$$\Sigma(x : A).\ B(x) \text{ is valid } \cong \text{finding } x : A \text{ s.t. } B(x) \text{ is computable}$$

- Certified program extraction:

$$\Pi(x : A).\ \Sigma(y : B).\ R(x, y)$$

  yields a program $\mathcal{P} : A \to B$ s.t. $\forall(x : A).\ R(x, \mathcal{P}(x))$
- Classical axiomatization of reals is invalid:

$$\text{Trichotomy} : \Pi(x : \mathsf{R}).\ (x < 0) + (x = 0) + (x > 0)$$

  The sign test of reals is not computable
- Axiomatization of exact reals s.t.
  proofs $\cong$ programs in ERC framework (viz. AERN in Haskell)
- **cAERN project** in Coq, certified AERN program extraction from analysis proofs

3/13

Constructive Axiomatic Reals

- Real numbers R : Type
  Non-determinism $\mathbf{M}$: Type $\to$ Type monad
- Comparison is a classical proposition $x < y$: Prop
  where we make Prop classical by adding LEM in Prop
- Field arithmetic and proper completeness
- Order theory by
  $\Pi(x, y, \epsilon : \mathsf{R}).\ \epsilon > 0 \to \mathbf{M}((x < y + \epsilon) + (y < x + \epsilon))$
  realized by testing $x < y + \epsilon$ *and* $y < x + \epsilon$ *in parallel*

  $\Pi(x, y : \mathsf{R}).\ ((x < y) + (x = y) + (y < x))$
  $\Pi(x, y, \epsilon : \mathsf{R}).\ \epsilon > 0 \to ((x < y + \epsilon) + (y < x + \epsilon))$
  are invalid!!

- Extraction mechanism to AERN (Haskell)

$$f : \Pi(x : \mathsf{R}).\ \mathbf{M}\Sigma(y : \mathsf{R}).\ P(x, y) \text{ extracts to}$$

  a non-deterministic AERN function `CReal` $\to$ `CReal`

4/13

Nice Example

Define $\mathtt{isMax}(m, x, y) :\equiv (x \geq y \to m = x) \land (y \geq x \to m = y)$ as a classical predicate and prove:

$$\Pi(x, y : \mathsf{R}).\ \Sigma(m : \mathsf{R}).\ \mathtt{isMax}(m, x, y)$$

Proof.

- limit as $n \to \infty$:
- *assume* $\mathbf{M}((x < y + 2^{-n}) + (y < x + 2^{-n}))$ ← axiom
- *assume* $((x < y + 2^{-n}) + (y < x + 2^{-n}))$
- *case 1:* $x < y + 2^{-n}$, y approximates the max by 2^{-n}
- *case 2:* $y < x + 2^{-n}$, x approximates the max by by 2^{-n}
- $\Sigma(m : \mathsf{R})$. m approximates the max by 2^{-n}
- $\mathbf{M}\Sigma(m : \mathsf{R})$. m approximates the max by 2^{-n} ← $\mathbf{M}$-lift
- $\Sigma(m : \mathbb{R}).\ \mathtt{isMax}(m, x, y)$ ← completeness □

extracts to the maximum function in AERN

5/13

Code Extraction Example

```
Lemma real_max_prop :
  forall x y, {z | (x >= y -> z = x)
                 /\ (x < y -> z = y)}.
 Proof.
  intros.
  apply real_mslimit_P_lt.
  + (* max is single-valued *)
   ...
  + (* construct limit *)
   intros.
   apply (mjoin (x>y - prec n)
                (y>x - prec n)).
   ++ intros [c1|c2].
      +++ (* when x > y - 2^-n *)
      exists x.
      ...
      +++ (* when x < y - 2^-n *)
      exists y.
      ...
   ++ apply M_split.
      apply prec_pos.
 Defined.
```

```
real_max_prop ::
    AERN2.CReal ->
    AERN2.CReal ->
    AERN2.CReal
real_max_prop x y =
  AERN2.limit (\n ->
    Prelude.id (\h -> case h of {
                  P.True  -> x;
                  P.False -> y})
   (m_split x y (prec n)))
```

```
Lemma real_max_prop :
  forall x y, {z | (x >= y -> z = x)
                 /\ (x < y -> z = y)}.
```

```
real_max_prop ::
    AERN2.CReal ->
    AERN2.CReal ->
```

6/13

Assemblies

- Assembly $X = (|X|, \Vdash_X)$ is a pair of a set $|X|$ and a binary relation $\Vdash_X \subseteq \mathbb{N}^\omega \times |X|$ that is surjective:

$$\forall (x \in |X|).\ \exists (\varphi \in \mathbb{N}^\omega).\ \varphi \Vdash_X x$$

- $f : |X| \to |Y|$ is computable if there is a *type-2 machine* $\tau :\subseteq \mathbb{N}^\omega \to \mathbb{N}^\omega$ that tracks f:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ \Vdash_X \vdots & & \vdots \Vdash_Y \\ \mathbb{N}^\omega & \xrightarrow{\tau} & \mathbb{N}^\omega \end{array}$$

$$\forall (x \in |X|).\ \forall (\varphi \in \mathbb{N}^\omega).\ \varphi \Vdash_X x \Rightarrow \tau(\varphi) \Vdash_Y f(x)$$

- Category of assemblies & computable functions $\mathsf{Asm}(\mathbb{N}^\omega)$ forms LCCC modeling Dependent Type Theory

What holds in $\mathsf{Asm}(\mathbb{N}^\omega)$, *let it be in cAERN!*

Validity of the Axiomatization

- Standard Cauchy assembly $|\mathsf{R}| = \mathbb{R}$:

$$\varphi \Vdash_{\mathsf{R}} x \quad :\iff \quad \varphi(n) \textit{ encodes } q_n \in \mathbb{Q}.\ |q_n - x| < 2^{-n} \text{ for all } n$$

- Nondeterminism monad $\mathbf{M} : \mathsf{Asm}(\mathbb{N}^\omega) \to \mathsf{Asm}(\mathbb{N}^\omega)$

$$|\mathbf{M}\, X| :\equiv \{A \subseteq |X| \mid A \neq \emptyset\} \quad \varphi \Vdash_{\mathbf{M}\, X} A \quad :\iff \quad \exists (x \in A).\ \varphi \Vdash_X x$$

- Sierpiński assembly $|\mathsf{S}| = \{\bot, \top\}$:

$$\varphi \Vdash_{\mathsf{S}} \bot :\iff \forall (i \in \mathbb{N}).\ \varphi(i) = 0$$
$$\varphi \Vdash_{\mathsf{S}} \top :\iff \exists (i \in \mathbb{N}).\ \varphi(i) \neq 0$$

- classifies opens/semi-decidable subsets:

$$f : \mathsf{R} \to \mathsf{S} \iff f \text{ characterize a semi-decidable } S \subseteq \mathbb{R}$$

- The axiom saying $x < y$ is semi-decidable is valid:

$$\Pi(x, y : \mathsf{R}).\ \Sigma(s : \mathsf{S}).\ s = \top \leftrightarrow x < y$$

- The set of axioms are valid and universal [Hertling99]

Part II

Choice and Continuity

- In $\mathrm{Asm}(\mathbb{N}^\omega)$, when N is nno, N → N admits choice:

$$\Pi(F : (\mathsf{N} \to \mathsf{N}) \to \mathbf{M}\ X).$$
$$\mathbf{M}\ \Sigma(f : (\mathsf{N} \to \mathsf{N}) \to X).\ \Pi(x : \mathsf{N} \to \mathsf{N}).\ f\ x \in F\ x$$

- *Continuity Principle*: $(\mathsf{N} \to X) \to \mathsf{S}$ for any X is open

$$\Pi(f : (\mathbf{N} \to X) \to \mathsf{S}).\ \Pi(x : (\mathbf{N} \to X)).$$
$$f\ x = \top \to \mathbf{M}\Sigma(n : \mathsf{N}).\ \Pi(y : (\mathbf{N} \to X)).\ \bar{x}_n = \bar{y}_n \to f\ y = \top$$

- The axioms are enough to prove that all S-valued functions from reals are *continuously continuous*:

For all $f : \mathsf{R} \to \mathsf{S}$, there nondeterministically is $\mu : \mathsf{R} \to \mathbf{M}\ \mathsf{N}$

- that is a modulus of continuity of f

$$\Pi(x : \mathsf{R}).\ \Pi(n : \mathsf{N}).\ |x - y| < 2^{-(\mu\ x\ n)} \to |f\ x - f\ y| < 2^{-n}$$

- and is again continuous.

9/13

Subsets

- Classical subsets of X : Type is $\mathcal{P}(X) :\equiv X \to \text{Prop}$
- Open subsets

$$\mathcal{O}(X) :\equiv \Sigma(U : \mathcal{P}(X)).\ \Sigma(\chi : X \to \mathrm{S}).\ \Pi(x : X).\ x \in U \leftrightarrow \chi\ x = \top$$

- Compact subsets

$$\begin{aligned}\mathcal{K}(X) &:\equiv \Sigma(K : \mathcal{P}(X)).\ \Sigma(f : \mathcal{O}(X) \to \mathrm{S}).\\ &\qquad \Pi(U : \mathcal{O}(X)).\ (f\ \ U) = \top \ \leftrightarrow\ K \subseteq U.\end{aligned}$$

- I.e., a subset $K :\subseteq X$ is compact *iff* $P : X \to \text{Prop}$ being semi-decidable implies $\forall(x \in K).\ P\ x$ is also semi-decidable.
- Overt subsets

$$\begin{aligned}\mathcal{V}(X) &:\equiv \Sigma(V : \mathcal{P}(X)).\ \Sigma(f : \mathcal{O}(X) \to \mathrm{S}).\\ &\qquad \Pi(U : \mathcal{O}(X)).\ (f\ \ U) = \top \ \leftrightarrow\ V \cap U \neq \emptyset\end{aligned}$$

- I.e., a subset $V :\subseteq X$ is overt *iff* $P : X \to \text{Prop}$ being semi-decidable implies $\exists(x \in V).\ P\ x$ is also semi-decidable.

10/13

Polish Spaces

- Polish space is complete, separable metric space
  including any discrete spaces, R, *Euclidean spaces,* ...
- A subset B is *Bishop-compact* if it is totally-bounded and complete;
  i.e., if B can be drawn exactly: for all n, there are finite boxes B_i such that
  - each box has size less than 2^{-n}
  - each box intersects S
  - S is covered by the boxes
- We can prove, in a polish space, a subset is Bishop-compact iff it is compact and overt; cf.
  `[Coquand, T., Palmgren, E., Spitters, B. (2009)]`
- Formalizing the various subsets and results so far, given a compact and overt subset, we obtain a program that correctly draws the set up to arbitrary resolution.

11/13

Completeness and Fractals

- Bishop-compact subsets is complete for the Hausdorff distance
- Define the sequence of Bishop-compact subsets in R^2:
  - $S_0 = T$ where T is an arbitrary triangle in the unit cube
  - $S_{i+1} = (0.5 \cdot S_i) \cup (0.5 \cdot S_i + (0.5, 0)) \cup (0.5 \cdot S_i + (0, 0.5))$
- $\lim_{i \to \infty} S_i$ is the Sierpinski Triangle and is constructible via the completeness theorem in cAERN

12/13

Completeness and Fractals

- Bishop-compact subsets is complete for.the Hausdorff distance
- Define the sequence of Bishop-compact subsets in R^2:
  - $S_0 = T$ where T is an arbitrary triangle in the unit cube
  - $S_{i+1} = (0.5 * S_i) \cup (0.5 * S_i + (0.5, 0)) \cup (0.5 * S_i + (0, 0.5))$
- $\lim S_i$ is the Sierpinski Triangle and is constructible via the completeness theorem in cAERN

12/13

Conclusion

In this talk, the followings were presented:

- Axiomatization of computable analysis in constructive dependent type theory and the cAERN project in Coq
- Recent formalization of various classes of subsets and those in Polish space and its application in exact drawing of fractals

Future work includes:

- Other applications e.g., extending the ODE solving:
  SP and Holger Thies: **A Coq Formalization of Taylor Models and Power Series for Solving Ordinary Differential Equations**. *ITP 2024*
- Formalizing and verifying program extraction using meta-level programming and reasoning using e.g. `MetaCoq`
- Relating ours to other famous classical formalization e.g. `mathcomp-analysis` (transfer principle, type-theoretic generalization of the double-negation translation)

⇒ Classical reasoning (of computational content) in cAERN can be done relying on those rich libraries

% ご清聴くれはって、ありがとうございます！

Integrating cost and behavior in type theory

Yue Niu

National Institute of Informatics

November 26, TPP 2024

About me

Postdoc at National Institute of Informatics; previously a PhD student at Carnegie Mellon University.

Collaborators: Harrison Grodin (CMU), Robert Harper (CMU), Jon Sterling (University of Cambridge).

Introduction

Define and prove cost bounds of programs in type theory — execution time, memory usage, *etc*. Discussed in the abstract for this talk.

1. Type theory is a theory of *functions*, *e.g*. $isort = msort$. **Nonstarter** to model algorithms as (plain) functions.
2. Nonetheless purely behavioral/mathematical reasoning of algorithms *qua* functions is indispensable for not only correctness but also *cost*. Example: worst case cost bound of *isort*.

Must integrate cost structure in a way that preserves the view/aspect of programs as functions.

A cost-aware logical framework

This talk outlines a cost-sensitive type theory dubbed **calf** (a **c**ost-**a**ware **l**ogical **f**ramework) [Niu+22].

Implemented in Agda by means of computational postulates [CTW21].

https://github.com/jonsterling/agda-calf

Outline

Three key ingredients of **calf**.

1. Cost structure as an **abstract effect** (call-by-push-value [Lev03]).
2. Functional/behavior aspect of programs as a **phase distinction**.
3. Cost analysis in terms of program **inequalities** (**decalf** — *directed*, *effectful* **calf**).

Cost as an abstract effect

Cost structure as an abstract effect

Only expose the *interface* of the writer monad.

postulate

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

$\mathsf{ret} : A \to \mathsf{F}A$

$\mathsf{bind} : \mathsf{F}A \to (A \to \mathsf{F}B) \to \mathsf{F}B$

$\mathsf{charge} : \mathbb{C} \times \mathsf{F}A \to \mathsf{F}A$

Implement $\mathsf{F}A = \mathbb{C} \times A$ under the hood, but not revealed to the user.

Example: *isort*

$$
\begin{aligned}
&\mathit{insert} : A \to \mathsf{list}(\mathsf{nat}) \to \mathsf{F}(\mathsf{list}(\mathsf{nat})) \\
&\mathit{insert}(x, \mathsf{nil}) = \mathsf{ret}([\,x\,]) \\
&\mathit{insert}(x, y :: l) = \mathsf{charge}; \mathsf{bind}(\mathit{lt}(x, y), \lambda b. \\
&\qquad \mathsf{case}\ b\ \mathsf{of} \begin{cases} \mathsf{bind}(\mathit{insert}(x, l), \lambda r. \mathsf{ret}(y :: r)) \\ \mathsf{ret}(x :: y :: l) \end{cases}) \\
&\mathit{isort} : \mathsf{list}(\mathsf{nat}) \to \mathsf{F}(\mathsf{list}(\mathsf{nat})) \\
&\mathit{isort}(\mathsf{nil}) = \mathsf{nil} \\
&\mathit{isort}(x :: l) = \mathsf{bind}(\mathit{isort}(l), \lambda l'. \mathit{insert}(x, l'))
\end{aligned}
$$

Example: *isort*

Equational reasoning by means of computational postulates.

postulate

$\text{bind}(\text{ret}(a), f) = f(a)$
$\text{bind}(\text{charge}(e), f) = \text{charge}(\text{bind}(e, f))$

For example we can derive the usual cost bound on insertion sort.

$$isort(l) = \text{charge}^{k \cdot |l|^2}(\text{ret}(sort(l)))$$

Here $sort : \text{list}(\text{nat}) \to \text{list}(\text{nat})$ is *the* sorting function.

The function-cost phase distinction

Algorithms *vs.* functions

We may distinguish algorithms in the presence of the cost effect, *e.g.* *msort* and *isort* may exhibit distinct *cost bounds*.

$$isort(l) = \mathsf{charge}^{k \cdot |l|^2}(\mathsf{ret}(sort(l)))$$
$$msort(l) = \mathsf{charge}^{k \cdot |l| \log |l|}(\mathsf{ret}(sort(l)))$$

Not immediate that purely **functional/mathematical** properties respects the *functional equality* $isort = msort$. How to *redact* cost information?

The phase distinction

Cost-sensitive data is redacted/trivialized in the **functional phase** (*cf*. phase distinctions in program modules [SH21]).

postulate
$\P : \mathsf{Prop}$
$redact : \P \to \mathsf{charge}^{k}(e) = e$

The phase distinction

Cost-sensitive data is redacted/trivialized in the **functional phase** (*cf*. phase distinctions in program modules [SH21]).

postulate
$\P$: Prop
$\mathit{redact} : \P \to \mathsf{charge}^k(e) = e$

Any context containing the assumption $\P$ validates ordinary functional/mathematical reasoning, *e.g*. $\P \vdash \mathit{isort} = \mathit{msort}$.

Introducing other effects

Coping with computational effects

In **calf** cost bounds are defined in terms of *normal forms* — $e : \mathsf{F}A$ bounded by c when $e = \mathsf{charge}^c(\mathsf{ret}(a))$ for some $a : A$.

Breaks down in the presence of nontrivial computational effects (those aside from cost).

Suppose we want to add a probabilistic effect $\mathsf{flip} : X \to X \to X$.

Expect $\mathsf{flip}(e, e)$ is bounded by c if e is bounded by c. But this is false when e is probabilistic — there may not be a single value $a : A$ witnessing the cost bound $\mathsf{flip}(e, e) = \mathsf{charge}^c(\mathsf{ret}(a))$!

Directed, effectful **calf**

decalf: introduce an intrinsic *inequality* relation $\sqsubseteq_X$ at all types [Gro+24].

Instead of equational cost bounds, work in terms of program inequalities. **Moral**: cost bounds are just (simpler) programs.

Example: $\mathsf{flip}(e_1, e_2) \sqsubseteq e$ when $e_1, e_2 \sqsubseteq e$ by $\mathsf{flip}(e, e) = e$ and monotonicity.

Bonus: inequality at all types makes it convenient to state cost bounds for higher-order functions.

Models

calf: internal type theory of the category of presheaves over the interval poset $\{0 \sqsubseteq 1\}$. Phase proposition ¶ is the intermediate proposition given by $¶(1) = \bot$ and $¶(0) = \top$.

decalf: internal type theory of the category of (augmented) simplicial sets, *e.g.* presheaves over the nonempty ordinals Δ. Also provides a *synthetic (partially discrete) poset theory*.

Other projects in **calf**

1. Verifying correctness and cost of data structures: Li, Grodin, and Harper [LGH23].
2. Coinductive algorithms and amortized analysis: Grodin and Harper [GH23].
3. Recursion and adequacy: Niu and Harper [NH23] and Niu, Sterling, and Harper [NSH24].

Call for ideas and collaborations

Computational adequacy and definability.

1. One can justify equational cost bounds on *operational* grounds by means of *internal* versions of computational adequacy in the sense of Plotkin [Plo77]. Refined to track cost in Niu, Sterling, and Harper [NSH24].
2. Good closure properties of operationally *definable* cost-sensitive programs — *cf*. **lambda definability**.
3. *Internal* cost-preserving program extraction.
4. Connection to automated type systems for resource analysis (*e*.*g*. [Hof11])?

20 / 27

Call for ideas and collaborations

Please help us **extend the theory and practice of cost-sensitive verification** in **calf** or suggest new directions for research!

agda-calf

`yuesforest.com`

21 / 27

References I

[1] Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. "The Taming of the Rew: A Type Theory with Computational Assumptions". In: *Proceedings of the ACM on Programming Languages*. POPL 2021 (2021). URL: `https://hal.archives-ouvertes.fr/hal-02901011`.

22 / 27

References II

[2] Harrison Grodin and Robert Harper. "Amortized Analysis via Coinduction". In: *10th Conference on Algebra and Coalgebra in Computer Science (CALCO 2023)*. Ed. by Paolo Baldan and Valeria de Paiva. Vol. 270. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 23:1–23:6. ISBN: 978-3-95977-287-7. DOI: `10.4230/LIPIcs.CALCO.2023.23`. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CALCO.2023.23`.

23 / 27

References III

[3] Harrison Grodin, Yue Niu, Jonathan Sterling, and Robert Harper. "Decalf: A Directed, Effectful Cost-Aware Logical Framework". In: *Proceedings of the ACM on Programming Languages* 8.POPL (Jan. 2024). DOI: 10.1145/3632852.

[4] Jan Hoffmann. "Types with Potential: Polynomial Resource Bounds via Automatic Amortized Analysis". PhD thesis. Ludwig-Maximilians-Universität München, 2011. URL: https://www.cs.cmu.edu/~janh/assets/pdf/Hoffmann11.pdf.

References IV

[5] Paul Blain Levy. *Call-by-Push-Value: A Functional/Imperative Synthesis*. Kluwer, Semantic Structures in Computation, 2, Jan. 1, 2003. ISBN: 1-4020-1730-8.

[6] Runming Li, Harrison Grodin, and Robert Harper. *A Verified Cost Analysis of Joinable Red-Black Trees*. 2023. arXiv: 2309.11056.

[7] Yue Niu and Robert Harper. "A Metalanguage for Cost-Aware Denotational Semantics". In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2023, pp. 1–14. DOI: 10.1109/LICS56636.2023.10175777.

References V

[8] Yue Niu, Jonathan Sterling, Harrison Grodin, and Robert Harper. "A Cost-Aware Logical Framework". In: *Proceedings of the ACM on Programming Languages* 6.POPL (Jan. 2022). DOI: 10.1145/3498670. arXiv: 2107.04663 [cs.PL].

[9] Yue Niu, Jonathan Sterling, and Robert Harper. *Cost-sensitive computational adequacy of higher-order recursion in synthetic domain theory*. To appear, MFPS 2024. 2024. arXiv: 2404.00212 [cs.PL].

References VI

[10] Gordon D. Plotkin. "LCF considered as a programming language". In: *Theoretical Computer Science* 5.3 (1977), pp. 223–255. ISSN: 0304-3975. DOI: 10.1016/0304-3975(77)90044-5.

[11] Jonathan Sterling and Robert Harper. "Logical Relations as Types: Proof-Relevant Parametricity for Program Modules". In: *Journal of the ACM* 68.6 (Oct. 2021). ISSN: 0004-5411. DOI: 10.1145/3474834. arXiv: 2010.08599 [cs.PL].

Leanを用いたスイッチング回路の安全性検証

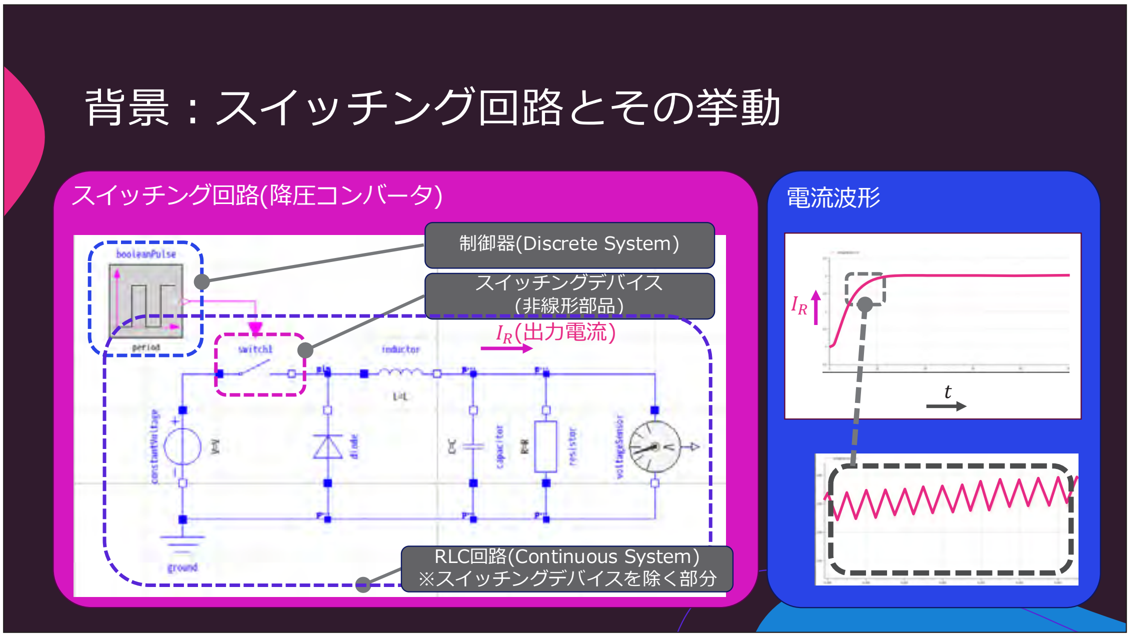
瀬川 秀一 ※1 ※2

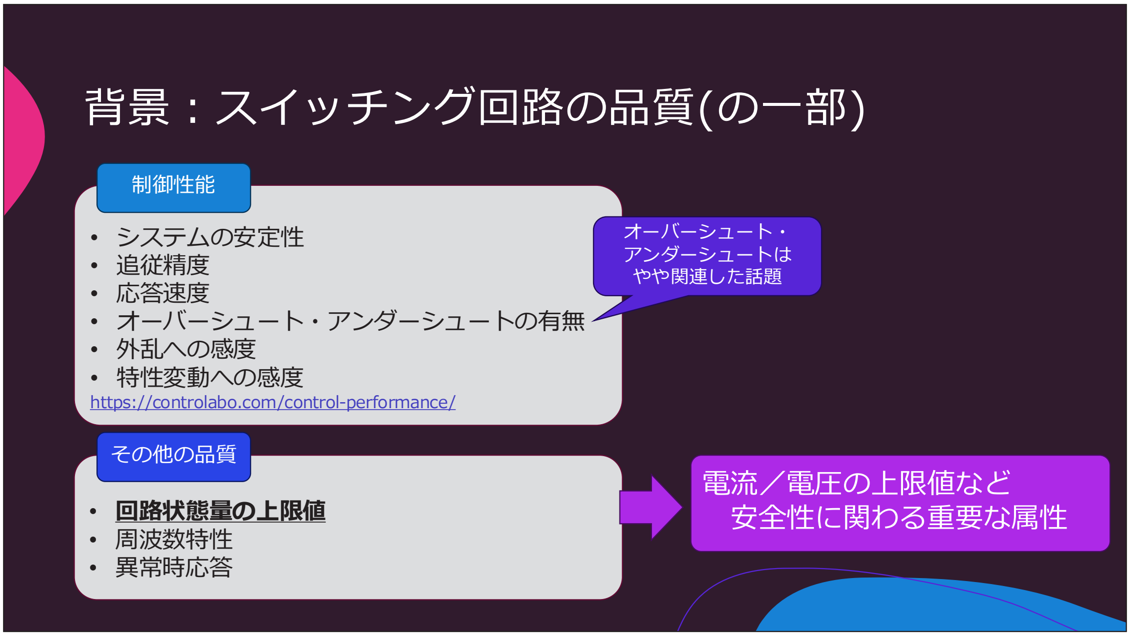
石井 大輔 ※1

※1 北陸先端科学技術大学院大学
※2 三菱電機株式会社

背景・動機

- スイッチング回路はアナログ回路の一種であり、充電器の電源回路などに使用される一般的な回路のカテゴリである。
- 電気回路の安全性の基準には各種規格標準や独自仕様に基づくものがある。いずれにせよ、製品には安全性のエビデンスが求められる。
- 網羅性の高い検証エビデンスとして定理証明支援系を用いた証明を用いる手法が考えられる。


背景：スイッチング回路とその挙動
スイッチング回路(降圧コンバータ)
制御器(Discrete System)
スイッチングデバイス
(非線形部品)
I_R(出力電流)
RLC回路(Continuous System)
※スイッチングデバイスを除く部分
電流波形
I_R
t


背景：スイッチング回路の品質(の一部)
制御性能
• システムの安定性
• 追従精度
• 応答速度
• オーバーシュート・アンダーシュートの有無
• 外乱への感度
• 特性変動への感度
https://controlabo.com/control-performance/
オーバーシュート・
アンダーシュートは
やや関連した話題
その他の品質
• 回路状態量の上限値
• 周波数特性
• 異常時応答
電流／電圧の上限値など
安全性に関わる重要な属性

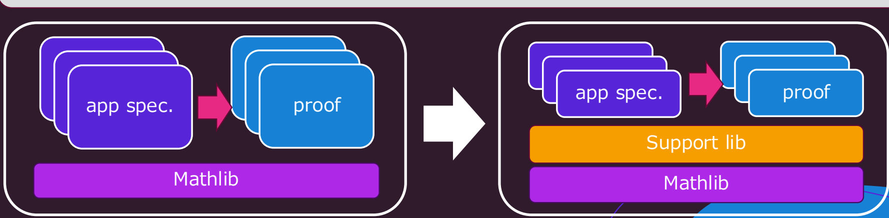
目的

目的

スイッチング回路の安全性を、Leanを用いて形式的に証明する。

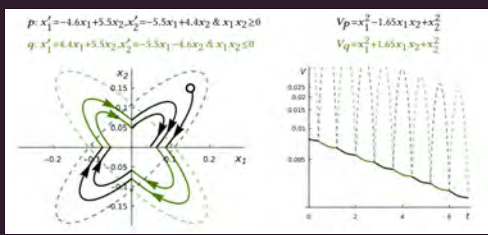
究極的には

電気回路製品の安全性の網羅的で効率的なエビデンスの作成プロセスを構築する。
- 検証サンプルの作成（←現在の作業）
- 必要な証明パターンの洗い出し
- 効率的に証明を行うためのライブラリ化

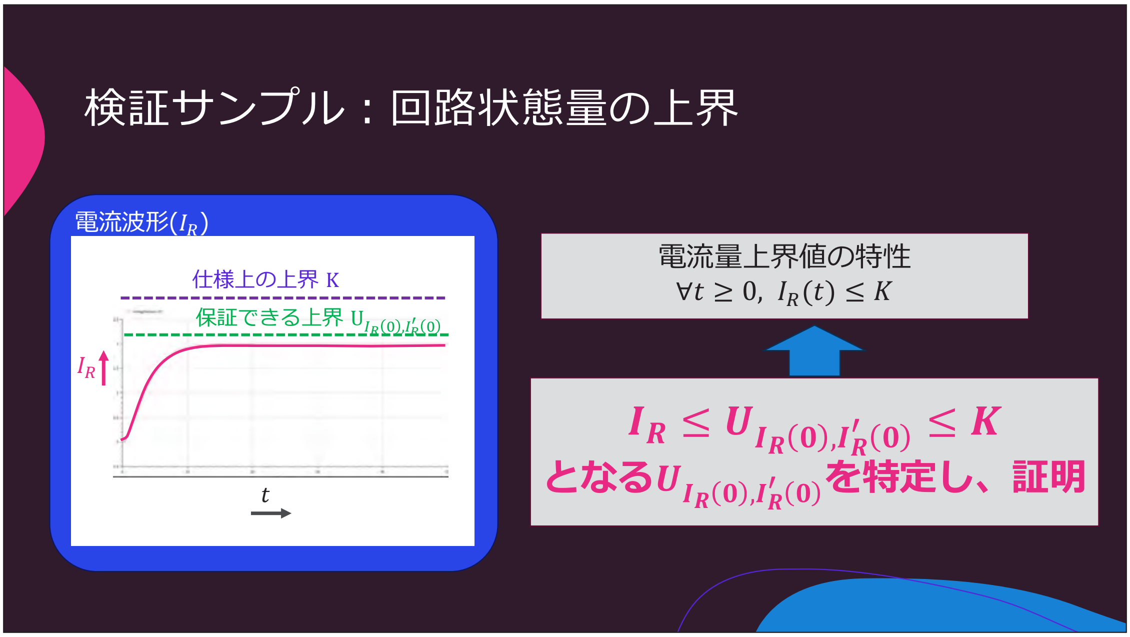


先行研究

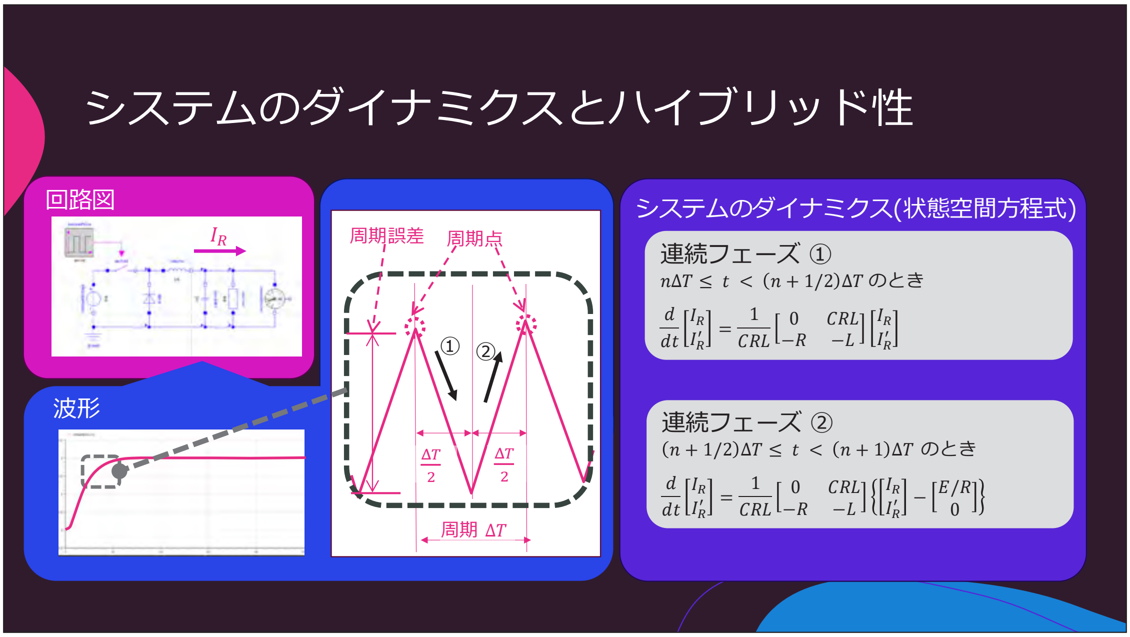
- Verifying Switched System Stability With Logic [Tan et al., 2022]
  - スイッチトシステムに対する定理証明
  - 定理証明支援系 KeYmaera Xを使用
  - 対象プロパティは安定性の証明



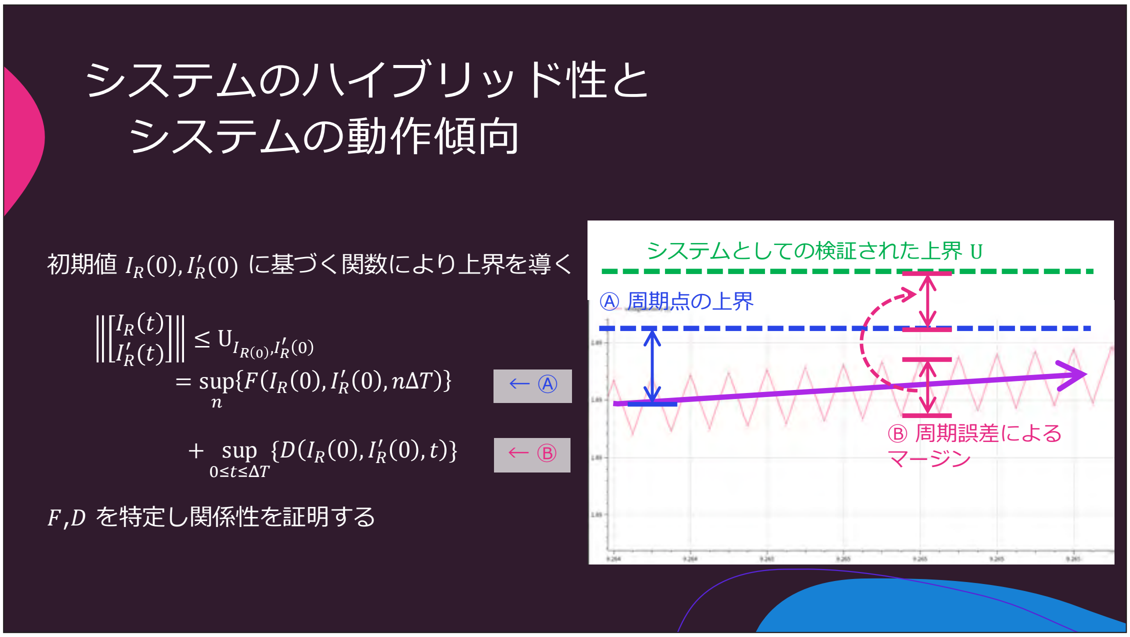
Y. K. Tan, S. Mitsch, and A. Platzer, “Verifying Switched System Stability With Logic,” in *25th ACM International Conference on Hybrid Systems: Computation and Control*, Milan Italy: ACM, May 2022, pp. 1–11. doi: 10.1145/3501710.3519541.


検証サンプル：回路状態量の上界
電流波形(I_R)
仕様上の上界 K
保証できる上界 $U_{I_R(0),I'_R(0)}$
I_R
t
電流量上界値の特性
$\forall t \geq 0,\ I_R(t) \leq K$
$I_R \leq U_{I_R(0),I'_R(0)} \leq K$
となる$U_{I_R(0),I'_R(0)}$を特定し、証明


証明のスケッチ

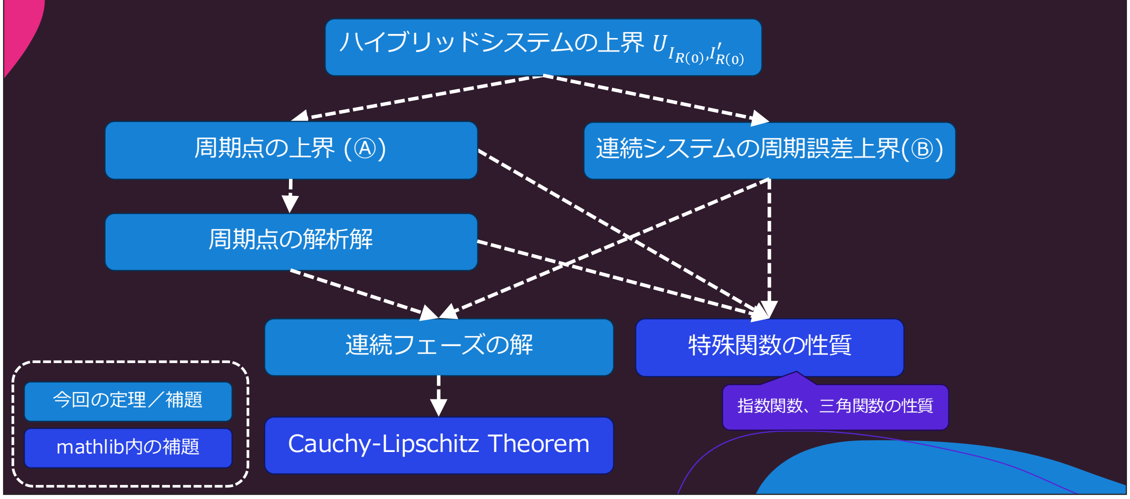


システムのダイナミクスとハイブリッド性
回路図
I_R
波形
周期誤差
周期点
①
②
$\frac{\Delta T}{2}$
$\frac{\Delta T}{2}$
周期 ΔT
システムのダイナミクス(状態空間方程式)
連続フェーズ ①
$n\Delta T \le t < (n+1/2)\Delta T$ のとき
$\frac{d}{dt}\begin{bmatrix} I_R \\ I'_R \end{bmatrix} = \frac{1}{CRL}\begin{bmatrix} 0 & CRL \\ -R & -L \end{bmatrix}\begin{bmatrix} I_R \\ I'_R \end{bmatrix}$
連続フェーズ ②
$(n+1/2)\Delta T \le t < (n+1)\Delta T$ のとき
$\frac{d}{dt}\begin{bmatrix} I_R \\ I'_R \end{bmatrix} = \frac{1}{CRL}\begin{bmatrix} 0 & CRL \\ -R & -L \end{bmatrix}\left\{\begin{bmatrix} I_R \\ I'_R \end{bmatrix} - \begin{bmatrix} E/R \\ 0 \end{bmatrix}\right\}$



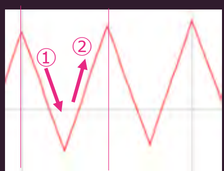
システムのハイブリッド性と
システムの動作傾向
初期値 $I_R(0), I'_R(0)$ に基づく関数により上界を導く
$\left\|\begin{bmatrix} I_R(t) \\ I'_R(t) \end{bmatrix}\right\| \le U_{I_R(0), I'_R(0)}$
$= \sup_n \{F(I_R(0), I'_R(0), n\Delta T)\}$
← Ⓐ
$+ \sup_{0 \le t \le \Delta T} \{D(I_R(0), I'_R(0), t)\}$
← Ⓑ
F, D を特定し関係性を証明する
システムとしての検証された上界 U
Ⓐ 周期点の上界
Ⓑ 周期誤差による
マージン

証明のスケッチ



連続フェーズ解

$L^2 - 4CR^2L > 0$ (強ダンプ条件)のとき $\psi = \frac{1}{2CR}, \phi = \frac{\sqrt{L^2-4CR^2L}}{2CRL}, \epsilon = \frac{E}{R}$ とすると



状態空間表現

① $\frac{d}{dt}\begin{bmatrix} I_R \\ I'_R \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\psi^2-\phi^2) & -2\psi \end{bmatrix}\begin{bmatrix} I_R \\ I'_R \end{bmatrix}$

② $\frac{d}{dt}\begin{bmatrix} I_R \\ I'_R \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\psi^2-\phi^2) & -2\psi \end{bmatrix}\left\{\begin{bmatrix} I_R \\ I'_R \end{bmatrix} - \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}\right\}$

解析解

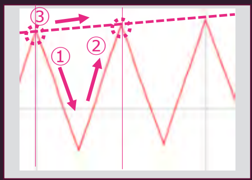
① $\begin{bmatrix} I_R \\ I'_R \end{bmatrix}(t+T) = \Psi_2(t)\begin{bmatrix} I_R \\ I'_R \end{bmatrix}(T)$

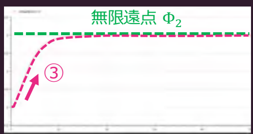
② $\begin{bmatrix} I_R \\ I'_R \end{bmatrix}(t+T) = \begin{bmatrix} \epsilon \\ 0 \end{bmatrix} + \Psi_2(t)\left\{\begin{bmatrix} I_R \\ I'_R \end{bmatrix}(T) - \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}\right\}$

$\psi > \phi > 0$ であり、$\lim_{t\to\infty} \Psi_2(t) = 0$

ただし、$\Psi_2(t) = \frac{e^{-\psi t}}{\phi}\begin{bmatrix} \phi\cosh\phi t + \psi\sinh\phi t & \sinh\phi t \\ -(\psi^2-\phi^2)\sinh\phi t & \phi\cosh\phi t - \psi\sinh\phi t \end{bmatrix}$

周期点の解析解





連続フェーズ解析解

① $\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(t+T) = \Psi_2(t)\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(T)$

② $\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(t+T) = \begin{bmatrix} \epsilon \\ 0 \end{bmatrix} + \Psi_2(t)\left\{\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(T) - \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}\right\}$

周期点解析解

③ $\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(n\Delta T+T) = \Phi_2 + \Psi_2(n\Delta T)\left\{\begin{bmatrix} I_R \\ I_R' \end{bmatrix}(T) - \Phi_2\right\}$

Leanによる形式化

ダイナミクスの定義

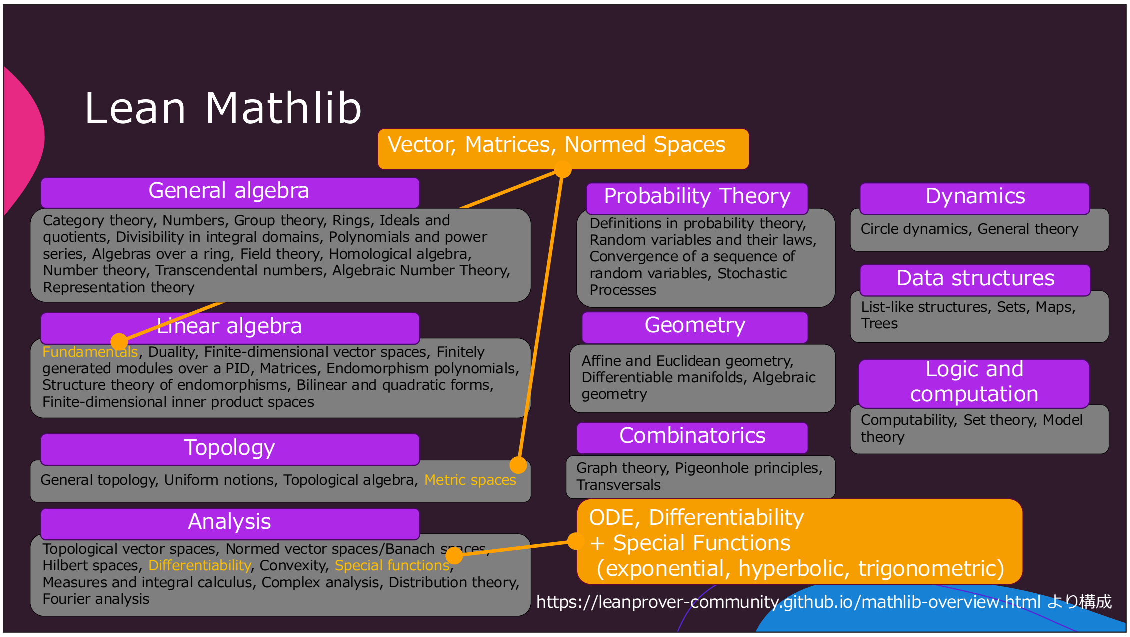
① $\frac{d}{dt}\begin{bmatrix} I_R \\ I_R' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\psi^2 - \phi^2) & -2\psi \end{bmatrix}\begin{bmatrix} I_R \\ I_R' \end{bmatrix}$ ② $\frac{d}{dt}\begin{bmatrix} I_R \\ I_R' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\psi^2 - \phi^2) & -2\psi \end{bmatrix}\left\{\begin{bmatrix} I_R \\ I_R' \end{bmatrix} - \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}\right\}$

```
variable {ir ir' : ℝ → ℝ}{ir0 ir'0 : ℝ}
    (hir0 : ir 0 = ir0)
    (hir'0 : ir' 0 = ir'0)
    (hird0 : ∀ n : ℕ,
        -- deriv (deriv ir) t + 2 * ψ * deriv ir t + (ψ ^ 2 - ϕ ^ 2) * ir t = 0
        let f : ℝ → Fin 2 → ℝ := fun t => ![ir t, ir' t]
        let v : ℝ → (Fin 2 → ℝ) → Fin 2 → ℝ := fun _ val =>
          ![
            ![ 0, 1],
            ![ - 2 * ψ, -(ψ ^ 2 - ϕ ^ 2) ]
          ] *ᵥ val
        ∀ t ∈ Ico (↑n * ΔT) ((↑n + 1/2) * ΔT), HasDerivWithinAt f (v t (f t)) (Ici t) t
    )
    (hird1 : ∀ n : ℕ,
        --  deriv (deriv ir) t + 2 * ψ * deriv ir t + (ψ ^ 2 - ϕ ^ 2) * (ir t - ε) = 0
        let f : ℝ → Fin 2 → ℝ := fun t => ![ir t, ir' t]
        let v : ℝ → (Fin 2 → ℝ) → Fin 2 → ℝ := fun _ val =>
          ![
            ![ 0, 1],
            ![ - 2 * ψ, -(ψ ^ 2 - ϕ ^ 2) ]
          ] *ᵥ (val - ![ε , 0])
        ∀ t ∈ Ico (↑n * ΔT) ((↑n + 1/2) * ΔT), HasDerivWithinAt f (v t (f t)) (Ici t) t
    )
```

定理の説明

$$\left\|\begin{bmatrix} I_R(t) \\ I_R'(t) \end{bmatrix}\right\| \leq \sup_n\{F(I_R(0), I_R'(0), n\Delta T)\} + \sup_{0 \leq t \leq \Delta T}\{D(I_R(0), I_R'(0), t)\}$$

```
/-
  ode_overall_evaluation_03
-/
lemma ode_overall_evaluation_03
  {t : ℝ}
  :
  ‖![ir t, ir' t]‖₊
  ≤ sSup ((λ n => (‖(@Φ₂ ψ ϕ ΔT ε + (@Ψ₂ ψ ϕ (n * ΔT)) *ᵥ (![ir (0), ir' (0)] - @Φ₂ ψ ϕ ΔT ε))‖₊
    + sSup ((λ δ => ‖(I₂ - @Ψ₂ ψ ϕ δ) *ᵥ (@Φ₂ ψ ϕ ΔT ε)‖₊) '' (Icc 0 ΔT))
    + sSup ((λ δ => ‖(I₂ - @Ψ₂ ψ ϕ δ) *ᵥ (@Φ₂ ψ ϕ ΔT ε + (@Ψ₂ ψ ϕ (n * ΔT)) *ᵥ (![ir (0), ir' (0)] - @Φ₂ ψ ϕ ΔT ε))‖₊)
      '' (Icc 0 ΔT))))
      '' {n | n : ℕ}
      )
```



Cauchy-Lipschitz Theorem

Lipschitz連続性を満たすODEの解は一意である
→ ダイナミクスと初期条件が一致すれば解が一致することを保証
→ すなわち、「自分が与えた解が正しいことをLeanに解らせることができる」

Mathlib.Analysis.ODE.Gronwall

```
variable {E : Type*} [NormedAddCommGroup E] [NormedSpace ℝ E]
variable {v : ℝ → E → E} {K : ℝ≥0} {f g : ℝ → E} {a b t₀ : ℝ}

/-- There exists only one solution of an ODE \(\dot x=v(t, x)\) with
a given initial value provided that the RHS is Lipschitz continuous in `x`. -/
theorem ODE_solution_unique
    (hv : ∀ t, LipschitzWith K (v t))
    (hf : ContinuousOn f (Icc a b))
    (hf' : ∀ t ∈ Ico a b, HasDerivWithinAt f (v t (f t)) (Ici t) t)
    (hg : ContinuousOn g (Icc a b))
    (hg' : ∀ t ∈ Ico a b, HasDerivWithinAt g (v t (g t)) (Ici t) t)
    (ha : f a = g a) :
    EqOn f g (Icc a b)
```

状態空間方程式に関する補題

Cauchy-Lipschitz Theoremは一般ノルム空間について示されている
→ 制御理論でよく使用する状態空間方程式を扱うには固定次元ベクトル空間＋行列の組み合わせで便利に使用する方法を確立したい

```
/-
  ODE_solution_unique_statespace_sq
  This is a theorem that ensures the solution of state space expression
    x' = A x + B
  to be unique in 2-dimension real vector spaces.
-/
theorem ODE_solution_unique_statespace_sq
  {x y: ℝ → (Fin 2) → ℝ}
  {t0 t1: ℝ}
  {A : Matrix (Fin 2) (Fin 2) ℝ}
  {B : Fin 2 → ℝ}
  (hzero : x t0 = y t0)
  (hx_state : ∀ t ∈ Ico t0 t1, HasDerivWithinAt x (A *ᵥ x t + B) (Ici t) t)
  (hy_state : ∀ t ∈ Ico t0 t1, HasDerivWithinAt y (A *ᵥ y t + B) (Ici t) t)
  (hx_cont : ContinuousOn x (Icc t0 t1))
  (hy_cont : ContinuousOn y (Icc t0 t1))
  :
  EqOn x y (Icc t0 t1)
```

一様Lipschitz連続性を状態空間方程式に置換

考察・まとめ

作業の進行状況

- 検証サンプルの作成
  - → ローカル解の構築について
    - 状態空間方程式の解の唯一性補題を証明済み
    - 実際の解への適用を試行中
  - → その他、不等式の充足について
    - 微分方程式の極値の取り扱いなどが主な課題
- 必要な証明パターンの洗い出し
- 効率的に証明を行うためのライブラリ化

定理証明支援系の選択

KeYmaera X

https://keymaerax.org/

特徴：
- ハイブリッドシステム特化型の定理証明支援系
- dL(differential dynamic logic)を基礎理論とする
- FOL + dynamic logic + HP(Hybrid Program)

Pros
- HPにおいてODE(常微分方程式)を構文としてサポート
- 自動限量子消去(QE)やMathematica連携が組み込み済み
- ハイブリッドシステムに特化した半自動の不変量探索
- 基本的にわからないことはA. Platzer氏に聞いて解決

Cons
- ベースがFOLのため、式表現が複雑になりがちである(ベクトル表現などが使用できない)
- Mathlibと比較すると定理ライブラリが小さい

Lean

https://lean-lang.org/

特徴：
- 汎用定理証明支援系
- CoC(Caliculus of Construction)を基礎理論とする
- 高階論理、依存型

Pros
- 高階論理による高い記述自由度
- 定理／Corpus／Tactics／ツールの充実(含Github Copilot)
- Zulip Chatの返答が速い
- 利用者が多いため、同じことをしても貢献度が高い

Cons
- ハイブリッドシステムの標準的モデリングスキームがない

「手さぐりで証明を探す」フェーズではLeanがマッチすると判断

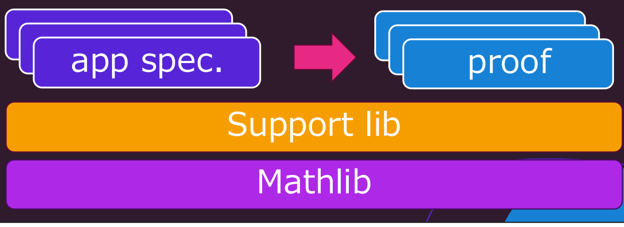
製品検証向けライブラリの必要性

Mathlibなどで見られる証明

・できるだけ抽象度の高い証明を求める

製品検証における定理証明

・類似した証明を製品毎に作る必要がある
・式の規模が増えがち、パターンマッチに失敗しがち
→ アダプテーションのための補題／タクティクスのライブラリ整備が必要



まとめ

- スイッチング回路の検証に客観性／網羅性をもたらすため、Leanを用いた検証を進めている
  - 証明のスケッチを定め、必要な定理を特定した
  - Mathlib上で対象システムのモデリングを実施した
  - Mathlibの関連機能および追加すべき補題を特定し証明した
    - 状態空間方程式の解の一意性に関する補題を証明した
- 今後、製品の性質を検証するために有用なライブラリを構築したい

導出原理の逆適用による定理の自動生成手法の提案

西島 海斗
山口大学

目次 2

研究課題: 学習用の形式化数学ライブラリの不足

機械学習の発展により，自動定理証明(ATP)の性能は向上した

既存のATPは，人間が作成した形式化数学ライブラリを学習している

しかし，学習データの不足が性能向上の妨げとなっている

研究目的: 形式化数学ライブラリの自動生成

質の高い学習データを機械的に大量に生成することを目指す

形式化数学ライブラリの自動生成

先行研究は主に4つの手法に分類される

1. ルールベース
2. 反復的方法
3. 補題作成
4. オートフォーマライゼーション(自動形式化)

1. ルールベースによる生成

推論規則をもとに新しい定理や証明を生成する方法

研究事例	特徴
MetaGen (Wang & Deng, 2020)	ニューラルネットワークを用いて，人間が書いたMetaMathの直観主義論理の定理と証明を学習し，新たに定理と証明を生成
INT(Wu et al, 2021)	論理命題にランダムに指定した公理を順次適用し，新しい定理と証明手順を生成
LIMU(Wu et al, 2021)	数学記号，定数，変数等をランダムに生成して，帰納，演繹，アブダクションを適用することにより問題を生成
FwdP(Firoiu et al, 2021)	**TPTPの公理に線形導出を用いて新しい定理を生成**
AIPS(Wei et al, 2024)	ランダムに生成した対象式に不等式や式変形を施して，定理を生成
AlphaGeometry (Trinh et al, 2024)	**モデル学習のためにユークリッド幾何学の公理を用いて定理と証明を生成**
PropL(An et al, 2024)	Lean上で命題論理の定理とともに失敗例を含む証明を生成

2. 反復的方法による生成

ATPに定理を証明させて，証明に成功した定理と証明を学習データに加える．このループを繰り返すことによって，学習データを増やす方法

研究事例	特徴
DeepHOL(Bansal et al, 2019)	ATPのベンチマーク用に複素解析やケプラー予想に関する定理と補題29,462件のHOL Lightのデータセットを作成
GPT-f(Polu & Sutskever, 2020)	**ランダムに生成した式に対して算術や環代数の規則を適用して学習データを作成し，証明に成功したデータをMetaMathのデータセットに追加**
HER(Aygun et al, 2022)	証明が失敗した場合に，証明過程で生成された中間結果を新しい学習データに加える方法を提案
FMSCL(Polu et al, 2023)	miniF2Fのデータセットをモデルに証明させ，証明が完了した定理を学習データに追加

3. 補題作成による生成

証明の途中で生成した補題を学習データに追加する方法

研究事例	特徴
LEGO-Prover(Xin et al, 2024)	**LLMにminiF2Fの自然言語の証明から補題を生成させ，補題をIsabellの形式に変換して学習データを生成**
REFACTOR (Zhou et al, 2024)	ニューラルネットワークを用いて，証明から有用な補題を抽出 MetaMathのライブラリset.mmに適用したところ，新たに16個の補題が抽出された
ATG(Lin et al, 2024)	モンテカルロ木探索と強化学習を組み合わせて，MetaMath形式で補題を作成

4. オートフォーマライゼーションによる生成

LLMを用いて，自然言語で書かれた数学記述を形式言語に変換することにより，学習データを生成する方法

研究事例	特徴
MUSTARD (Huang et al, 2024)	LLMとLean Proverを活用し，5,866問の問題と証明ステップごとの解答を含むデータセットを構築
DeepSeek-Prover (2024)	**自然言語の高校，大学レベルの数学の問題をLean4に変換して，869,659問の大規模なデータセットを構築**
Lean Workbook (Ying et al, 2024)	MiniF2F, ProofNetなどの自然言語の数学問題をLeanに形式化して57,231問(17.5%)のデータセットを構築

導出原理を逆適用して，学習データを自動生成

ルールベースの新しい生成手法として，導出原理を逆向きに適用する

導出原理を採用した理由:

- 導出原理の健全性から，導出原理の逆適用によって得られる論理式は全て定理となる
- 導出原理の完全性から，導出原理の逆適用によって任意の(節形式の)定理が導かれる

一階述語論理における導出原理

導出原理のアルゴリズム

1. 仮定と結論の否定をスコーレム化して，連言標準形に直し，節集合を構築
2. 節集合から互いに否定し合うリテラルを持つ2つの節を探索
3. リテラルに単一化アルゴリズムを適用し，リテラルを統一
4. 統一したリテラルを導出規則をもとに削除し，残りのリテラルで新しい節を生成
5. 生成した節を節集合に追加し，上記の処理を繰り返す
6. 矛盾(□)が生成されれば，元の節集合は充足不能

導出規則: $$\frac{l \lor P, \sim l' \lor Q}{\sigma P \lor \sigma Q} \sigma \text{ where } \frac{\sigma l \lor \sim \sigma l'}{\top}$$

定理の反駁を節集合に変換し，導出規則を繰り返し適用することで矛盾を導き出す.

導出原理の具体例 その1

仮定と結論の否定をスコーレム化して，連言標準形に直し，節集合を構築

↑ 定理の反駁を取り，ここから矛盾を示す

仮定と結論

仮定:
1.(∀x)(P(x)→(S(x)∧R(x)))

2. (∃x)(P(x)∧Q(x))

結論:
(∃x)(Q(x)∧R(x))

⇒ 結論の否定
∃の削除

(∀x)(~P(x)∨S(x))
(∀x)(~P(x)∨R(x))

P(a)
Q(a)

(∀x)(~Q(x)∨~R(x))

⇒

節集合

C_1 = ~P(x)∨S(x)
C_2 = ~P(x)∨R(x)

C_3 = P(a)
C_4 = Q(a)

C_5 = ~Q(x)∨~R(x)

導出原理の具体例 その2

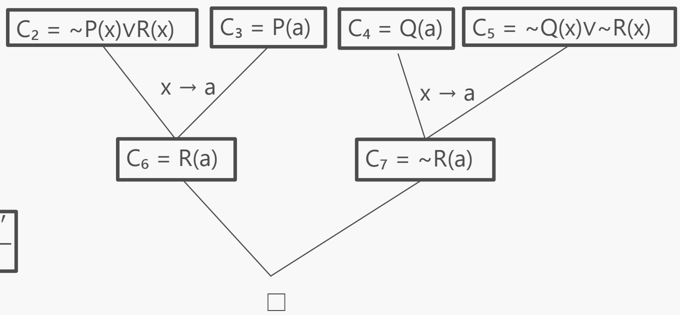
節集合

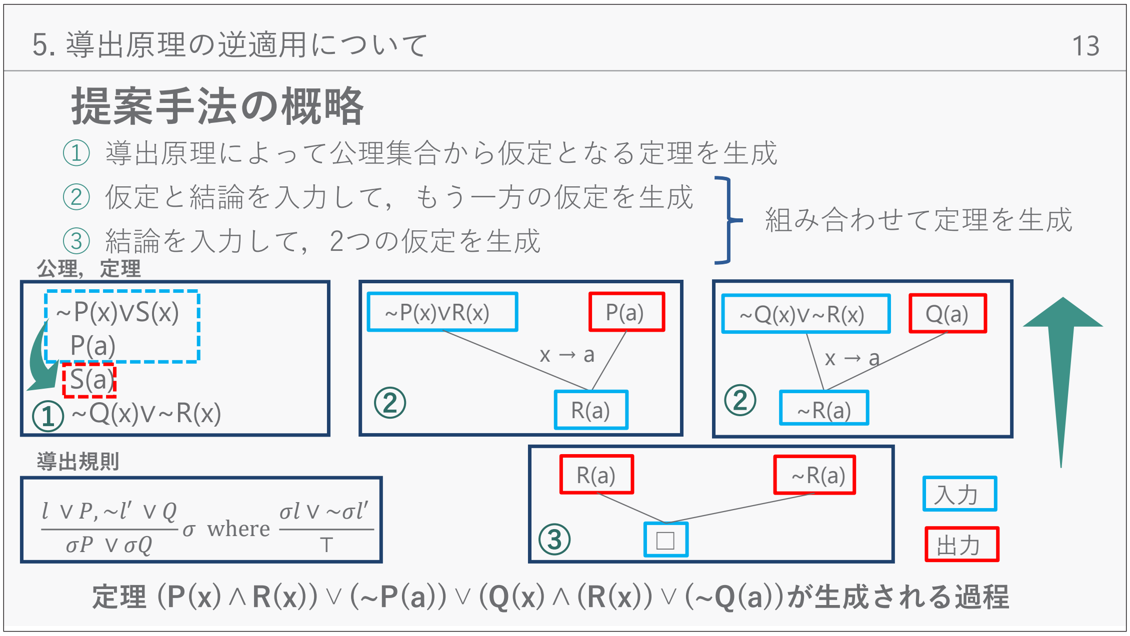
C_1 = ~P(x)∨S(x)
C_2 = ~P(x)∨R(x)
C_3 = P(a)
C_4 = Q(a)
C_5 = ~Q(x)∨~R(x)

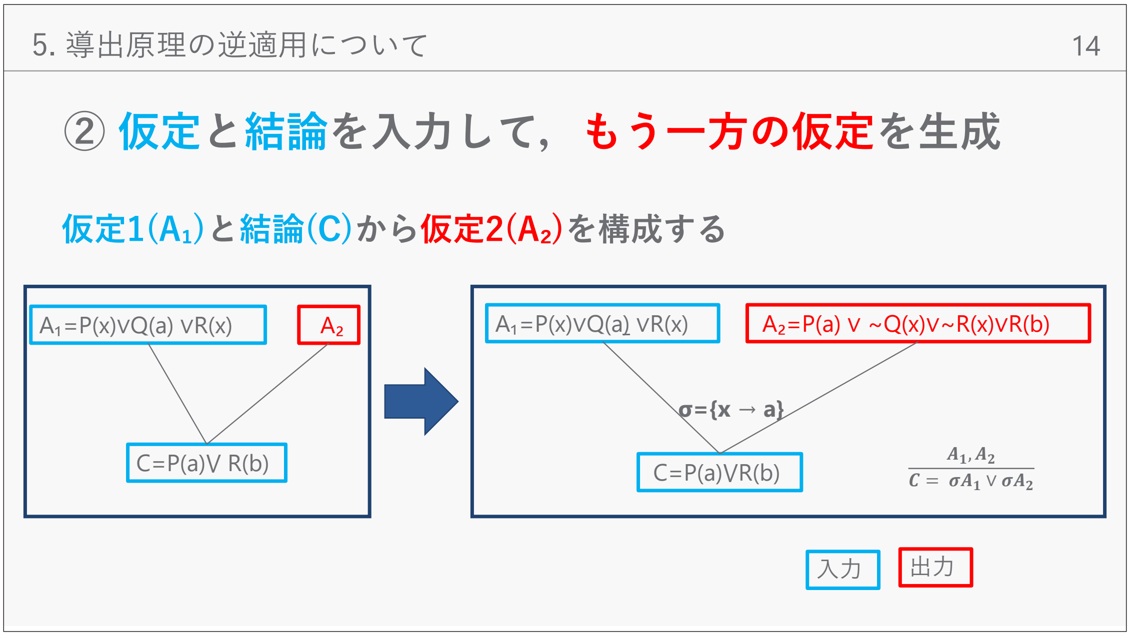
導出規則

$$\frac{l \vee P, \sim l' \vee Q}{\sigma P \vee \sigma Q}\sigma \text{ where } \frac{\sigma l \vee \sim \sigma l'}{\top}$$

矛盾(□)を導けたら証明完了




5. 導出原理の逆適用について
13
提案手法の概略
① 導出原理によって公理集合から仮定となる定理を生成
② 仮定と結論を入力して，もう一方の仮定を生成
③ 結論を入力して，2つの仮定を生成
組み合わせて定理を生成
公理，定理
~P(x)VS(x)
P(a)
S(a)
① ~Q(x)V~R(x)
~P(x)VR(x)
P(a)
x → a
② R(a)
~Q(x)V~R(x)
Q(a)
x → a
② ~R(a)
導出規則
$\frac{l \vee P, \sim l' \vee Q}{\sigma P \vee \sigma Q} \sigma$ where $\frac{\sigma l \vee \sim \sigma l'}{\top}$
R(a)
~R(a)
③ □
入力
出力
定理 (P(x)∧R(x))∨(~P(a))∨(Q(x)∧(R(x))∨(~Q(a))が生成される過程


5. 導出原理の逆適用について
14
② 仮定と結論を入力して，もう一方の仮定を生成
仮定1(A_1)と結論(C)から仮定2(A_2)を構成する
A_1=P(x)VQ(a) VR(x)
A_2
C=P(a)V R(b)
A_1=P(x)VQ(a) VR(x)
A_2=P(a) V ~Q(x)V~R(x)VR(b)
σ={x → a}
C=P(a)VR(b)
$\frac{A_1, A_2}{C = \sigma A_1 \vee \sigma A_2}$
入力
出力

5. 導出原理の逆適用について

15

具体例②-a. 仮定: $A_1 = P(x) \vee Q(a) \vee R(x)$, 結論: $C = P(a) \vee R(b)$

手順1. 仮定A_1から結論Cへ置換可能なリテラルのペアとその置換を列挙する
ただし，Cのリテラルを書き換える置換は許容しない.

仮定: $A_1 = P(x) \vee Q(a) \vee R(x)$

結論: $C = P(a) \vee R(b)$

ケース1. $\{P(x), P(a)\}$, $\sigma_1 = \{x \to a\}$

ケース2. $\{R(x), R(b)\}$, $\sigma_2 = \{x \to b\}$

手順2. 手順1で得られたリテラルのペアに含まれるCのリテラル集合から
0個または1個のリテラルを選択する.
ここでは，ケース1の$P(a)$を選択する.

$C = P(a) \vee R(b)$ $\sigma_1 = \{x \to a\}$

$A_1 = X \vee Y$
$A_2 = X \vee \sim Y \vee Z$
$C = X \vee Z$

5. 導出原理の逆適用について

16

手順3. A_1とCのリテラル集合をX, Y, Zに分割する．ただし，手順2で選択したリテラルをXとする.

X: $P(x)$, $P(a)$; Y: $Q(a) \vee R(x)$; Z: $R(b)$

$A_1 = P(x) \vee Q(a) \vee R(x)$
$C = P(a) \vee R(b)$

$A_1 = X \vee Y$
$A_2 = X \vee \sim Y \vee Z$
$C = X \vee Z$

手順4. 単一化によって手順2で選択したリテラルに一致するように
A_2に属するリテラルとmguを決定する.

$A_1 = P(x) \vee Q(a) \vee R(x)$
$A_2 = P(a)$
$C = P(a) \vee R(b)$

$\sigma = \{x \to a\}$

手順5. 置換σを利用して，仮定A_2を構成する．

Y，Zに関しては，σによる置換を行った後の項が一致する範囲内で選択する．

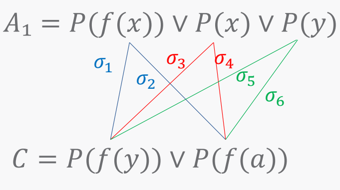
	X		Y		Z
$A_1 =$	$P(x)$	$\vee$	$Q(a) \vee R(x)$		
$A_2 =$	$P(a)$	$\vee$	$\sim Q(x) \vee \sim R(x)$	$\vee$	$R(b)$
$C =$	$P(a)$			$\vee$	$R(b)$

$\sigma = \{x \to a\}$

$A_1 = X \vee Y$
$A_2 = X \vee \sim Y \vee Z$
$C = X \vee Z$

具体例②-b. 仮定: $A_1 = P(f(x)) \vee P(x) \vee P(y)$, **結論**: $C = P(f(y)) \vee P(f(a))$

手順1.仮定A_1から結論Cへ置換可能なリテラルのペアとその置換を列挙する
ただし，Cのリテラルを書き換える置換は許容しない．



ケース1. $\{P(f(x)), P(f(y))\}, \sigma_1 = \{x \to y\}$
ケース2. $\{P(f(x)), P(f(a))\}, \sigma_2 = \{x \to a\}$
ケース3. $\{P(x), P(f(y))\}, \sigma_3 = \{x \to f(y)\}$
ケース4. $\{P(x), P(f(a))\}, \sigma_4 = \{x \to f(a)\}$
~~ケース5. $\{P(y), P(f(y))\}, \sigma_5 = \{y \to f(y)\}$~~
~~ケース6. $\{P(y), P(f(a))\}, \sigma_6 = \{y \to f(a)\}$~~
→ Cのリテラルが書き換わるのでNG

5. 導出原理の逆適用について　19

手順2. 手順1で得られたリテラルのペアに含まれるCのリテラル集合から 0個または1個のリテラルを選択する.

ここでは，ケース1の$P(f(y))$を選択する.

$C = P(f(y)) \vee P(f(a))$　　$\sigma_1 = \{x \rightarrow y\}$

手順3. A_1とCのリテラル集合をX, Y, Zに分割する．ただし，手順2で選択したリテラルをXとする.

	X	Y	Z
$A_1 =$	$P(f(x))$	$\vee\ P(x) \vee P(y)$	
$C =$	$P(f(y))$		$\vee\ P(f(a))$

$A_1 = X \vee Y$

$A_2 = X \vee {\sim}Y \vee Z$

$C = X \vee Z$

5. 導出原理の逆適用について　20

手順4. 単一化によって手順2で選択したリテラルに一致するように A_2に属するリテラルとmguを決定する.

	X	Y	Z
$A_1 =$	$P(f(x))$	$\vee\ P(x) \vee P(y)$	
$A_2 =$	$P(f(y))$		
$C =$	$P(f(y))$		$\vee\ P(f(a))$

$\sigma = \{x \rightarrow y\}$

$A_1 = X \vee Y$

$A_2 = X \vee {\sim}Y \vee Z$

$C = X \vee Z$

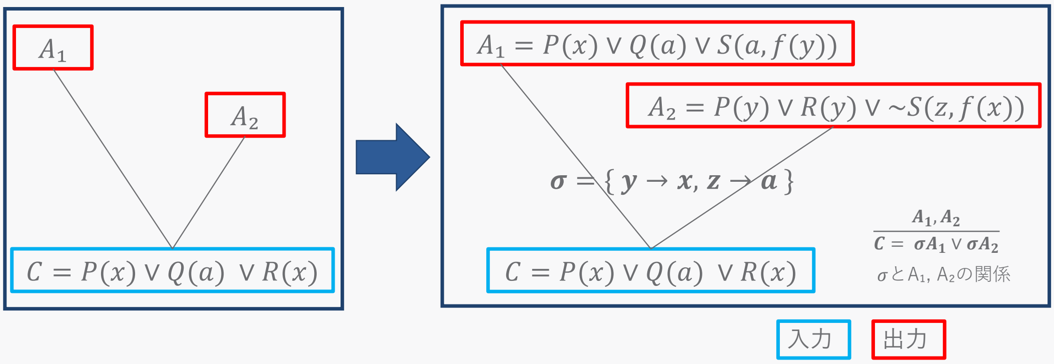
手順5. 置換σを利用して，仮定A_2を構成する.

Y, Zに関しては，σによる置換を行った後の項が一致する範囲内で選択する.

	X	Y	Z
$A_1 =$	$P(f(x))$	$\vee\ P(x) \vee P(y)$	
$A_2 =$	$P(f(y))$	$\vee\ {\sim}P(y)$ ~~$\vee\ {\sim}P(y)$~~	$\vee\ P(f(a))$
$C =$	$P(f(y))$		$\vee\ P(f(a))$

3. 結論を入力して，2つの仮定を生成

結論(C)から仮定1(A_1)と仮定2(A_2)を構成する



具体例③-a. 結論: $C = P(x) \lor Q(a) \lor R(x)$

手順1. 結論Cと公理から述語，関数，定数を収集する．

- $\boldsymbol{P}$：収集した述語の集合
- $\boldsymbol{F}$：収集した関数と定数の集合

$P = \{P, Q, R, S\}$, $F = \{f, a\}$とする．

$A_1 = X \lor Y \lor W$
$A_2 = X \lor Z \lor \sim W$
$C = X \lor Y \lor Z$

手順2. Cのリテラル集合から0個または1個のリテラルを選択する．

ここではP(x)を選択する．

$C = \boxed{\boldsymbol{P(x)}} \lor Q(a) \lor R(x)$

手順3. Cのリテラル集合をX, Y, Zに分割する．ただし，手順2で選択したリテラルはXに含めるものとする．

$C = \underbrace{\boxed{\boldsymbol{P(x)}}}_{X} \lor \underbrace{\boxed{Q(a)}}_{Y} \lor \underbrace{\boxed{R(x)}}_{Z}$

具体例③-a. 結論: $C = P(x) \vee Q(a) \vee R(x)$

手順4-a. 単一化によって手順2で選択したリテラルに一致するように A_1, A_2に属するリテラルとmguを決定する.

X Y Z

$A_1 = P(x)$

$A_2 = P(y)$

$C = \boldsymbol{P(x)} \vee Q(a) \vee R(x)$

$\boldsymbol{\sigma = \{y \rightarrow x\}}$

$A_1 = X \vee Y \vee W$

$A_2 = X \vee Z \vee \sim W$

$C = X \vee Y \vee Z$

$$\frac{\boldsymbol{A_1, A_2}}{\boldsymbol{C = \sigma A_1 \vee \sigma A_2}}$$

手順5-a. 置換σを利用して, 仮定A_1, A_2を構成する.

Y, Zに関しては, σによる置換を行った後の項が一致する範囲内で選択する.

$A_1 = P(x) \vee Q(a)$

$A_2 = P(y) \vee R(x)$

$C = \boldsymbol{P(x)} \vee Q(a) \vee R(x)$

具体例③-b. 結論: $C = P(x) \vee Q(a) \vee R(x)$

手順1. 結論Cと公理から述語, 関数, 定数を収集する.

- $\boldsymbol{P}$: 収集した述語の集合
- $\boldsymbol{F}$: 収集した関数と定数の集合

$P = \{P, Q, R, S\}$, $F = \{f, a\}$とする.

$A_1 = X \vee Y \vee W$

$A_2 = X \vee Z \vee \sim W$

$C = X \vee Y \vee Z$

手順2. Cのリテラル集合から0個または1個のリテラルを選択する.

ここでは選択しない.

$C = P(x) \vee Q(a) \vee R(x)$

手順3. Cのリテラル集合をX, Y, Zに分割する. ただし, 手順2で選択したリテラルはXに含めるものとする.

X Y Z

$C = P(x) \vee Q(a) \vee R(x)$

具体例③-b．結論: $C = P(x) \lor Q(a) \lor R(x)$

手順4-b.単一化によって打ち消し合うようなA₁，A₂に属するリテラルとmgu を決定する．ただし，Cのリテラルを書き換える置換は許容しない．

	X		Y		Z		W
$A_1 =$							$S(a, f(y))$
$A_2 =$							$\sim S(z, f(x))$
$C =$	$P(x)$	$\lor$	$Q(a)$	$\lor$	$R(x)$		

$A_1 = X \lor Y \lor W$
$A_2 = X \lor Z \lor \sim W$
$C = X \lor Y \lor Z$

$\boldsymbol{\sigma = \{y \to x, z \to a\}}$

$$\frac{A_1, A_2}{C = \sigma A_1 \lor \sigma A_2}$$

手順5-b. 置換σを利用して，仮定A_1, A_2を構成する．

X, Y, Zに関しては，σによる置換を行った後の項が一致する範囲内で選択する．

$A_1 = P(x) \lor Q(a) \lor S(a, f(y))$
$A_2 = P(y) \lor R(x) \lor \sim S(z, f(x))$
$C = P(x) \lor Q(a) \lor R(x)$

残課題

- 導出原理の逆適用のアルゴリズムを実装
- 上記を使用して，学習用データセットを作成
- データセットの品質を計測する枠組みの検討

Coq 証明支援系による DNA 計算の形式化

早川　銀河

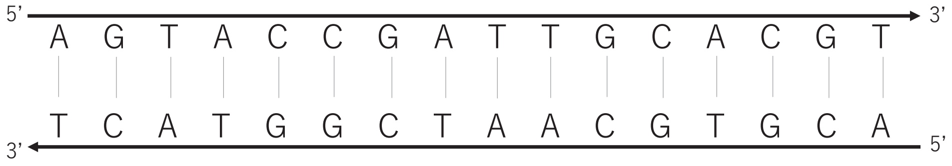
九州大学　数理学府

November, 26, 2024

目次

1.1　DNA

- 塩基配列により文字情報を保持できる生体高分子
- 二本の相補的なDNA鎖は水素結合により二重らせん構造をとる
- 結合や書き換えなどの演算が充実しており、計算に応用できる



1.2　DNAコンピューティング

DNAの塩基配列を文字列情報として扱い、結合、切断、複製などの分子生物学的操作により計算を行う。

1994年にAdlemanがハミルトン閉路問題をDNAの結合操作で解いたことにより注目が集まった。

PCRによる複製が容易で並列性に優れているほか、SSDの10^7倍ほどの高い情報密度[4]など、DNAの持つ様々な性質によりナノマシンや生体内コンピュータへの応用が期待されている。

2.1　DNAドミノ

- DNA の二本鎖構造を参考にして設計された分子装置
- 中央部分で相補的に結合する二本鎖と、両端に露出した粘着末端からなる

$$L_\rho(V)=\left(\binom{V^*}{\lambda}\cup\binom{\lambda}{V^*}\right)\begin{bmatrix}V^*\\V^*\end{bmatrix}_\rho,\quad R_\rho(V)=\begin{bmatrix}V^*\\V^*\end{bmatrix}_\rho\left(\binom{V^*}{\lambda}\cup\binom{\lambda}{V^*}\right),\quad LR_\rho(V)=\overbrace{\left(\binom{V^*}{\lambda}\cup\binom{\lambda}{V^*}\right)}^{\text{粘着末端}}\overbrace{\begin{bmatrix}V^+\\V^+\end{bmatrix}_\rho}^{\text{二本鎖部}}\overbrace{\left(\binom{V^*}{\lambda}\cup\binom{\lambda}{V^*}\right)}^{\text{粘着末端}}$$

$$\mathfrak{D}_\rho(V)=L_\rho(V)\cup R_\rho(V)\cup LR_\rho(V)$$

V：アルファベット　　V^*：クリーネ閉包(+は非空)

λ：空文字　　$\rho\subseteq V\times V$：相補的対応関係

2.2　粘着演算

- DNAドミノの粘着末端が相補的関係にあるとき、ドミノ同士で結合する

$$\mu:\mathfrak{D}\times\mathfrak{D}\to\mathfrak{D}\cup\{\perp\}$$

例）
$$\mu\left(\binom{A}{\lambda}\begin{bmatrix}T\\A\end{bmatrix}_\rho\binom{GC}{\lambda},\binom{\lambda}{CG}\begin{bmatrix}G\\C\end{bmatrix}_\rho\right)=\binom{A}{\lambda}\begin{bmatrix}TGCG\\ACGC\end{bmatrix}_\rho$$

- DNA分子の物理的性質から、結合演算は交換法則を満たす。

$$^\forall x,y,z\in\mathfrak{D},\qquad \mu(x,\mu(y,z))=\mu(\mu(x,y),z)$$

2.3　スティッカーシステム

スティッカーシステム：$\gamma = (V, \rho, S, D)$

V：記号の有限集合　　　　$\rho \subseteq V \times V$

$S \subset LR_\rho(V)$(有限集合)　　　　$D \subseteq \mathfrak{D}_\rho(V) \times \mathfrak{D}_\rho(V)$(有限集合)

スティッカーシステムの生成言語

$$L(\gamma) = \left\{ w \in V^* \middle| \begin{bmatrix} w \\ w' \end{bmatrix}_\rho \in \mu_D^n(S), n \in \mathbb{N} \right\}$$

$$\mu_D(X) \coloneqq \{\mu(l, \mu(x, r)) | x \in X, (l, r) \in D\}$$

スティッカーシステム(例)

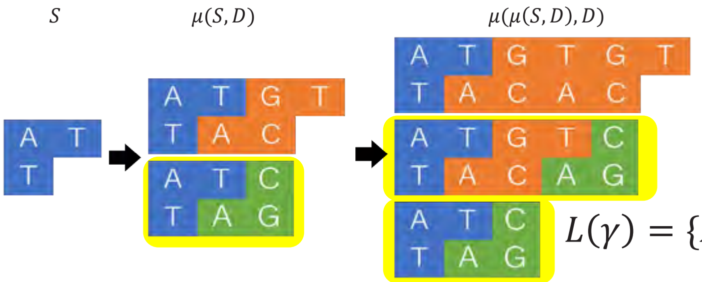
$V = \{A, T, G, C\}$

$\rho = \{(A, T), (T, A), (G, C), (C, G)\}$

$S = \left\{\begin{bmatrix} A \\ T \end{bmatrix}\begin{pmatrix} T \\ \lambda \end{pmatrix}\right\}$

$D = \left\{\left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ A \end{pmatrix}\begin{bmatrix} G \\ C \end{bmatrix}\begin{pmatrix} T \\ \lambda \end{pmatrix}\right), \left(\begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ A \end{pmatrix}\begin{bmatrix} C \\ G \end{bmatrix}\right)\right\}$

$\gamma = (V, \rho, S, D)$



$L(\gamma) = \{AT(GT)^n C | n \in \mathbb{N}\}$

2.4　オートマトン vs スティッカーシステム

(右側)スティッカーシステムは有界遅延の原始的な計算で正則文法と同じ生成能力を持つことが知られている。

$REG = RSL$　(REG: 正則言語, RSL: 右側スティッカーの生成言語)

今回、オートマトンを模倣するスティッカーシステムを構成することで $REG \subseteq RSL$ を形式的に示した。

```
Theorem REG_RSL{state symbol:finType}(M:@automaton state symbol)(s:seq symbol):
s <> nil -> exists n:nat ,forall m:nat, n <= m ->
accept M s = (s\in(ss_language_prime m (Aut_to_Stk M))).
```

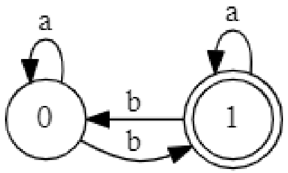
オートマトン

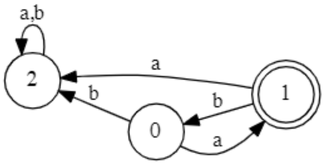
有限オートマトン $M = (K, V, s_0, F, \delta)$

K:状態の有限集合　　V: アルファベット

$\delta: K \times V \to K$: 遷移関数　　$s_0 \in K$:初期状態　$F \subseteq K$:受理状態

オートマトンの受理言語 $L(M) = \{V^* | \delta^*(s_0, V^*) \in F\}$





3.1　Coq/SSReflect/finType

有限の要素のみからなる型
eqType,countType等の特徴を引き継いでいる
enumにより要素の全列挙が可能となる
(例)enum bool_finType = [::true;false]

オートマトンやスティッカーシステムで扱う文字、文字列を
ascii,stringの代わりにfinTypeとそのリストとすることで、より
厳密な定義が可能となる.

オートマトンをCoq上で実装

有限オートマトン $\mathrm{M} = (K, V, s_0, F, \delta)$
K:状態の有限集合,　　　　V: アルファベット(文字の有限集合)
$\delta: K \times V \to K$: 遷移関数　$s_0 \in K$初期状態,　　$F \subseteq K$受理状態

```
Structure automaton{state symbol:finType}:= Automaton {
  init : state;
  final : {set state};
  delta : state -> symbol -> state
}.
```

オートマトンをCoq上で実装

$\delta:(K,V) \to K$: 遷移関数
$\delta^*: K \times V^* \to K$:遷移関数の推移的反射閉包
オートマトンの受理判定:$\delta^*(s_0, w) \in F \; or \notin F \qquad (w \in V^*)$

```
Fixpoint dstar{state symbol:finType}(delta:state->symbol->state)
  (q:state)(str:seq symbol):state :=
match str with
|nil => q
|h::str' => dstar delta (delta q h) str'
end.
Definition accept{state symbol:finType}(M:@automaton state symbol)
  (str:seq symbol):bool := dstar (delta M) (init M) str\in final M.
```

DNAドミノをCoq上で実装

中央の二本鎖部と末端の一本鎖部をそれぞれ定義し，その組み合わせで様々なドミノの形状を表現する

```
Inductive domino{symbol:finType}{rho:Rho symbol}:=
|null : domino
|Simplex : @stickyend symbol -> domino
|WK : @wk symbol rho -> domino
|L : @stickyend symbol -> @wk symbol rho -> domino
|R : @wk symbol rho -> @stickyend symbol -> domino
|LR : @stickyend symbol -> @wk symbol rho -> @stickyend symbol -> domino.
```

スティッカーシステムをCoq上で実装

スティッカーシステム：$\gamma = (V, \rho, S, D)$

V：アルファベット（記号の有限集合）　$\rho \subseteq V \times V$

S：二本鎖部を持つドミノの有限集合　　D：ドミノ対の有限集合

```
Structure sticker{symbol : finType}{rho : seq(symbol*symbol)} :={
  start : seq (@domino symbol rho);
  extend : seq ((@domino symbol rho)*(@domino symbol rho))
}.
```

$\boldsymbol{REG} \subseteq \boldsymbol{RSL}$の形式証明

約2800行からなる形式証明の全文
証明の正しさはCoqが保証するため
手動で確かめる必要はない

詳細については以下のページで公開しています。
https://github.com/KyushuUniversityMathematics/CoqSticker

```
Theorem REG_RSL{state symbol:finType}(M:@automaton state symbol)(s:seq symbol):
s <> nil -> exists n:nat ,forall m:nat, n <= m ->
accept M s = (s\in(ss_language_prime m (Aut_to_Stk M))).
```

まとめと今後の課題

今回、(片側)スティッカーシステムが正則文法以上の生成能力を持つことをCoqで形式的に検証した。

DNA計算モデルは他にも多数存在し、中には帰納的加算言語を生成可能なものも存在する。

- (両側)スティッカーシステム
- ワトソンクリックオートマトン
- 挿入・削除システム
- スプライシングシステム
- 環状スプライシングシステム

今後、これらのモデルに関する証明も形式化していきたい。

参考文献

[1] H.Tanaka,I.Sakashita,S.Inokuchi,Y.Mizoguchi,*Formal Proofs for Automata and Sticker Systems*,Proc. of First International Symposium on Computing and Networking, 2013.12.

[2] 萩原学,アフェルト・レナルド, *Coq/ SSreflect/ MathCompによる形式証明*, 森北出版, 2018.

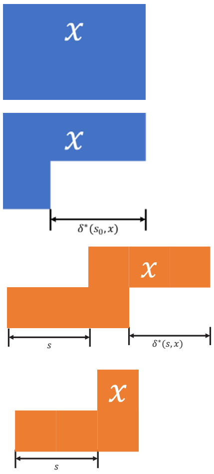
[3]G.パウン/G.ローゼンバーグ/A.サローマ,*DNAコンピューティング 新しい計算パラダイム,*Springer-Verlag Tokyo,1999.

[4] L. C. Meiser, et.al, *DNA applications in information technology*, Nature Communications , Vol.13, No.352, 2022.

付録:オートマトンの模倣

S (開始ドミノ)

D(結合させるドミノ)



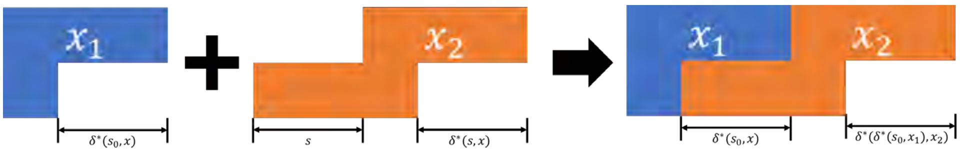
オートマトンがxを受理する
$\delta^*(s_0,x) \in F$

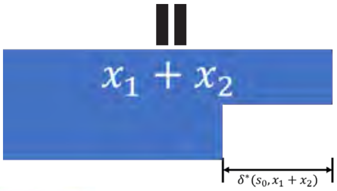
オートマトンにxを作用させた状態
$\delta^*(s_0,x)$を粘着末端で表現

オートマトンがある状態sからxにより
遷移した状態$\delta^*(s,x)$を粘着末端で表現

オートマトンがある状態sから
xを受理する　$\delta^*(s,x) \in F$

付録:オートマトンの模倣



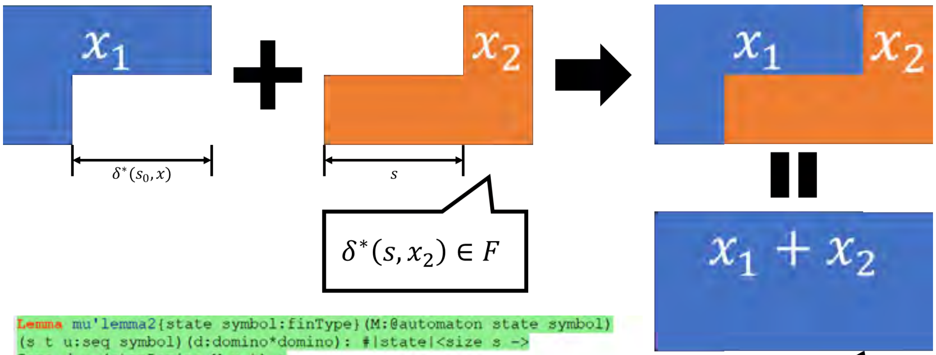


$$\delta^*(\delta^*(s_0,x_1),x_2) = \delta^*(s_0,x_1+x_2)$$

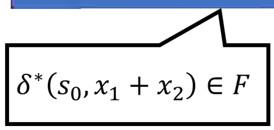
オートマトンの遷移は結合的

```
Lemma mu'lemma(state symbol:finType)(M:@automaton state symbol)
(s t u:seq symbol):#|state|.+1<=size s -> size t = #|state|.+1 -> u<>nil->
mu' (startDomino M s)(extentionDomino M t u)=
if drop(size s-((index(dstar(delta M)(init M)s)(enum state)).+1))s == u then
  Some (startDomino M (s++t))
else
  None.
```

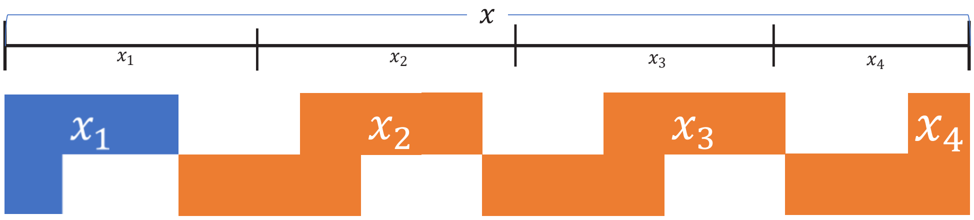
付録:オートマトンの模倣




```
Lemma mu'lemma2{state symbol:finType}(M:@automaton state symbol)
(s t u:seq symbol)(d:domino*domino): #|state|<size s ->
Some d = (stopDomino M u t)->
mu'(startDomino M s)d =
if (drop (size s-(index(dstar(delta M)(init M)s)(enum state)).+1)s)
  ==u then
  mkWK (s++t)
else
  None.
```



付録:オートマトンの模倣



$\delta^*(\delta^*(\delta^*(\delta^*(s_0,x_1),x_2),x_3),x_4) \in F \quad \Leftrightarrow \delta^*(s_0,x_1+x_2+x_3+x_4)$

$\Leftrightarrow\ \delta^*(s_0,x) \in F\ \Leftrightarrow\ x$はオートマトンにより受理される

任意の文字列は分割した有限長の文字列に対応するドミノの組み合わせで受理判定が可能

⇒オートマトンの受理判定をスティッカーで模倣できる

Developing Practical Lemmas for Real Analysis in Coq

(Coq による実解析のための実用的な補題の開発)

Yoshihiro Ishiguro [1]

Graduate School of Mathematics, Nagoya University,
Digital Architecture Research Center, AIST

November 26, 2024

[1] Joint work with **Reynald Affeldt**, Digital Architecture Research Center, AIST

The main topic: calculation of integrals in Coq

- integrals over bounded intervals
- integrals over unbounded intervals by limit of integration area (improper intergal, 広義積分)
- example: Gaussian integral

$$\int_0^{+\infty} \exp(-x^2)(\mathbf{d}\, x) = \frac{\sqrt{\pi}}{2}$$

Motivation and Approach

- Motivation:
  - probability theory
  - varification of the semantics of probabilistic programs
- Approach
  - use and extend MathComp-Analysis
    - The library of Coq mainly about analysis
  - add and generalize lemmas for real analysis
  - milestone: Gaussian integral
    - → focus on calculation of integrals
    - → formalize lemmas for continuous function
    - → generalize some lemmas if needed (bounded → unbounded)

remark: FTC for continuous function

Fundamental Theorem of Calculus(FTC, 微分積分学の基本定理)
contatins two part of statement
in this slide, these are written as:

- FTC1 For continuous function f and $a \in \mathbb{R}$,
  an antiderivative of f can be obtain as
  the function that x maps to the integral of f over $[a, x]$

$$\left(\int_a^x f(t)(\mathrm{d}\, t)\right)' = f(x).$$

- FTC2 For any continuous function f and $a, b \in \mathbb{R}$,
  the integral of f over $[a, b]$ is equal to
  the change of any antiderivative from a to b

$$\int_a^b f(x)(\mathrm{d}\, x) = F(b) - F(a)$$

background: about MathComp-Analysis

- MathComp-Analysis is a library of Coq for analysis.
  - derivation (derive.v), measure theory (measure.v), lebesgue_integral (lebesgue_integral.v)
  - Lemmas about integrals for real function

GitHub: https://github.com/math-comp/analysis

results before gauss integral

In MathComp-Analysis, there are already some important lemma for integration:

- lebesgue differentiation theorem
- FTC1

→ but not enough.
So we formalize theorems for derivation and integrals for continuous function, over bounded interval, such as:

- FTC2
- integration by parts(部分積分)
- integration by substitution(積分の変数変換)

→ make easier to deal with evaluation of integration
→ also to deal with semantics of probabilistic programs

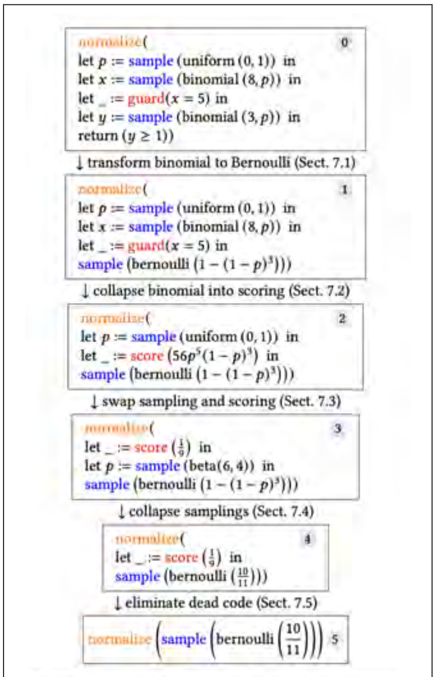


Figure 1. Proof of the table game by equational reasoning

Example of normalizing a probabilistic program in Coq

- each steps are evaluation by calculation of integrals
- distributions `uniform(-,-)`,`binomial(-,-)`, `beta(-,-)`
- using Radon-Nikodým theorem, integration by parts, etc.

highlight: FTC2 for bounded interval

For a real function $f(x), F(x)$ over $[a, b]$,
if f is derivation of F, then

$$\int_a^b f(x)(\mathrm{d}\, x) = F(b) - F(a)$$

- allow to calculate integrals via derivation
- for elementary function, the relation $F'(x) = f(x)$ is almost automatically evaluate with tactic `rewrite derive_val`, or applying small number of lemmas in MathComp-Analysis

FTC2 in Coq (bounded)

```
Theorem Rintegral_continuous_FTC2 {R: realType}
  (f F : R -> R) (a b : R) :
  (a < b)%R ->
  {within `[a, b], continuous f} ->
  derivable_oo_continuous_bnd F a b ->
  {in `]a, b[, F^`() =1 f} ->
  (\int[lebesgue_measure]_(x in `[a, b]) (f x)
        = (F b) - (F a))%R.
```

`derivable_oo_continuous_bnd F a b` is a notation for derivability in $]a, b[$, and right continuity at a and left continuity at b.

FTC2 in Coq (unbounded)

```
Lemma ge0_within_pinfty_continuous_FTC2
      {R : realType} (f F : R -> R) a (l : R) :
   (forall x, (a <= x)%R -> 0 <= f x)%R ->
   F x @[x --> +oo%R] --> l ->
   {within `[a, +oo[, continuous f} ->
   (forall x, (a < x)%R -> derivable F x 1) ->
   F x @[x --> a^'+] --> F a ->
   {in `]a, +oo[, F^`() =1 f} ->
   (\int[lebesgue_measure]_(x in `[a, +oo[) (f x)
        = l - (F a))%R.
```

$\forall x, 0 \leq f(x)$ is sufficient to exists integral over unbounded interval also needed convergence of limit to $+\infty$

highlight 2: differentiation under integral

a measurable set $A \subset \mathbb{R}$, and an open interval $I \subset \mathbb{R}$
For continuous functions $f(x, y) : I \times A \to \mathbb{R}$ satisfies following conditions:

- any $x \in I$, the function $y \mapsto f(x, y)$ is integrable on A
- $f(x, y)$ is derivable for any $x \in I, y \in A$
- there exists a integrable function $G(y)$ which satisfies
  For all $x \in I, y \in A, |f(x, y)| \leq G(y)$

then, $F : x \mapsto \int_A f(x, y)(\mathbf{d}\, y)$ is derivable for all x, and satisfies

$$\frac{dF}{dx} = \int_A \frac{\partial f}{\partial x}(x, y)(\mathbf{d}\, y)$$

outline of proof of gaussian integral

Consider the following functions

$$\begin{aligned} f(x) &= \int_0^x \exp(-y^2)(\mathbf{d}\, y) \\ u(x, y) &= \exp(-(y^2+1) * x^2)/(y^2+1) \\ g(x) &= \int_0^1 u(x, y)(\mathbf{d}\, y) \\ h(x) &= g(x) + (f(x))^2 \end{aligned}$$

By calculation, $h(0) = \pi/2$ and

$$h'(x) = g'(x) + (\{f(x)\}^2)' = (-\{f(x)\}^2)' + (\{2 * f'(x) * f(x)\}) = 0$$

Then, we can obtain

$$(\text{Gaussian integral}) = \lim_{x \to +\infty} f(x) = \sqrt{h(0)} = \sqrt{\lim_{x \to +\infty} h(x)} = \sqrt{\pi/2}$$

Future work: Leibniz rule

Generalization of differentiation under integrals: The case of the boundries of integration area depends on x

$$\left(\int_{a(x)}^{b(x)} f(x,y)(\mathbf{d}\,y)\right)' =$$

$$(f(x,b(x)) * b'(x)) - (f(x,a(x)) * b'(x)) + \int_{a(x)}^{b(x)} \frac{df}{dx}(x,y)(\mathbf{d}\,y)$$

This lemma also implies FTC1

Future work: FTC2 for absolute continuous function

more general version of FTC2 is work in progress:
For an absolute continuous function F over $[a,b]$,
there is a function $f(x)$ which is defined at almost every where and satisfies

$$\int_a^b f(x)(\mathbf{d}\,x) = F(b) - F(a)$$

- → this infers that such f is unique up to difference at almost everywhere
- → convenience of evaluation of derivation

Conclusion
We formalize useful theorems for integrals and differentiation for continuous functions such as

- FTC
- integration by substitution
- differentiation under integrals

and formalize Gaussian integral as an application

Future work
Formalize more general lemmas, such as

- Leibniz rule
- FTC2 for absolute continuous functions

Formalize other evaluation probability distribution function, such as

- Gamma distribution

TPPmark2024 : Permutation Sort

Jacques Garrigue

Graduate School of Mathematics, Nagoya University

TPP 2024, Kyushu University, November 26, 2024

The problem

We want to sort a list in place, using a minimal number of exchanges. We do not count the time needed to decide the sequence of exchanges (but it should probably be tractable.)

1. Define a function which, given a permutation, returns the a minimal sequence of exchanges which realizes it.
2. Prove that this function is correct, and that the number of exchanges is minimal.
3. Define a function which, given a list of natural numbers without repetition, returns a minimal sequence of exchanges which sorts it. Prove it correct and minimal.
4. Define a function which, given a list of natural numbers possibly with repetitions, returns a minimal sequence of exchanges which sorts it. Prove it correct and minimal.

Permutations

You have probably learnt in you linear algebra course that, considering a finite set of n elements, there exists a group S_n, the *symmetric group*, containing all the permutations.
In particular

1. any permutation p has a unique decomposition into *cycles* (permutations defined by a list of elements $(a_1\ a_2 \ldots a_m)$ such $p(a_i) = a_{i+1}$ and $p(a_m) = a_1$). All these cycles have disjoint supports, so they commute.
2. A cycle of length m can be realized by the composition of $m-1$ *exchanges* (permutations that only exchange two elements).

$$(a_1\ a_2 \ldots a_m) = (a_1\ a_m) \cdot \cdots \cdot (a_1\ a_3) \cdot (a_1\ a_2)$$

   Any a_i is first exchanged with a_1, but in the step it is exchanged back to a_{i+1}.

Expected solution

1. Define a function doing the above decomposition.
2. Prove that it is optimal.
3. Use it to provide optimal sorting without duplicates.
4. Problem 4 is actually NP-hard: (citation from Yamamoto-sensei)

   Amihood Amir, Tzvika Hartman, Oren Kapah, Avivit Levy, and Ely Porat.
   On the cost of interchange rearrangement in strings.
   SIAM Journal on Computing, 39(4):1444-1461, 2010.

Alternative solution

Without hours of publishing the problem, Yamamoto-sensei sent me his solution.

1. Any permutation has a decompostion into exchanges (the proof is available in MathComp/fingroup), so just use `argmin` to return the shortest sequence.
2. Problem 3 and 4 can use the same solution: by definition a sorting function can only permute its input, so just return the shortest sequence.

Of course, while this is in theory computable, you will probably have to wait a while for the answer. . . .

The solutions

- Mitsuharu Yamamoto (both `argmin` and direct decomposition, MathComp/fingroup)
- Takefumi Saikawa (`argmin` approach, MathComp/fingroup)
- Kenta Inoue (standard approach?, MathComp/fingroup)
- Myself (standard approach, MathComp/ssreflect)

Main lemma

The idea of the proof of minimality is that

- the identity permutation has n cycles
  (if we see x such that $p(x) = x$ as a cycle of length 1)
- when composing a permutation p with an exchange t, the number of cycles may not decrease by more than 1
- as a result one cannot do better than $n - m$ exchanges (which is the number of exchanges in the standard decomposition)

We can indeed prove the following lemma.

Lemma

If p has m cycles, then $p \cdot t$ has either $m - 1$ or $m + 1$ cycles.

Actually, this lemma is available in MathComp/fingroup/perm.v,
but nobody seems to have used it :)

MI レクチャーノートシリーズ刊行にあたり

本レクチャーノートシリーズは、文部科学省 21 世紀 COE プログラム「機能数理学の構築と展開」（H15-19 年度）において作成した COE Lecture Notes の続刊であり、文部科学省大学院教育改革支援プログラム「産業界が求める数学博士と新修士養成」（H19-21 年度）および、同グローバル COE プログラム「マス・フォア・インダストリ教育研究拠点」（H20-24 年度）において行われた講義の講義録として出版されてきた。平成 23 年 4 月のマス・フォア・インダストリ研究所（IMI）設立と平成 25 年 4 月の IMI の文部科学省共同利用・共同研究拠点として「産業数学の先進的･基礎的共同研究拠点」の認定を受け、今後、レクチャーノートは、マス・フォア・インダストリに関わる国内外の研究者による講義の講義録、会議録等として出版し、マス・フォア・インダストリの本格的な展開に資するものとする。

2022 年 10 月
マス・フォア・インダストリ研究所
所長　梶原 健司

2024年度採択分　九州大学マス･フォア･インダストリ研究所 共同利用研究集会

コンピュータによる定理証明支援とその応用

発　行　2025年 3月17 日
編　集　Jacques Garrigue
発　行　九州大学マス・フォア・インダストリ研究所
九州大学大学院数理学府
〒819-0395 福岡市西区元岡744
九州大学数理・IMI 事務室
TEL 092-802-4402　FAX 092-802-4405
URL https://www.imi.kyushu-u.ac.jp/

印　刷　城島印刷株式会社
〒810-0012 福岡市中央区白金 2 丁目 9 番 6 号
TEL 092-531-7102　FAX 092-524-4411

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note	Mitsuhiro T. NAKAO Kazuhiro YOKOYAMA	Computer Assisted Proofs - Numeric and Symbolic Approaches - 199pages	August 22, 2006
COE Lecture Note	M.J.Shai HARAN	Arithmetical Investigations - Representation theory, Orthogonal polynomials and Quantum interpolations- 174pages	August 22, 2006
COE Lecture Note Vol.3	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2005 155pages	October 13, 2006
COE Lecture Note Vol.4	宮田　健治	辺要素有限要素法による磁界解析 - 機能数理学特別講義 21pages	May 15, 2007
COE Lecture Note Vol.5	Francois APERY	Univariate Elimination Subresultants - Bezout formula, Laurent series and vanishing conditions - 89pages	September 25, 2007
COE Lecture Note Vol.6	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2006 209pages	October 12, 2007
COE Lecture Note Vol.7	若山　正人 中尾　充宏	九州大学産業技術数理研究センター キックオフミーティング 138pages	October 15, 2007
COE Lecture Note Vol.8	Alberto PARMEGGIANI	Introduction to the Spectral Theory of Non-Commutative Harmonic Oscillators 233pages	January 31, 2008
COE Lecture Note Vol.9	Michael I.TRIBELSKY	Introduction to Mathematical modeling 23pages	February 15, 2008
COE Lecture Note Vol.10	Jacques FARAUT	Infinite Dimensional Spherical Analysis 74pages	March 14, 2008
COE Lecture Note Vol.11	Gerrit van DIJK	Gelfand Pairs And Beyond 60pages	August 25, 2008
COE Lecture Note Vol.12	Faculty of Mathematics, Kyushu University	Consortium “MATH for INDUSTRY” First Forum 87pages	September 16, 2008
COE Lecture Note Vol.13	九州大学大学院 数理学研究院	プロシーディング「損保数理に現れる確率モデル」 — 日新火災・九州大学 共同研究2008年11月 研究会 — 82pages	February 6, 2009

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.14	Michal Beneš, Tohru Tsujikawa Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2008 77pages	February 12, 2009
COE Lecture Note Vol.15	Faculty of Mathematics, Kyushu University	International Workshop on Verified Computations and Related Topics 129pages	February 23, 2009
COE Lecture Note Vol.16	Alexander Samokhin	Volume Integral Equation Method in Problems of Mathematical Physics 50pages	February 24, 2009
COE Lecture Note Vol.17	矢嶋　　徹 及川　正行 梶原　健司 辻　　英一 福本　康秀	非線形波動の数理と物理　66pages	February 27, 2009
COE Lecture Note Vol.18	Tim Hoffmann	Discrete Differential Geometry of Curves and Surfaces　75pages	April 21, 2009
COE Lecture Note Vol.19	Ichiro Suzuki	The Pattern Formation Problem for Autonomous Mobile Robots ―Special Lecture in Functional Mathematics―　23pages	April 30, 2009
COE Lecture Note Vol.20	Yasuhide Fukumoto Yasunori Maekawa	Math-for-Industry Tutorial: Spectral theories of non-Hermitian operators and their application　184pages	June 19, 2009
COE Lecture Note Vol.21	Faculty of Mathematics, Kyushu University	Forum "Math-for-Industry" Casimir Force, Casimir Operators and the Riemann Hypothesis 95pages	November 9, 2009
COE Lecture Note Vol.22	Masakazu Suzuki Hoon Hong Hirokazu Anai Chee Yap Yousuke Sato Hiroshi Yoshida	The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium on Computer Mathematics Mathematical Aspects of Computer and Information Sciences　436pages	December 14, 2009
COE Lecture Note Vol.23	荒川　恒男 金子　昌信	多重ゼータ値入門　111pages	February 15, 2010
COE Lecture Note Vol.24	Fulton B.Gonzalez	Notes on Integral Geometry and Harmonic Analysis　125pages	March 12, 2010
COE Lecture Note Vol.25	Wayne Rossman	Discrete Constant Mean Curvature Surfaces via Conserved Quantities 130pages	May 31, 2010
COE Lecture Note Vol.26	Mihai Ciucu	Perfect Matchings and Applications　66pages	July 2, 2010

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.27	九州大学大学院 数理学研究院	Forum “Math-for-Industry” and Study Group Workshop Information security, visualization, and inverse problems, on the basis of optimization techniques 100pages	October 21, 2010
COE Lecture Note Vol.28	ANDREAS LANGER	MODULAR FORMS, ELLIPTIC AND MODULAR CURVES LECTURES AT KYUSHU UNIVERSITY 2010 62pages	November 26, 2010
COE Lecture Note Vol.29	木田 雅成 原田 昌晃 横山 俊一	Magma で広がる数学の世界 157pages	December 27, 2010
COE Lecture Note Vol.30	原 隆 松井 卓 廣島 文生	Mathematical Quantum Field Theory and Renormalization Theory 201pages	January 31, 2011
COE Lecture Note Vol.31	若山 正人 福本 康秀 高木 剛 山本 昌宏	Study Group Workshop 2010 Lecture & Report 128pages	February 8, 2011
COE Lecture Note Vol.32	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2011 “TSUNAMI-Mathematical Modelling” Using Mathematics for Natural Disaster Prediction, Recovery and Provision for the Future 90pages	September 30, 2011
COE Lecture Note Vol.33	若山 正人 福本 康秀 高木 剛 山本 昌宏	Study Group Workshop 2011 Lecture & Report 140pages	October 27, 2011
COE Lecture Note Vol.34	Adrian Muntean Vladimír Chalupecký	Homogenization Method and Multiscale Modeling 72pages	October 28, 2011
COE Lecture Note Vol.35	横山 俊一 夫 紀恵 林 卓也	計算機代数システムの進展 210pages	November 30, 2011
COE Lecture Note Vol.36	Michal Beneš Masato Kimura Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2010 107pages	January 27, 2012
COE Lecture Note Vol.37	若山 正人 高木 剛 Kirill Morozov 平岡 裕章 木村 正人 白井 朋之 西井 龍映 栄 伸一郎 穴井 宏和 福本 康秀	平成23年度 数学・数理科学と諸科学・産業との連携研究ワークショップ 拡がっていく数学 ～期待される“見えない力”～ 154pages	February 20, 2012

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.38	Fumio Hiroshima Itaru Sasaki Herbert Spohn Akito Suzuki	Enhanced Binding in Quantum Field Theory 204pages	March 12, 2012
COE Lecture Note Vol.39	Institute of Mathematics for Industry, Kyushu University	Multiscale Mathematics; Hierarchy of collective phenomena and interrelations between hierarchical structures 180pages	March 13, 2012
COE Lecture Note Vol.40	井ノ口順一 太田　泰広 筧　三郎 梶原　健司 松浦　望	離散可積分系・離散微分幾何チュートリアル2012 152pages	March 15, 2012
COE Lecture Note Vol.41	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2012 “Information Recovery and Discovery” 91pages	October 22, 2012
COE Lecture Note Vol.42	佐伯　修 若山　正人 山本　昌宏	Study Group Workshop 2012 Abstract, Lecture＆Report 178pages	November 19, 2012
COE Lecture Note Vol.43	Institute of Mathematics for Industry, Kyushu University	Combinatorics and Numerical Analysis Joint Workshop 103pages	December 27, 2012
COE Lecture Note Vol.44	萩原　学	モダン符号理論からポストモダン符号理論への展望 107pages	January 30, 2013
COE Lecture Note Vol.45	金山　寛	Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University “Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)” 121pages	February 19, 2013
COE Lecture Note Vol.46	西井　龍映 栄　伸一郎 岡田　勘三 落合　啓之 小磯　深幸 斎藤　新悟 白井　朋之	科学・技術の研究課題への数学アプローチ ―数学モデリングの基礎と展開― 325pages	February 28, 2013
COE Lecture Note Vol.47	SOO TECK LEE	BRANCHING RULES AND BRANCHING ALGEBRAS FOR THE COMPLEX CLASSICAL GROUPS 40pages	March 8, 2013
COE Lecture Note Vol.48	溝口　佳寛 脇　隼人 平坂　貢 谷口　哲至 島袋　修	博多ワークショップ「組み合わせとその応用」 124pages	March 28, 2013

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.49	照井　　章 小原　功任 濱田　龍義 横山　俊一 穴井　宏和 横田　博史	マス・フォア・インダストリ研究所　共同利用研究集会 II 数式処理研究と産学連携の新たな発展　137pages	August 9, 2013
MI Lecture Note Vol.50	Ken Anjyo Hiroyuki Ochiai Yoshinori Dobashi Yoshihiro Mizoguchi Shizuo Kaji	Symposium MEIS2013: Mathematical Progress in Expressive Image Synthesis　154pages	October 21, 2013
MI Lecture Note Vol.51	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2013 “The Impact of Applications on Mathematics”　97pages	October 30, 2013
MI Lecture Note Vol.52	佐伯　　修 岡田　勘三 髙木　　剛 若山　正人 山本　昌宏	Study Group Workshop 2013 Abstract, Lecture ＆ Report　142pages	November 15, 2013
MI Lecture Note Vol.53	四方　義啓 櫻井　幸一 安田　貴徳 Xavier Dahan	平成25年度　九州大学マス・フォア・インダストリ研究所 共同利用研究集会　安全・安心社会基盤構築のための代数構造 ～サイバー社会の信頼性確保のための数理学～　158pages	December 26, 2013
MI Lecture Note Vol.54	Takashi Takiguchi Hiroshi Fujiwara	Inverse problems for practice, the present and the future　93pages	January 30, 2014
MI Lecture Note Vol.55	栄　伸一郎 溝口　佳寛 脇　　隼人 渋田　敬史	Study Group Workshop 2013 数学協働プログラム Lecture & Report 98pages	February 10, 2014
MI Lecture Note Vol.56	Yoshihiro Mizoguchi Hayato Waki Takafumi Shibuta Tetsuji Taniguchi Osamu Shimabukuro Makoto Tagami Hirotake Kurihara Shuya Chiba	Hakata Workshop 2014 ~ Discrete Mathematics and its Applications ~　141pages	March 28, 2014
MI Lecture Note Vol.57	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2014: “Applications + Practical Conceptualization + Mathematics = fruitful Innovation”　93pages	October 23, 2014
MI Lecture Note Vol.58	安生健一 落合啓之	Symposium MEIS2014: Mathematical Progress in Expressive Image Synthesis　135pages	November 12, 2014

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.59	西井　龍映 岡田　勘三 梶原　健司 髙木　　剛 若山　正人 脇　　隼人 山本　昌宏	Study Group Workshop 2014 数学協働プログラム Abstract, Lecture & Report　196pages	November 14, 2014
MI Lecture Note Vol.60	西浦　　博	平成26年度九州大学 IMI 共同利用研究・研究集会（I） 感染症数理モデルの実用化と産業及び政策での活用のための新たな展開　120pages	November 28, 2014
MI Lecture Note Vol.61	溝口　佳寛 Jacques Garrigue 萩原　　学 Reynald Affeldt	研究集会 高信頼な理論と実装のための定理証明および定理証明器 Theorem proving and provers for reliable theory and implementations (TPP2014)　138pages	February 26, 2015
MI Lecture Note Vol.62	白井　朋之	Workshop on “β-transformation and related topics”　59pages	March 10, 2015
MI Lecture Note Vol.63	白井　朋之	Workshop on “Probabilistic models with determinantal structure” 107pages	August 20, 2015
MI Lecture Note Vol.64	落合　啓之 土橋　宜典	Symposium MEIS2015: Mathematical Progress in Expressive Image Synthesis　124pages	September 18, 2015
MI Lecture Note Vol.65	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2015 “The Role and Importance of Mathematics in Innovation”　74pages	October 23, 2015
MI Lecture Note Vol.66	岡田　勘三 藤澤　克己 白井　朋之 若山　正人 脇　　隼人 Philip Broadbridge 山本　昌宏	Study Group Workshop 2015 Abstract, Lecture & Report 156pages	November 5, 2015
MI Lecture Note Vol.67	Institute of Mathematics for Industry, Kyushu University	IMI-La Trobe Joint Conference “Mathematics for Materials Science and Processing” 66pages	February 5, 2016
MI Lecture Note Vol.68	古庄　英和 小谷　久寿 新甫　洋史	結び目と Grothendieck-Teichmüller 群 116pages	February 22, 2016
MI Lecture Note Vol.69	土橋　宜典 鍛治　静雄	Symposium MEIS2016: Mathematical Progress in Expressive Image Synthesis　82pages	October 24, 2016
MI Lecture Note Vol.70	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2016 “Agriculture as a metaphor for creativity in all human endeavors” 98pages	November 2, 2016
MI Lecture Note Vol.71	小磯　深幸 二宮　嘉行 山本　昌宏	Study Group Workshop 2016 Abstract, Lecture & Report　143pages	November 21, 2016

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.72	新井　朝雄 小嶋　　泉 廣島　文生	Mathematical quantum field theory and related topics　133pages	January 27, 2017
MI Lecture Note Vol.73	穴田　啓晃 Kirill Morozov 須賀　祐治 奥村　伸也 櫻井　幸一	Secret Sharing for Dependability, Usability and Security of Network Storage and Its Mathematical Modeling　211pages	March 15, 2017
MI Lecture Note Vol.74	QUISPEL, G. Reinout W. BADER, Philipp MCLAREN, David I. TAGAMI, Daisuke	IMI-La Trobe Joint Conference Geometric Numerical Integration and its Applications　71pages	March 31, 2017
MI Lecture Note Vol.75	手塚　　集 田上　大助 山本　昌宏	Study Group Workshop 2017 Abstract, Lecture & Report　118pages	October 20, 2017
MI Lecture Note Vol.76	宇田川誠一	Tzitzéica 方程式の有限間隙解に付随した極小曲面の構成理論 ―Tzitzéica 方程式の楕円関数解を出発点として―　68pages	August 4, 2017
MI Lecture Note Vol.77	松谷　茂樹 佐伯　　修 中川　淳一 田上　大助 上坂　正晃 Pierluigi Cesana 濵田　裕康	平成29年度　九州大学マス・フォア・インダストリ研究所 共同利用研究集会（I） 結晶の界面，転位，構造の数理　148pages	December 20, 2017
MI Lecture Note Vol.78	瀧澤　重志 小林　和博 佐藤憲一郎 斎藤　　努 清水　正明 間瀬　正啓 藤澤　克樹 神山　直之	平成29年度　九州大学マス・フォア・インダストリ研究所 プロジェクト研究　研究集会（I） 防災・避難計画の数理モデルの高度化と社会実装へ向けて 136pages	February 26, 2018
MI Lecture Note Vol.79	神山　直之 畔上　秀幸	平成29年度　AIMaP チュートリアル 最適化理論の基礎と応用　96pages	February 28, 2018
MI Lecture Note Vol.80	Kirill Morozov Hiroaki Anada Yuji Suga	IMI Workshop of the Joint Research Projects Cryptographic Technologies for Securing Network Storage and Their Mathematical Modeling　116pages	March 30, 2018
MI Lecture Note Vol.81	Tsuyoshi Takagi Masato Wakayama Keisuke Tanaka Noboru Kunihiro Kazufumi Kimoto Yasuhiko Ikematsu	IMI Workshop of the Joint Research Projects International Symposium on Mathematics, Quantum Theory, and Cryptography　246pages	September 25, 2019
MI Lecture Note Vol.82	池森　俊文	令和２年度　AIMaP チュートリアル 新型コロナウイルス感染症にかかわる諸問題の数理 145pages	March 22, 2021

シリーズ既刊

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.89	松谷　茂樹 落合　啓之 井上　和俊 小磯　深幸 佐伯　　修 白井　朋之 垂水　竜一 内藤　久資 中川　淳一 濵田　裕康 松江　　要 加葉田雄太朗	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 材料科学における幾何と代数 III　356pages	December 7, 2022
MI Lecture Note Vol.90	中山　尚子 谷川　拓司 品野　勇治 近藤　正章 石原　　亨 鍛治　静雄 藤澤　克樹	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 データ格付けサービス実現のための数理基盤の構築　58pages	December 12, 2022
MI Lecture Note Vol.91	Katsuki Fujisawa Shizuo Kaji Toru Ishihara Masaaki Kondo Yuji Shinano Takuji Tanigawa Naoko Nakayama	IMI Workshop of the Joint Usage Research Projects Construction of Mathematical Basis for Realizing Data Rating Service 610pages	December 27, 2022
MI Lecture Note Vol.92	丹田　　聡 三宮　　俊 廣島　文生	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 時間・量子測定・準古典近似の理論と実験 ～古典論と量子論の境界～　150pages	Janualy 6, 2023
MI Lecture Note Vol.93	Philip Broadbridge Luke Bennetts Melanie Roberts Kenji Kajiwara	Proceedings of Forum “Math-for-Industry” 2022 -Mathematics of Public Health and Sustainability-　170pages	June 19, 2023
MI Lecture Note Vol.94	國廣　　昇 池松　泰彦 伊豆　哲也 穴田　啓晃 縫田　光司	2023年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 現代暗号に対する安全性解析・攻撃の数理　260pages	Janualy 11, 2024
MI Lecture Note Vol.96	澤田　茉伊	2023年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 デジタル化時代に求められる斜面防災の思考法　70pages	March 18, 2024

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.97	Shariffah Suhaila Syed Jamaludin Zaiton Mat Isa Nur Arina Bazilah Aziz Taufiq Khairi Ahmad Khairuddin Shaymaa M.H.Darwish Ahmad Razin Zainal Abidin Norhaiza Ahmad Zainal Abdul Aziz Hang See Pheng Mohd Ali Khameini Ahmad	International Project Research-Workshop (I) Proceedings of 4th Malaysia Mathematics in Industry Study Group (MMISG2023) 172pages	March 28, 2024
MI Lecture Note Vol.98	中澤　嵩	2024年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 自動車性能の飛躍的向上を目指す Data-Driven 設計　82pages	January 30, 2025

Institute of Mathematics for Industry
Kyushu University

九州大学マス・フォア・インダストリ研究所
九州大学大学院 数理学府
〒819-0395 福岡市西区元岡744　TEL 092-802-4402　FAX 092-802-4405
URL https://www.imi.kyushu-u.ac.jp/