Math-for-industry
Education & Research Hub

ISSN2188-1200

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

編集：溝口佳寛, Jacques Garrigue, 萩原学, Reynald Affeldt

九州大学マス・フォア・インダストリ研究所

MI Lecture Note Vol.61 : Kyushu University

# 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)

編集: 溝口佳寛, Jacques Garrigue, 萩原学, Reynald Affeldt

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 2014
Yasuhide Fukumoto
Director
Institute of Mathematics for Industry

# 序 文

高信頼なソフトウェア開発のために必要な形式手法・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, 22 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, プログラム検証について Adam Chlipala 氏 (米 MIT), 数学理論の形式証明ライブラリ開発について Cyril Cohen 氏 (仏 INRIA), 産業界からは, おサイフケータイの形式仕様策定について栗田太郎氏 (フェリカネットワークス), デジタル放送のプログラム検証について今井宣洋氏 (IT プランニング) , そして, 幾何学と幾何証明の数学ソフトウェアについて阿原一志氏 (明治大学) に特別講演を行って頂き, それぞれの分野における現状の把握, 問題点の明確化, そして, 定理証明器を用いた高信頼なソフトウェア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, TPPmark2014 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. 問題としては, 大学入試問題レベルの数学の論証問題を提示しました. 国内外から 7 種類の定理証明器による 15 件の応募があり, 有意義な意見交換が出来ました.

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 10 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).

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.

2015 年 3 月


| | |
|---|---|
| 溝口 佳寛 | (九州大学) |
| Jacques Garrigue | (名古屋大学) |
| 萩原 学 | (千葉大学) |
| Reynald Affeldt | (産業技術総合研究所) |


[1] http://imi.kyushu-u.ac.jp/lasm/tpp2014/index_ja.html
[2] https://github.com/KyushuUniversityMathematics/TPP2014/wiki

# Preface

This workshop, held at the Nishijin Plaza of Kyushu University on December 3–5, 2014, was intended as a forum for discussions about formal methods, 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 22 talks and lectures.

Five special lectures were given. Adam Chlipala (MIT, USA) talked about program verification, and Cyril Cohen (INRIA, France) about a mathematical library for formalized mathematics. From the industry, Taro Kurita (Sony Corp., Japan) talked about development of firmware using formal methods, and Yoshihiro Imai (IT Plannig Inc., Japan) about program verification for digital terrestrial broadcasting. Finally Prof. Kazushi Ahara (Meiji Univ., Japan) gave a lecture about mathematical software for elementary geometry and its proof system. 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.

TPPmark2014 allowed us to compare proofs of the same theorem using several different proof assistant systems, emphasizing the different features and approaches. The problem was one of elementary mathematics, which has been used for the entrance examination fof Kyushu University. 15 proofs were submitted using 7 kinds of proof assistant systems. The discussions were much enjoyed.

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 year. This was the tenth 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).

You can find link to those from the TPP2014 homepage [3]. The problem and answers of the TPPmark2014 are available from a repository [4].

March, 2015.


| | |
|---|---|
| Yoshihiro Mizoguchi | (Kyushu University) |
| Jacques Garrigue | (Nagoya University) |
| Manabu Hagiwara | (Chiba University) |
| Reynald Affeldt | (AIST) |


[3] http://imi.kyushu-u.ac.jp/lasm/tpp2014/
[4] https://github.com/KyushuUniversityMathematics/TPP2014/wiki

# 目　次

# Table of Contents

Theorem proving and provers for reliable theory and implementations

## Date:

December 3, 2014 – December 5, 2014

## Venue:

Nishijin Plaza, Kyushu University
(2-16-23 Nishijin, Sawara-ku, Fukuoka City, JAPAN)

## Program

**Dec. 3**

| | |
|---|---|
| 13:00 - | Registration |
| 13:30 - 14:00 | Shogo Sekine (Chiba Univ.) |
| | Formalizing Karp and Miller Acceralation for Petri Nets using SSReflect |
| 14:00 - 14:30 | Sosuke Moriguchi (Kwansei Gakuin Univ.) |
| | Towards verification of program generations |
| 14:30 - 15:00 | Ryosuke Obi (Chiba Univ.) |
| | Formalization of Variable-Length Source Coding Theorem: Converse Part |
| 15:00 - 15:30 | Kazuhiko Sakaguchi (University of Tsukuba) |
| | Formalizing Strong Normalization Proofs |
| 15:45 - 16:15 | Masahiko Sato (Kyoto Univ.) |
| | A name-free lambda calculus |
| 16:15 - 16:45 | Kazuhisa Nakasho (Shinshu Univ.) |
| | Formalization of Direct Sum Decomposition of Groups in Mizar |
| 16:45 - 17:15 | Hiroyuki Okazaki (Shinshu Univ.) |
| | Formalization of polynomially bounded functions in Mizar |

**Dec. 4**

| | |
|---|---|
| 09:30 - 10:30 | Yoshihiro Imai (IT Planning, Inc.) |
| | Formalization and verification of stream data processing for datacasting in Coq |
| 10:40 - 11:40 | Taro Kurita (FeliCa Networks, Inc.) |
| | The Application of VDM to the Industrial Development of Firmware |
| | for a Smart Card IC Chip |
| 13:00 - 14:00 | Adam Chlipala (MIT) |
| | Correct-by-Construction Program Synthesis in Coq |
| 14:15 - 14:45 | Reynald Affeldt(AIST) Jacques Garrigue(Nagoya Univ.) |
| | Formalization of Error-correcting Codes using SSReflect |
| 14:45 - 15:15 | Fadoua Ghourabi (Kwansei Gakuin Univ.) |
| | Formalization of matrix representation of direction relations |
| | with application to the superposition of rectangles |
| 15:15 - 15:45 | Yoshinori Tanabe (NII) |
| | Coq code extraction to Scala and its correctness |
| 16:00 - 16:30 | Kentaro Okumura (Kyoto Univ.) |
| | Formalization of Featherweight Java on Coq by using weak HOAS |
| 16:30 - 17:00 | Masahiro Yasugi (Kyushu Institute of Technology) |
| | Towards Design of Universal Typed Intermediate Languages |
| | in Consideration of Memory Models |
| 17:00 - 17:30 | Masahiro Sato (Nagoya Univ.) |
| | An intuitionistic Set-theoretical Model of the Extend Calculus of Construction |
| | 15-minute Break |
| 17:45 - 18:30 | **TPPmark2014** |

**Dec. 5**

| | |
|---|---|
| 09:30 - 10:30 | Shunsuke Yatabe (JR West) |
| | Preparing formal specifications: uses of semiformal methods in the concept phase of a system design |
| 10:40 - 11:10 | Kenichi Kuga(Chiba Univ.) Manabu Hagiwara(Chiba Univ.) |
| | On formalization of basic geometric topology |
| 11:10 - 11:40 | Takafumi Saikawa (Nagoya Univ.) |
| | Formalizing a coding theory |
| 11:40 - 12:10 | Tetsuo Ida (University of Tsukuba) |
| | Formalization of geometric algebra with application to computational origami |
| 13:30 - 14:30 | Cyril Cohen (INRIA) |
| | Mathematical Components and Algebraic Numbers |
| 14:45 - 15:45 | Kazushi Ahara (Meiji Univ.) |
| | Interactive geometry software and automated theorem proving - Cinderella and KidsCindy |
| 16:00 | Closing |

## Committee

Yoshihiro Mizoguchi (Kyushu University, Japan)
Garrigue Jacques (Nagoya University, Japan)
Manabu Hagiwara (Chiba University, Japan)
Reynald Affeldt (AIST, Japan)

## Oranizer

Laboratory of Advanced Software in Mathematics, Institute of Mathematics for Industry, Kyushu University.

## Co-organizer

Coop with Math Program, The Institute of Statistical Mathematics

## Supported by

IEICE Kyushu Section
This workshop is supported by in part by JSPS KAKENHI(Grant-in-Aid for Exploratory Research)
Grant Number 25610034 and 25289118.

# SSReflectを用いたペトリネットにおける Karp-Miller加速の形式化

関根　祥吾

joint work with
山本 光晴（千葉大学）

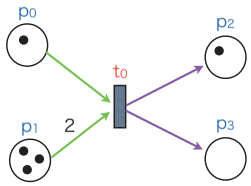
千葉大学理学研究科

December, 3, 2014

## Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 1, 3, 1, 0\rangle}$

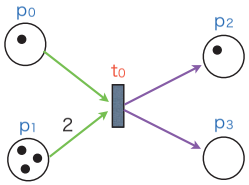



○ : プレース
▮ : トランジション
⇒ : アーク
• : トークン

## Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 1, 3, 1, 0\rangle}$



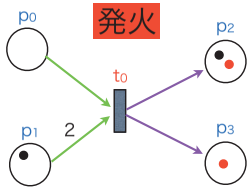

$1 \leqq$ ($p_0$のトークン数)
&
$2 \leqq$ ($p_1$のトークン数)
↓ なら
発火可能

## Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 0, 1, 2, 1\rangle}$

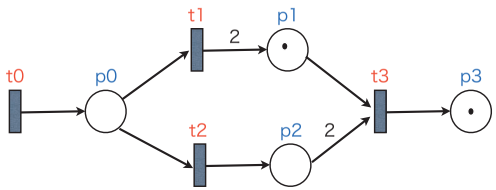
発火




$p_0$から1個
$p_1$から2個　トークンを除去

$p_2$から1個
$p_3$から1個　トークンを付加

## Petri Nets




<0,0,0,0> $\xrightarrow{t0;\ t1;\ t0;\ t2;\ t3}$ <0,1,0,1>

到達可能

## Petri Nets

ペトリネットの解析

可達問題

Mi <0,0,0,0> $\xrightarrow{\exists s}$ Md <0,1,0,1>

被覆問題

Mi <0,0,0,0> $\xrightarrow{\exists s}$ <0,2,0,2>

被覆

<0,1,0,1> Md

## 研究背景

[Karp & Miller, 1969]

Karp-Miller Tree → Minimal Coverability Set

KMTを用いてMCSを求める

計算量が多い

被覆性判定に使える

MCTを構成して効率化 [Finkel, 1991]

アルゴリズムの正当性に誤りを発見し、これを改善 [Luttge, 1995]

改善されたアルゴリズムの停止性に誤りを発見 [Geeraerts, 2007]

最適化で間違いが起き易い

## 研究背景

Karp-Miller Tree → Minimal Coverability Set

KMTを用いてMCSを求める

KMTのベースとなる部分の形式化

最適化の正当性、停止性の保証

（に貢献できるかも）

今回

KMTの構成におけるKM加速の一部を形式化

## Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

## ペトリネットの形式化

SSReflect：証明支援系Coqの拡張

有限集合についての性質に強い

有限集合の型が既に用意されている

Def:

`P:finType` プレース

`T:finType` トランジション

## ペトリネットの形式化

| | 関数 | tuple |
|---|---|---|
| 型の定義 | `M := P → nat` | `M := #\|P\|.-tuple nat` |
| プレースpにおけるトークン数 | `m p` | `tnth m (enum_rank p)` |
| マーキングのequalityと"="との一致 | 外延性が使えないので一致しない | グラフなので一致する |

`m:M p:P`

## ペトリネットの形式化

| | 有限集合上の関数 |
|---|---|
| 型の定義 | `M := {ffun P → nat}` |
| プレースpにおけるトークン数 | Coercionによって `m p` |
| マーキングのequalityと"="との一致 | 実態がグラフなので一致 |

`m:M p:P`

## 状態遷移の形式化

マーキングの　比較　計算　がしたい

↕　↕

ffun上の　関係　演算　として定義

## 状態遷移の形式化

各要素毎の比較

関係のcomponent-wiseへの拡張

cwR = component-wise relation

```
cwR (r:A → B → bool)
    (ma:{ffun P → A}) (mb:{ffun P → B}):=
        [forall (p:P), r (ma p) (mb p)]
```

bool値

有限だから

Notation : `m <=m m' := cwR leq m m'`

## 状態遷移の形式化

component-wiseな演算

```
ffmap2 (f:A → B → C)
    (ma:{ffun P → A}) (mb:{ffun P → B}) :=
            [ffun (p:P)=> f (ma p) (mb p)]
```

マーキングの加法　　自然数上の加法

↑　　↑

Notation: `m +m m' := ffmap2 addn m m'`

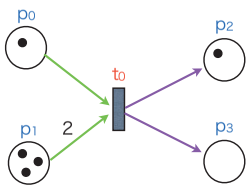
`m -m m' := ffmap2 subn m m'`

↓　　↓

マーキングの減法　　自然数上の減法

## 状態遷移の形式化

状態遷移




$m_0$ = <1,3,1,0>

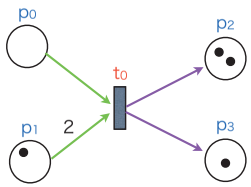
Pre $t_0$ = <1,2,0,0>

Post $t_0$ = <0,0,1,1>

発火条件　　Pre $t_0$ <1,2,0,0> <=m $m_0$ <1,3,1,0>

## 状態遷移の形式化

状態遷移




$m_0$ = <1,3,1,0>

Pre $t_0$ = <1,2,0,0>

Post $t_0$ = <0,0,1,1>

発火 $m_1$ <0,1,2,1> = $m_0$ <1,3,1,0> -m Pre $t_0$ <1,2,0,0> +m Post $t_0$ <0,0,1,1>

## 状態遷移の形式化

状態遷移

```
trans_one m t
```

Some (m -m (Pre t) +m (Post t))

(Pre t) <=m m

部分関数と考えられる

else → None

発火可能な場合 → 発火後のマーキング

発火不可能な場合 → None

Notation:

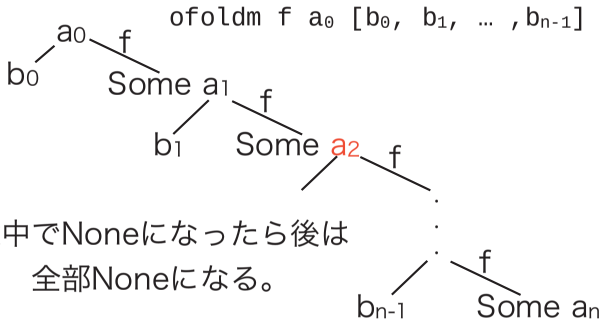
```
m =1{ t }> m' :=(trans_one m t  = Some m')
```

## 状態遷移の形式化

複数stepの遷移

```
foldM :: (Monad m) => (a -> b -> m a) -> a -> [b] -> ma
foldM f a []     = return a
foldM f a(x:s') = f a x >>= \y -> foldM f y s'
ofoldm {A B:Type} (f:A → B → option A) (a:A)(s:seq B) :=
  if s is x::s' then obind (fun fax => ofoldm f fax s')(f a x) else Some a.
```

ofoldm f $a_0$ [$b_0$, $b_1$, … ,$b_{n-1}$]




途中でNoneになったら後は
全部Noneになる。

## Definition

状態遷移

```
trans_seq := ofoldm trans_one
```

Notation:

```
m ={ s }> m' := trans_seq m s = Some m'
```

```
m ={ [t0,t1,t2,t3] }> m'
```

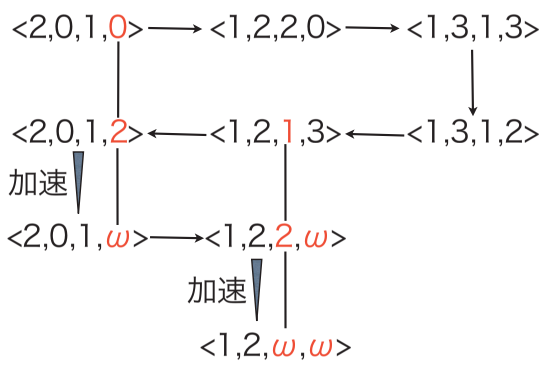
↕

```
m =1{ t0 }> m0 =1{ t1 }> m1 =1{ t2 }> m2
=1{ t3 }> m'
```

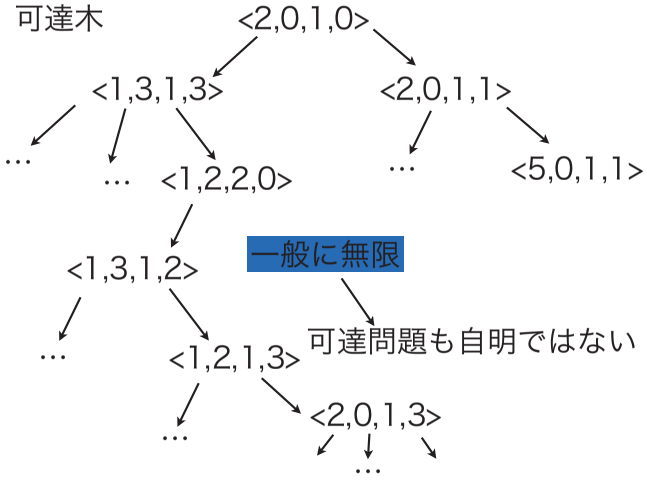
## Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

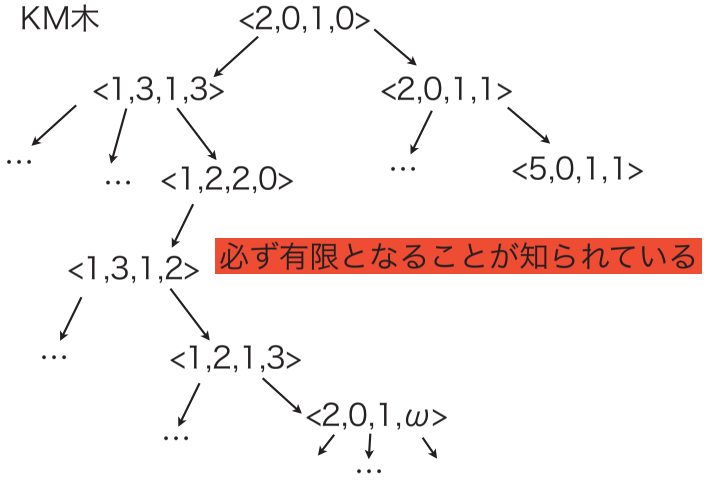
## KM加速




## KM加速

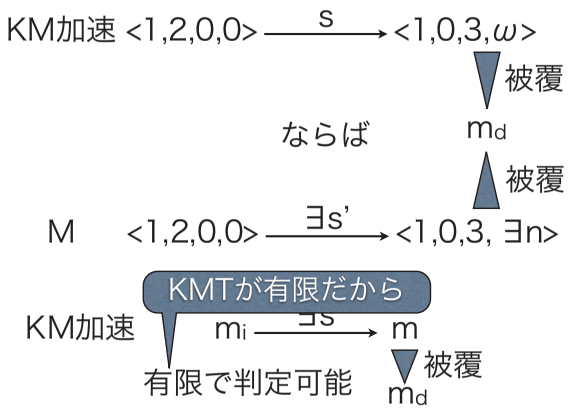



## KM加速



## KM加速

被覆性判定

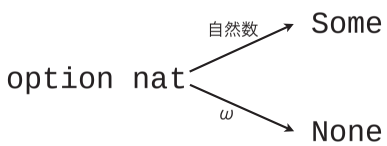



## Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

## KM加速の形式化

ω付きの自然数の定義

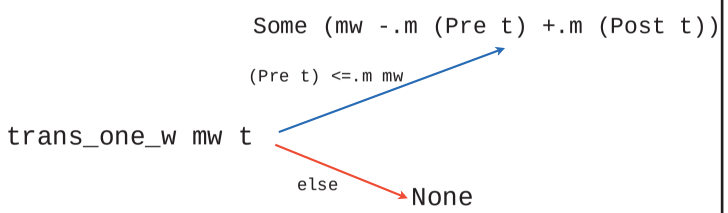



ω付きのマーキングの型

```
Mw:={ffun P -> option nat}
```

## KM加速の形式化

ω付きの遷移



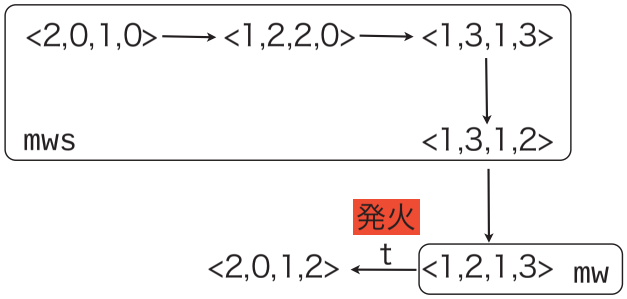

```
trans_seq_w := ofoldm trans_one_w
        m ={w s }> m'
```

## KM加速の形式化

履歴付きのマーキングから履歴付きのマーキングを求める

```
KM_step (mws, mw) t:=
  if trans_one_w mw t is Some mw'
          then Some (rcons mws mw,KM_help mw' (rcons mws mw))
          else None.
```

KM_step (mws,mw) t = Some (mws', mw')

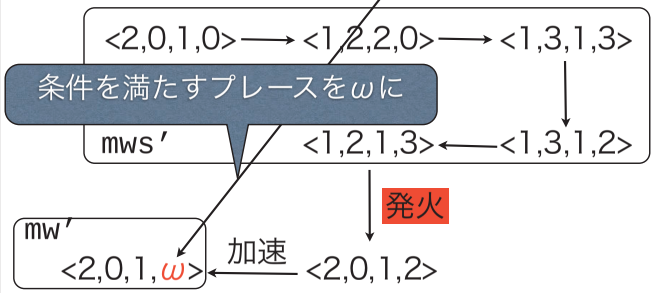



## KM加速の形式化

履歴付きのマーキングから履歴付きのマーキングを求める

```
KM_step (mws, mw) t:=
  if trans_one_w mw t is Some mw'
          then Some (rcons mws mw,KM_help mw' (rcons mws mw))
          else None.
```

KM_step (mws,mw) t = Some (mws', mw')




## KM加速の形式化

1stepのKM加速ができた

複数stepも

これもoptionを返すので

KM_fun := ofoldm KM_step

## まとめ

現在までの成果：ペトリネットの形式化

状態遷移の形式化

KM加速の定義の形式化

被覆性の片方（被覆される方）の形式化

深刻な思い違いが無ければ、後一歩。

今後の課題：被覆性のもう片方

停止性の証明

（これに関してはWQO,Almost-fullが利用できそう[Dim10]）

[Dim10]: Dimitrios et al.,Stop when you are Almost-Full Adventures in constructive termination, 2010

# Towards Verification of Program Generation

Sosuke Moriguchi
Kwansei Gakuin University

## 1 Introduction

A verification system should enable us to write *conditions* (specifications) rather than their results. We have several verification systems such as Frama-C[4], VST[1], and proof-assistants such as Coq, Isabelle, Agda, and so on. These systems are huge and hard to extend them drastically.

When we verify a program *generating other programs*, we would write *conditions about the generated program*. However, generated programs are just data in the generating program, thus we cannot describe *how they behave*. In this paper, we call verification of generated programs through the generating program *verification of program generation*. To support verification of program generation, we have to extend the verification system.

In this paper, we show an idea of extensions for verification systems to support verifying programs generating other programs. In the extensions, we add two functions of the systems:

- Define data types for syntax of languages the systems support. In the systems, we make translators from such data to programs.
- Extend definitions of conditions to describe verification of another program.

Hereafter, we explain these functions and a prototype system for the extensions.

## 2 Verification of Program Generation

In program generations, we have two levels of programs: base programs, which generates other programs based on parameters, and meta programs, which is generated by base programs. In base programs, meta programs are just data describing programs. This means that when we check behaviors of the programs, we have to define new semantics for meta programs, even if the intended language of meta programs is the same as the language of base programs.

To reuse the semantics of the language of the base programs, we make translators from meta programs to base programs. The translators can be defined as functions or relations.

The behaviors of meta programs are just the same as verification, thus conditions of base programs should accept verification. In many verification systems, we cannot describe verification in conditions, so the method requires extensions of definitions of conditions. These two extensions, translators and definitions of conditions, may be not difficult to add existing systems. In this research, we make the difficulties of extensions clear.

Program generation includes compilers since compilers generate programs from inputs. However, in many cases such as CompCert verified compilers[2], semantics for compilers and generated programs are based on separated semantics. We use Coq for the compilers (shallow embedded), but deep embedded syntax (and semantics on it) for generated programs (C or assembly codes). From this reason, the method introduced in this paper is used for existing compilers such as gcc and clang, but not for CompCert compilers.

# 3 Prototype system

To assure the extension is realizable, we implement a prototype verification system for restricted procedural language with Hoare logic. The prototype system is written in Coq proof assistant[3]. You can get source codes of the system from
`https://github.com/chiguri/MetaCert`.

Conditions in the system are in `Prop`, Coq's proposition type. We can describe verification in conditions directly since verification predicate is also in `Prop`. Translations from meta programs to base programs are described by predicates.

The prototype system is still under-developed, but we have an instance of the extension method. This shows the method is realizable.

# 4 Concluding remarks

We proposed a novel kind of extensions of verification systems to support a program generating another program. Currently, we only show the method is realizable at least some verification systems. Extending some existing verification systems such as VST with the method is a future work.

# References


[1] Andrew W. Appel, Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy, and Xavier Leroy. *Program Logics for Certified Compilers*. Cambridge University Press, 2014.

[2] The CompCert project. `http://compcert.inria.fr/`.

[3] The Coq proof assistant reference manual version8.4. `http://coq.inria.fr/distrib/current/files/Reference-Manual.pdf`.

[4] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C - A software analysis perspective. In George Eleftherakis, Mike Hinchey, and Mike Holcombe, editors, *SEFM*, volume 7504 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2012.

# SSReflect による 可変長情報源符号化逆定理の形式化

小尾　良介

joint work with
萩原 学（千葉大学）
山本 光晴（千葉大学）

千葉大学 理学研究科

December 3, 2014





## 背景

- 情報理論の形式化の意義
  - 曖昧な表現の存在 (e.g., $n$ を十分大きくすると ...)
  - 近年の情報理論は証明の大規模化 (e.g., 現代符号理論，シャノン理論).
- Coq/SSReflect による形式化
  - 四色定理（Gonthier，2008）
  - Feit-Thompson の定理（Gonthier et al.，2013）
  - Shannon の定理（Affeldt et al.，2012–2014）





## Shannon の定理

「A Mathematical Theory of Communication」(1948) における，
Shannon の定理一覧

| | Fixed-Length | Variable-Length | Channel coding theory |
|---|---|---|---|
| | Source coding theory | | |
| direct part | ○[1] | ○[2] | ○[1] |
| converse part | ○[1] | ● | ○[3] |

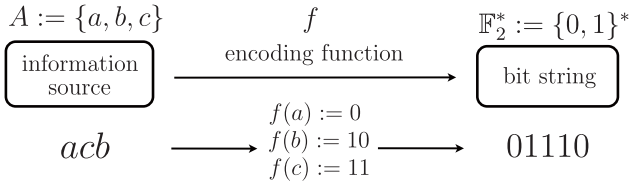
[1] R. Affeldt, M. Hagiwara, Formalization of Shannon's Theorems in SSReflect-Coq, 2012
[2] R. Obi et al., Formalization of the Variable-Length Source Coding Theorem: Direct Part, 2014
[3] R. Affeldt et al., Formalization of Shannon's Theorems, Springer, 2014





## 情報源符号化について 1/2

- 情報源符号化とはデータ圧縮のこと.
- 可変長情報源符号化のモデル




- 情報源が無記憶な分布になってる.
  例： $P(a) = 0.7,\ P(b) = 0.2,\ P(c) = 0.1$

  ある程度長い系列：$accaaababababaaabaaa\ldots$
  $a : b : c = 7 : 2 : 1$

## 情報源符号化について 2/2

- 一意復号可能な符号
  例 ：$A := \{a, b, c\}$
  $f,\ g : A \to \mathbb{F}_2^*$
  $f(a) := 0,\quad f(b) := 10,\quad f(c) := 11$
  $g(a) := 0,\quad g(b) := 1,\quad g(c) := 10$

$$\begin{tikzcd} & 01110 \arrow[r, "f^{-1}"] & acb \\ acb \arrow[ur, "f"] \arrow[dr, "g"'] & & \begin{array}{c} acb \\ \text{or} \\ abab \end{array} \\ & 0101 \arrow[r, "g^{-1}"] & \end{tikzcd}$$

**Definition（一意復号可能な符号 (uniquely decodable codes)）**

符号化写像 $f : A \to \mathbb{F}_2^*$ が一意復号可能な符号であるとは，$f$ の拡張 $f^* : A^* \to \mathbb{F}_2^*$ が単射であることである．





## Informal statement

- $X^n$ ：$n$ 個の確率変数（それぞれが分布 $P$ に従う）
- $H(P)$ ：分布 $P$ のエントロピー
- $\ell(\star)$ ：ビット列 $\star$ の長さ

**Theorem（可変長情報源符号化逆定理）**

任意の一意復号可能な符号 $f : A^n \to \mathbb{F}_2^*$ の平均符号長は，

$$E[\ell(f(X^n))] \geq nH(P).$$

を満足する．

一意復号可能である $f$ による像の長さの平均は，$nH(P)$ ビットより小さくできない．

$P(a) = 0.7,\ P(b) = 0.2,\ P(c) = 0.1$ ならば，圧縮後の長さの平均を $nH(P)(\approx 0.35n)$ ビットより小さくできない．





## Textbook Proof 1/4

**Lemma（正整数を取る確率変数のエントロピーの上界 (Csiszár 1969)）**

正整数を取る確率変数 $N$ について，

$$H(N) \leq \log E[N] + \log e$$

[証明]
確率変数 $N$ に従う確率分布を $P_N$，$\alpha := \log \frac{E[N]}{E[N]-1}$ とする．
このとき，対数和不等式[4] より，

$$\sum_{i=1}^{\infty} P_N(i) \log \frac{P_N(i)}{\alpha^i} \geq \log \frac{1-\alpha}{\alpha}.$$

$\alpha$ の定義の展開，及び，$\forall x \in \mathbb{R}\backslash\{0\}, \log x \leq (x-1)\log e$ が成り立つことから，

$$H(N) \leq \log E[N] + \log e. \quad \square$$

[4] 任意の無限和が収束する非負数列 $\{a_i\}_{i=1}^{\infty}$ 及び $\{b_i\}_{i=1}^{\infty}$ について，$\sum_{i=1}^{\infty} a_i \log a_i/b_i \geq (\sum_{i=1}^{\infty} a_i)\log(\sum_{i=1}^{\infty} a_i)/(\sum_{i=1}^{\infty} b_i)$ が成り立つ．





## Textbook Proof 2/4

**Theorem（可変長情報源符号化逆定理）**

任意の一意復号可能な符号 $f : A^n \to \mathbb{F}_2^*$ の平均符号長は，

$$E[\ell(f(X^n))] \geq nH(P).$$

を満足する．

正整数を取る確率変数 $N := \ell(f(X^n))$ に従う分布を $P_N$，
ビット列 $\mathbb{F}_2^*$ を取る確率変数 $f(X^n)$ に従う分布を $P_f$ とする．

$$\begin{aligned} H(f(X^n)) &= -\sum_{i=1}^{\infty} \sum_{x \in \mathbb{F}_2^i} P_f(x) \log P_f(x) \\ &= \sum_{\substack{i=1 \\ P_N(i) \neq 0}}^{\infty} P_N(i) \underline{\Big( -\sum_{x \in \mathbb{F}_2^i} \frac{P_f(x)}{P_N(i)} \log \frac{P_f(x)}{P_N(i)} \Big)} + H(N). \end{aligned}$$

## Textbook Proof 3/4

性質；エントロピーの上界，及び，補題より，

$$H(f(X^n)) \le \sum_{i=1}^{\infty} iP_N(i) + \log eE[N] = E[N] + \log eE[N]$$
$$= E[\ell(f(X^n))] + \log eE[\ell(f(X^n))]. \qquad \cdots ①$$

case 1： $E[\ell(f(X^n))] \ge n\log|A|$ のとき，
エントロピーの上界より，直ちに題意が満たされる．

$$E[\ell(f(X^n))] \ge n\log|A| \ge nH(P).$$

case 2： $E[\ell(f(X^n))] < n\log|A|$ のとき，① より，

$$H(f(X^n)) \le E[\ell(f(X^n))] + \log en\log|A|.$$

$H(f(X^n)) = H(X^n) = nH(P)$ より，

$$E[\ell(f(X^n))] \ge nH(P) - \log(en\log|A|).$$





## Textbook Proof 4/4

符号化写像 $f : A^n \to \mathbb{F}_2^*$ を $f_m : A^{mn} \to \mathbb{F}_2^*$ に拡張する．

$$\forall a \in A^{mn}, f_m(a) := f(a_1)f(a_2)\cdots f(a_m), (a = a_1a_2\cdots a_m)$$

$f_m$ を先の不等式に適用する，

$$E[\ell(f_m(X^{mn}))] \ge nmH(P) - \log(enm\log|A|).$$

$f_m$ の定義より

$$mE[\ell(f(X^n))] \ge nmH(P) - \log(enm\log|A|).$$

両辺を $m$ で割る

$$E[\ell(f(X^n))] \ge nH(P) - \frac{1}{m}\log(enm\log|A|).$$

$m \to \infty$

$$E[\ell(f(X^n))] \ge nH(P). \qquad \square$$





## Formal statement

- 符号化写像の型（$A^n \to \mathbb{F}_2^*$）

```
Definition var_enc A n := n.-tuple A → seq bool.
Variable f : var_enc A n.
```

- 平均符号長（$E[\ell(f(X^n))]$）の定義

```
Definition E_leng_cw f P :=
    ℰ (mkRvar (P^n) (fun x ⇒(size (f x)))).
```

**Theorem （可変長情報源符号化逆定理）**

```
Variable A : finType.
Variable P : dist A.
Variable n : nat.
Variable f : var_enc A n.

Theorem v_scode_converse :
  uniquely_decodable f →
  E_leng_cw f P ≥ n * ℋ P.
```

**Theorem**

任意の一意復号可能な符号 $f$ の平均符号長は，

$$E[\ell(f(X^n))] \ge nH(P).$$

を満たす．





## 総和の扱い 1/4

- 情報理論では，総和が頻出する．

例）
- 有限集合 $A$ 上の分布 $P$ の満たす性質：$\sum_{a\in A} P(a) = 1$
- 分布のエントロピー：$H(P) := -\sum_{a\in A} P(a)\log P(a)$

- 有限和であれば，SSReflect では bigop 関数を用いて，総和を記述できる．

| 紙上 | SSReflect |
|---|---|
| $\sum_{a\in A} F(a)$ | `\big[Rplus/0]_(a in A) F a`<br>`\rsum_(a in A) F a` |
| $\sum_{i=0}^{n-1} F(i)$ | `\rsum_(0 <= i < n) F i`<br>`\rsum_(i in 'I_n) F i` [5] |

[5] `'I_n`：有限順序数 $n$ $(:= \{0, \cdots, n-1\})$

## 総和の扱い 2/4

- テキストの証明では無限和を扱う．

  例：$\sum_{i=0}^{\infty} P_N(i) = 1$
- Coq は計算可能な関数しか書けない．

  $\Longrightarrow$ 無限和は，bigop 関数だけでは書けないので，述語を使う．
- もし無限和を扱うと，述語原因で，式が煩雑に...

  例：$\sum_{i=0}^{\infty} P_N(i) = 1$，が SSReflect で

```
∀ eps, 0 < eps →
  ∃ n0, (∀ n, (n0 < n)%nat →
            \rsum_(i in 'I_n) PN i = 1).
```

- 無限和を避けたいので，証明を工夫．

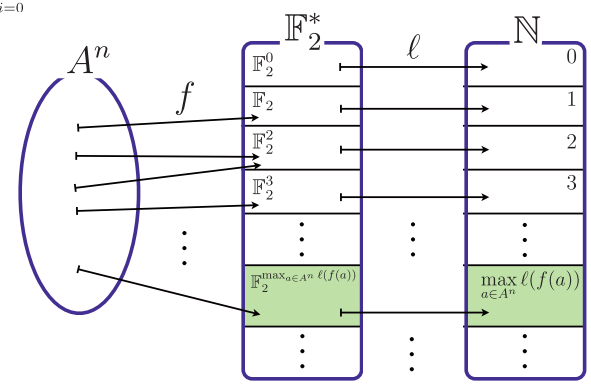




## 総和の扱い 3/4

$N := \ell(f(X^n))$ に従う分布を $P_N$．

$\sum_{i=0}^{\infty} P_N(i) = 1$ だが，無限まで和を取る必要がない．








## 総和の扱い 4/4

- $\mathrm{Nmax} := \max_{a\in A^n} \ell(f(a))$ とする．

  $\forall m$，$\mathrm{Nmax} < m$ ならば $P_N(m) = 0$ なので...

$$\sum_{i=0}^{\infty} P_N(i) = \sum_{i=0}^{\mathrm{Nmax}} P_N(i),$$

$$E[N] = \sum_{i=0}^{\infty} iP_N(i) = \sum_{i=0}^{\mathrm{Nmax}} iP_N(i).$$

- Nmax に置き換えることで，bigop 関数が使える：

```
\rsum_(i in 'I_Nmax+1) PN i = 1.
ℰ N = \rsum_(i in 'I_Nmax+1) i * PN i = 1.
```





## $\varepsilon - \mathrm{N}$ 論法

無限和を避けることはできたが，証明の最後の

$$E[\ell(f(X^n))] \geq nH(P) - \frac{1}{m}\log(enm\log|A|).$$

$$m \to \infty$$

$$E[\ell(f(X^n))] \geq nH(P).$$

は極限を取らざるを得ない．

Coq では，$\varepsilon - \mathrm{N}$ 論法を扱うための定理がある：

```
Lemma le_ε : ∀ r1 r2,
  (∀ eps:R, 0 < eps → r1 ≤ r2 + eps) → r1 ≤ r2.
```

ここで，任意に取れる $m$ を以下の値に定めた：

$$m := \max\left[\lfloor e^{\lceil \log_e(ne\log|A| + n\varepsilon\log_e 2)\rceil}\rfloor, \lfloor e^{\lceil \frac{4}{n\varepsilon\log_e 2}\rceil}\rfloor, 1\right].$$

## Lines of the proof

Textbook Proof [6]

4pages

6weeks

Formal Proof

```
Theorem v_scode_converse :
 uniquely_decodable f ->
 E_leng_cw f P >= INR n * `H P.
Proof.
```

⋮

940lines

[6] 植松友彦，植松 友彦，現代シャノン理論 タイプによる情報理論，培風館，1998





## まとめ

- 本研究にて形式化した定理：
  - 可変長情報源符号化逆定理
  - $H(P^n) = nH(P)$
  - 一意復号可能な符号の性質，

    符号化写像 $f$ が一意復号可能ならば $\begin{cases} f \text{ は単射.} \\ f \text{ の拡張も一意復号可能} \end{cases}$
- 今後の研究題目：
  - 別証明での，可変長情報源符号化定理の形式化

# Formalizing Strong Normalization Proofs

Kazuhiko Sakaguchi

College of Information Science, University of Tsukuba

2014/12/3 TPP2014

## Strong Normalization Theorem

In typed $\lambda$-calculi, strong normalization (SN) theorem is as follows.

$$\forall \Gamma, t, \tau.\ \underbrace{\Gamma \vdash t : \tau}_{\text{If } t \text{ is a typed term,}} \Rightarrow \underbrace{\mathrm{SN}(t)}_{\text{then all reduction sequences from } t \text{ are finite.}}$$

Non-terminating example of a untyped $\lambda$-term:

$$\begin{aligned} & (\lambda x.\, x\, x)\ (\lambda x.\, x\, x) \\ \to_\beta\ & (\lambda x.\, x\, x)\ (\lambda x.\, x\, x) \end{aligned}$$



## Our Formalization of the $\lambda$-Calculus [Sak14]

- https://github.com/pi8027/lambda-calculus
- Goal
  - Formalize many differnt proofs of the strong normalization theorem in Coq.
  - Build a general framework for formalizations of the strong normalization theorem.
- Current developments
  - using de Bruijn representation
  - untyped $\lambda$-calculus
    - Church-Rosser theorem
  - simply typed $\lambda$-calculus ($\lambda^{\to}$) and System F ($\lambda 2$)
    - subject reduction theorem
    - strong normalization theorem (contains 3 different definitions of the reducibility for each system)



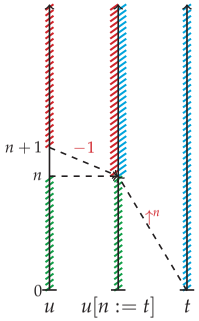
## Substitution for Nameless Terms

$$m[n := t] = \begin{cases} m - 1 & \text{if } n < m \\ t \uparrow^n & \text{if } n = m \\ m & \text{if } n > m \end{cases}$$

$$(u\, v)[n := t] = u[n := t]\, u[n := t]$$

$$(\lambda u)[n := t] = \lambda u[n + 1 := t]$$

$\mathbf{t} \uparrow^{\mathbf{n}}$ is a term which is obtained by adding $n$ to all the free variables of $t$. This operation is called a **shift** or **lift**.



## Shift

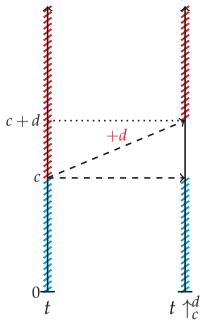
$\mathbf{t} \uparrow^{\mathbf{d}}_{\mathbf{c}}$ adds $d$ to all the free variable of $t$ that are greater than or equal to $c$.

$$n \uparrow^d_c = \begin{cases} n + d & \text{if } c \leq n \\ n & \text{if } n < c \end{cases}$$

$$(t\,u) \uparrow^d_c = t \uparrow^d_c \; u \uparrow^d_c$$

$$(\lambda\, t) \uparrow^d_c = \lambda\, t \uparrow^d_{c+1}$$

$$t \uparrow^d = t \uparrow^d_0$$






## (Restricted) Parallel Substitution

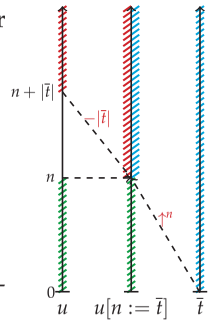
$\mathbf{u}[\mathbf{n} := \mathbf{t_1}, \ldots, \mathbf{t_m}]$ substitutes $t_1, \ldots, t_m$ for free variables $n, \ldots, n+m-1$ in $u$.

$$m[n := \overline{t}] = \begin{cases} m - |\overline{t}| & \text{if } n + |\overline{t}| \leq m \\ \overline{t}_{m-n} \uparrow^n & \text{if } n \leq m < n + |\overline{t}| \\ m & \text{if } m < n \end{cases}$$

$$(u\,v)[n := \overline{t}] = (u[n := \overline{t}])\,(v[n := \overline{t}])$$

$$(\lambda\, u)[n := \overline{t}] = \lambda\,(u[n+1 := \overline{t}])$$

This is useful for proving the strong normalization theorem.






## Equational Properties

of Shift and Parallel Substitution

$$t \uparrow^0_n = t \tag{1}$$

$$c \leq c' \leq c + d \Rightarrow t \uparrow^d_c \uparrow^{d'}_{c'} = t \uparrow^{d'+d}_c \tag{2}$$

$$c' \leq c \Rightarrow t \uparrow^d_c \uparrow^{d'}_{c'} = t \uparrow^{d'}_{c'} \uparrow^d_{d'+c} \tag{3}$$

$$c \leq n \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_c [d + n := \overline{u}] \tag{4}$$

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}] \tag{5}$$

$$c \leq n \wedge |\overline{u}| + n \leq d + c \Rightarrow$$

$$t \uparrow^d_c [n := \overline{u}] = t \uparrow^{d-|\overline{u}|}_c \tag{6}$$

$$m \leq n \Rightarrow t[m := \overline{u}][n := \overline{v}] = t[|\overline{u}| + n := \overline{v}][m := \overline{u}[n - m := \overline{v}]] \tag{7}$$

$$t[|\overline{v}| + n := \overline{u}][n := \overline{v}] = t[n := \overline{v} +\!\!+ \overline{u}] \tag{8}$$

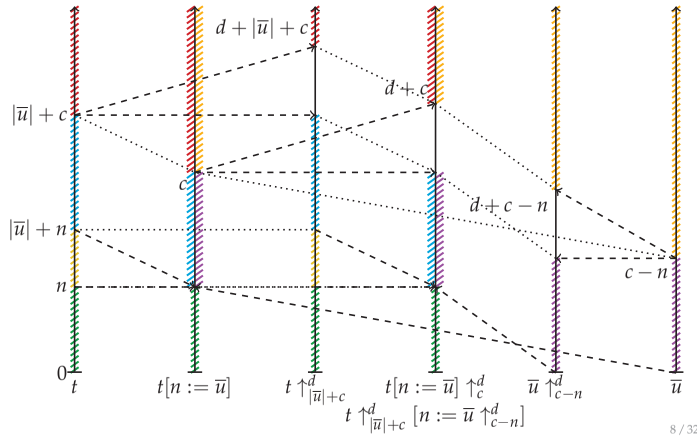
$$t[n := []] = t \tag{9}$$

where
$$\overline{t} \uparrow^d_c = [t \uparrow^d_c \mid t \leftarrow \overline{t}]$$
$$\overline{t}[n := \overline{u}] = [t[n := \overline{u}] \mid t \leftarrow \overline{t}]$$



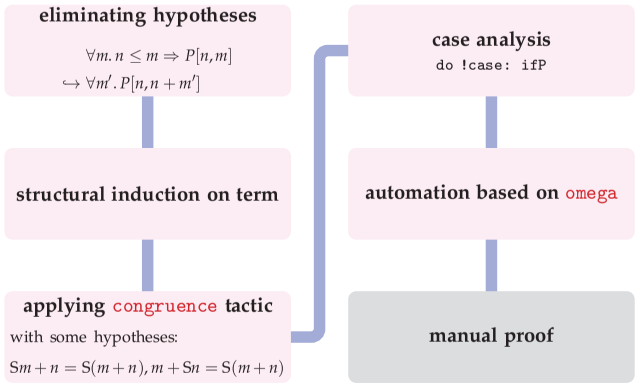
## Equational Property (5)

$$n \leq c \Rightarrow t[n := \overline{u}] \uparrow^d_c = t \uparrow^d_{|\overline{u}|+c} [n := \overline{u} \uparrow^d_{c-n}]$$



## Proof Outline of the Equational Properties

We can apply some automation techniques for these proofs.






## Performance Problem of omega

minn is the smallest number of two natural numbers.

$x - (x - y)$ is a frequently appearing pattern in proofs relevant to the de Bruijn representation.

```
Notation minn x y := (x - (x - y)).

Lemma minnA x y z :
  minn x (minn y z) = minn (minn x y) z.
Proof. omega.
```

associativity of minn

**runs forever**

Some techniques are required to use the omega tactic for proving the equational properties.



## $\lambda^{\to}$: Simply Typed $\lambda$-Calculus

**types and terms:**

$$U ::= X \mid (U \to U)$$

$$t ::= x \mid (t\,t) \mid (\lambda x : U.\, t)$$

**typing rules:**

$$\frac{\Gamma(x) = U}{\Gamma \vdash x : U}$$

$$\frac{\Gamma \vdash t : U \to V \qquad \Gamma \vdash u : U}{\Gamma \vdash t\,u : V}$$

$$\frac{\{x : U\} + \Gamma \vdash t : V}{\Gamma \vdash \lambda x : U.\, t : U \to V}$$

**reduction rules:**

$$(\lambda x : U.\, t)\, u \to_\beta t[x := u]$$

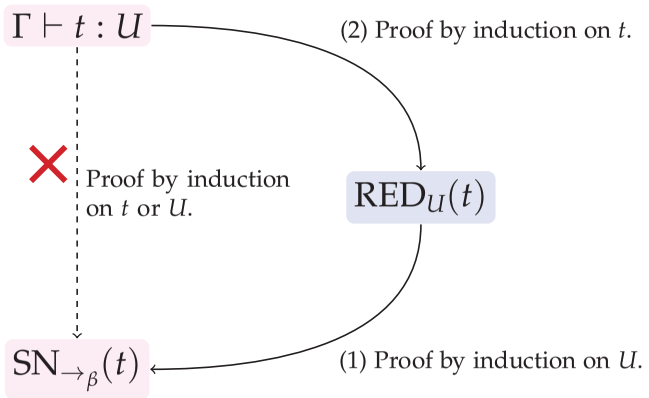
$$\frac{t_1 \to_\beta t_2}{t_1\, u \to_\beta t_2\, u} \qquad \frac{u_1 \to_\beta u_2}{t\, u_1 \to_\beta t\, u_2}$$

$$\frac{t \to_\beta t'}{\lambda x : U.\, t \to_\beta \lambda x : U.\, t'}$$



## Proof Outline of the SN Theorem

Our proofs are based on the Girard's proof [GTL89].



## What is $\mathrm{RED}_U$?

in the Simply Typed $\lambda$-Calculus

$\mathrm{RED}_U$ (**reducibility**) is defined by induction on the type $U$ as follows:

$$\mathrm{RED}_X(t) \overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$

$$\mathrm{RED}_{U\to V}(t) \overset{\text{def}}{\iff} \forall u.\, \mathrm{RED}_U(u) \Rightarrow \mathrm{RED}_V(t\,u).$$

In typical definitions, $\mathrm{RED}_U$ is a set of typed terms. But this definition of $\mathrm{RED}_U$ contains untyped terms. For example,

$$(\lambda x : U.\, x\, x) \in \mathrm{RED}_X.$$



## Part 1: Reducible Terms are SN

$\mathrm{CR}_1$ $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$

$\mathrm{CR}_2$ $t \to_\beta t' \wedge \mathrm{RED}_U(t) \Rightarrow \mathrm{RED}_U(t')$

$\mathrm{CR}_3$ $\mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$

$t$ is not of the form $\lambda x.\, u$.

$\mathrm{CR}_2$ is proved by induction on $U$. $\mathrm{CR}_{1,3}$ are proved together by induction on $U$.



## Part 2: Typed Terms are Reducible

First, we prove the following proposition (reducibility theorem) by induction on $t$.

$$\begin{aligned}\{x_1 : U_1, \ldots, x_n : U_n, y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \\ \wedge (\forall i \in \{1, \ldots, n\}.\, \mathrm{RED}_{U_i}(t_i)) \\ \Rightarrow \mathrm{RED}_U(t[x_1, \ldots, x_n := t_1, \ldots, t_n])\end{aligned}$$

In case of $n = 0$, this proposition is equivalent to

$$\{y_1 : V_1, \ldots, y_m : V_m\} \vdash t : U \Rightarrow \mathrm{RED}_U(t).$$

Finally, we get a proof of the strong normalization theorem.



## Typed Reducibility

Unsuccessful Example

Now, we redefine the reducibility as a set of typed terms.

$$\mathrm{RED}'^{\Gamma}_X(t) \overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$

$$\mathrm{RED}'^{\Gamma}_{U\to V}(t) \overset{\text{def}}{\iff} \forall u.\, \mathrm{RED}^{\Gamma}_U(u) \Rightarrow \mathrm{RED}^{\Gamma}_V(t\,u)$$

$$\mathrm{RED}^{\Gamma}_U(t) \overset{\text{def}}{\iff} \Gamma \vdash t : U \wedge \mathrm{RED}'^{\Gamma}_U(t)$$

In this definition, proof of $\mathrm{CR}_1$ is unsuccessful.

## A Proof of $CR_1$
in Untyped Settings

$CR_1$ $\mathrm{RED}_U(t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$
$CR_3$ $\mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(t')) \Rightarrow \mathrm{RED}_U(t)$
$CR_{1,3}$ *are proved together by induction on* $U$.

**Proof.** If $U$ is a type variable, $CR_1$ is a tautology. The only remaining case is $U = V \to W$.

$$
\begin{aligned}
& \mathrm{RED}_{V\to W}(t) & \\
\Leftrightarrow\ & \forall u.\, \mathrm{RED}_V(u) \Rightarrow \mathrm{RED}_W(t\,u) & \text{(definition of RED)} \\
\Rightarrow\ & \mathrm{RED}_V(x) \Rightarrow \mathrm{RED}_W(t\,x) & (x \text{ is a fresh variable}) \\
\Rightarrow\ & \mathrm{RED}_W(t\,x) & \text{(I.H. of } CR_3) \\
\Rightarrow\ & \mathrm{SN}_{\to_\beta}(t\,x) & \text{(I.H. of } CR_1) \\
\Rightarrow\ & \mathrm{SN}_{\to_\beta}(t) & \text{(basic property of SN)}
\end{aligned}
$$

In typed settings, a term of type $V$ is not always existing.



## Solutions

There are 2 ways to solve this issue:

- Construct finite set of types by traversing proof tree of $\Gamma \vdash t : U$, and add it to $\Gamma$ with fresh variables.
- Redefine $\mathrm{RED}_U$ as a Kripke logical predicate.



## Solution 1: Proof Tree Traversal

$$
\begin{aligned}
\mathcal{T}\left(\frac{\Gamma(x) = U}{\Gamma \vdash x : U}\right) &= \mathcal{T}'(U) \\
\mathcal{T}\left(\frac{P_1 \qquad P_2}{\Gamma \vdash t\,u : V}\right) &= \mathcal{T}'(V) \cup \mathcal{T}(P_1) \cup \mathcal{T}(P_2) \\
\mathcal{T}\left(\frac{P_1}{\Gamma \vdash \lambda x : U.\, t : U \to V}\right) &= \mathcal{T}'(U \to V) \cup \mathcal{T}(P_1)
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{T}'(X) &= \emptyset \\
\mathcal{T}'(U \to V) &= \{U\} \cup \mathcal{T}'(U) \cup \mathcal{T}'(V)
\end{aligned}
$$



## Solution 1: Proof Tree Traversal

It is possible to prove the following $CR_{1,2,3}$ in a similar method.

$CR_1$ $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \mathrm{RED}^\Gamma_U(t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$
$CR_2$ $t \to_\beta t' \wedge \mathrm{RED}^\Gamma_U(t) \Rightarrow \mathrm{RED}^\Gamma_U(t')$
$CR_3$ $(\forall V \in \mathcal{T}'(U).\, V \in \Gamma) \wedge \Gamma \vdash t : U$
$\wedge\, \mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}^\Gamma_U(t')) \Rightarrow \mathrm{RED}^\Gamma_U(t)$

## Solution 2: Kripke Logical Predicate

$\forall x \in \mathrm{dom}(\Gamma).\, \Gamma(x) = \Delta(x)$

$$\mathrm{RED}'_X(\Gamma, t) \overset{\text{def}}{\iff} \mathrm{SN}_{\to_\beta}(t)$$

$$\mathrm{RED}'_{U \to V}(\Gamma, t) \overset{\text{def}}{\iff} \forall \Delta, u.\, \Gamma \leq \Delta \wedge \mathrm{RED}_U(\Delta, u) \Rightarrow \mathrm{RED}_V(\Delta, t\, u)$$

$$\mathrm{RED}_U(\Gamma, t) \overset{\text{def}}{\iff} \Gamma \vdash t : U \wedge \mathrm{RED}'_U(\Gamma, t)$$

$\mathrm{CR}_1$ $\mathrm{RED}_U(\Gamma, t) \Rightarrow \mathrm{SN}_{\to_\beta}(t)$

$\mathrm{CR}_2$ $t \to_\beta t' \wedge \mathrm{RED}_U(\Gamma, t) \Rightarrow \mathrm{RED}_U(\Gamma, t')$

$\mathrm{CR}_3$ $\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathrm{RED}_U(\Gamma, t'))$
$\Rightarrow \mathrm{RED}_U(\Gamma, t)$



## $\lambda 2$: System F [Gir72, GTL89]

**types and terms:**

$U ::= \ldots$
$\quad \mid (\Pi X.\, U)$

$t ::= \ldots$
$\quad \mid (t\, U)$
$\quad \mid (\Lambda X.\, t)$

**additional typing rules:**

$$\frac{\Gamma \vdash t : \Pi X.\, U}{\Gamma \vdash t\, V : U[X := V]}$$

$$\frac{X \notin \Gamma \qquad \Gamma \vdash t : U}{\Gamma \vdash \Lambda X.\, t : \Pi X.\, U}$$

**additional reduction rules:**

$$(\Lambda X.\, t)\, U \to_\beta t[X := U]$$

$$\frac{t_1 \to_\beta t_2}{t_1\, U \to_\beta t_2\, U}$$

$$\frac{t_1 \to_\beta t_2}{\Lambda X.\, t_1 \to_\beta \Lambda X.\, t_2}$$



## Strong Normalization Proofs for System F

It is impossible to define a reducibility for System F directly.
Proof outline of the part 1:

1. Define the **reducibility candidates**. This is a (type indexed) family of terms, and defined by conditions like $\mathrm{CR}_{1,2,3}$.
2. Define the **reducibility with parameters**. This corresponds to the reducibility of $\lambda^{\to}$.
3. Prove that the reducibility with parameters is a reducibility candidate.



## Untyped Reducibility Candidates

Set of terms $\mathcal{R}$ is **reducibility candidate** if and only if

$\mathrm{CR}_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

$\mathrm{CR}_2$ $t \to_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t)$

$\mathrm{CR}_3$ $\mathrm{neutral}(t) \wedge (\forall t'.\, t \to_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

$t$ is not of the form $\lambda x.\, u$ or $\Lambda X.\, u$.

For example, SN is a reducibility candidate.

## Untyped Reducibility with Parameters

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}}](t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\begin{aligned} \mathrm{RED}_{U \to V}[\overline{X} := \overline{\mathcal{R}}](t) &\overset{\text{def}}{\iff} \forall u.\, \mathrm{RED}_U[\overline{X} := \overline{\mathcal{R}}](u) \\ &\qquad \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}}](t\,u) \\ \mathrm{RED}_{\Pi Y.\,U}[\overline{X} := \overline{\mathcal{R}}](t) &\overset{\text{def}}{\iff} \forall V, \mathcal{S}.\, \mathrm{RC}(\mathcal{S}) \\ &\qquad \Rightarrow \mathrm{RED}_U[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}}](t\,V) \end{aligned}$$

**Lemma** If $\overline{\mathcal{R}}$ is a sequence of reducibility candidates, $\mathrm{RED}_U[\overline{X} := \overline{\mathcal{R}}]$ is a reducibility candidate.



## Typed Reducibility Candidates 1 [Hur10]

Set of terms $\mathcal{R}$ is reducibility candidate of $\Gamma, U$ if and only if

$\mathrm{CR}_1$ $\#\Gamma \vdash t : U \wedge \mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

$\mathrm{CR}_2$ $\#\Gamma \vdash t : U \wedge t \to_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

$\mathrm{CR}_3$ $\#\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \to_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

where $\#\Gamma = \{b : \Pi X.\, X\} + \Gamma$

$\#\Gamma \vdash b\,U : U$

$b\,U$ is neutral and normal



## Typed Reducibility with Parameters 1

$$\mathrm{RED}^\Gamma_Y[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}(t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\begin{aligned} \mathrm{RED}^\Gamma_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) &\overset{\text{def}}{\iff} \forall u.\, \#\Gamma \vdash u : V[\overline{X} := \overline{U}] \\ &\qquad \Rightarrow \mathrm{RED}^\Gamma_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}](u) \\ &\qquad \Rightarrow \mathrm{RED}^\Gamma_W[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t\,u) \\ \mathrm{RED}^\Gamma_{\Pi Y.\,V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](t) &\overset{\text{def}}{\iff} \forall W, \mathcal{S}.\, \mathrm{RC}^\Gamma_W(\mathcal{S}) \\ &\qquad \Rightarrow \mathrm{RED}^\Gamma_V[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\,W) \end{aligned}$$

**Lemma** If $\overline{\mathcal{R}}_i$ is a reducibility candidate of $\Gamma, \overline{U}_i$ for all $i \leq |\overline{X}|$, $\mathrm{RED}^\Gamma_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}]$ is a reducibility candidate of $\Gamma, V[\overline{X} := \overline{U}]$.



## Typed Reducibility Candidates 2 [Gal89]

Set of pairs of type environment and term $\mathcal{R}$ is reducibility candidate of type $U$ if and only if

$\mathrm{CR}_{\text{typed}}$ $\mathcal{R}(\Gamma, t) \Rightarrow \Gamma \vdash t : U$

$\mathrm{CR}_0$ $\Gamma \leq \Delta \wedge \mathcal{R}(\Gamma, t) \Rightarrow \mathcal{R}(\Delta, t)$

$\mathrm{CR}_1$ $\mathcal{R}(t) \Rightarrow \mathrm{SN}(t)$

$\mathrm{CR}_2$ $t \to_\beta t' \wedge \mathcal{R}(t) \Rightarrow \mathcal{R}(t')$

$\mathrm{CR}_3$ $\Gamma \vdash t : U \wedge \mathrm{neutral}(t) \wedge (\forall t'.t \to_\beta t' \Rightarrow \mathcal{R}(t')) \Rightarrow \mathcal{R}(t)$.

## Typed Reducibility with Parameters 2

$$\mathrm{RED}_Y[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} \begin{cases} \overline{\mathcal{R}}_i(\Gamma, t) & \text{if } Y = \overline{X}_i \\ \mathrm{SN}'(\Gamma, t) & \text{if } Y \notin \overline{X} \end{cases}$$

$$\begin{aligned} \mathrm{RED}_{V \to W}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} & \ \Gamma \vdash t : V \to W \\ & \wedge (\forall \Delta, u. \Gamma \leq \Delta \\ & \qquad \Rightarrow \mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, u) \\ & \qquad \Rightarrow \mathrm{RED}_W[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Delta, t\,u)) \end{aligned}$$

$$\begin{aligned} \mathrm{RED}_{\Pi Y.\,V}[\overline{X} := \overline{\mathcal{R}} : \overline{U}](\Gamma, t) \overset{\text{def}}{\iff} & \ \forall W, \mathcal{S}. \mathrm{RC}_W(\mathcal{S}) \\ & \qquad \Rightarrow \mathrm{RED}_V[Y, \overline{X} := \mathcal{S}, \overline{\mathcal{R}} : W, \overline{U}](t\,W) \end{aligned}$$

**Lemma** If $\overline{\mathcal{R}}_i$ is a reducibility candidate of $\overline{U}_i$ for all $i \leq |\overline{X}|$, $\mathrm{RED}_V[\overline{X} := \overline{\mathcal{R}} : \overline{U}]$ is a reducibility candidate of $V[\overline{X} := \overline{U}]$.



## Comparison of the SN Proofs

- SN proofs with typed reducibility requires type preservation lemmas. On the other hand, SN proofs with untyped reducibility are completed without type preservation lemmas. (Untyped proofs are relatively simple.)
- Typed reducibilities are capturing the features of reducible terms.



## Conclusion

- We formalized strong normalization proofs with 6 different definitions of the reducibility.
- `$ wc -lc **/*.v`

```
...
  1808  72327 coq/LC/Debruijn/F.v
   647  24413 coq/LC/Debruijn/STLC.v
...
  3746 138149 total
```

- https://github.com/pi8027/lambda-calculus



## Bibliography I


Jean H. Gallier.
On Girard's "candidats de reductibilité".
In *Logic and Computer Science*. Academic Press, 1989.

Jean-Yves Girard.
*Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.*
PhD thesis, Université de Paris 7, 1972.

Jean-Yves Girard, Paul Taylor, and Yves Lafont.
*Proofs and Types*.
Cambridge University Press, 1989.

Chung-Kil Hur.
Heq : a Coq library for heterogeneous equality, 2010.
URL: http://sf.snu.ac.kr/gil.hur/Heq/.

Kazuhiko Sakaguchi.
A formalization of typed and untyped $\lambda$-calculus in SSReflect-Coq and Agda2, 2011-2014.
URL: https://github.com/pi8027/lambda-calculus.

## A Name-Free Lambda Calculus

Masahiko Sato

Graduate School of Informatics, Kyoto University

(Joint work with Randy Pollack and Takafumi Sakurai)

TPP 2014
Kyushu Universrity
December 3, 2014

## Raw $\lambda$-terms

Definition of raw lambda-terms.

$$\Lambda \ni M, N, P ::= x \mid (M\ N) \mid \lambda x.\ M$$

$(M\ N)$ stands for the application of (function) $M$ to $N$.

We write $[x := N]M$ for the result of subsittuting $N$ for $x$ in $M$.

## $\lambda_\beta$-calculus

$$\frac{}{(\lambda x.\ M\ N) \to_\beta [x := N]M}\ \beta$$

$$\frac{M \to_\beta M'}{(M\ N) \to_\beta (M'\ N)}\ \mathsf{L} \qquad \frac{N \to_\beta N'}{(M\ N) \to_\beta (M\ N')}\ \mathsf{R}$$

$$\frac{M \to_\beta N}{\lambda x.\ M \to_\beta \lambda x.\ N}\ \xi$$

$$\frac{}{M \to_\beta M}\ \mathsf{Rfl} \qquad \frac{M \to_\beta N \quad N \to_\beta P}{M \to_\beta P}\ \mathsf{Trn}$$

The $\beta$-rule captures the informal notion of function application.

## Problems with raw $\lambda$-terms

A problem with raw lambda-terms is that substitution is non-trivial.

Let $M$ be $\lambda y.\ (x\ y)$ and $N$ be $y$. Then, what is $[x := N]M$?

$[x := y]\lambda y.\ (x\ y) = \lambda y.\ (y\ y)$ is not correct. $y$ was a free variable before substitution, but it becomes a bound variable after susbstitution.

The problem is solved by renaming $y$ in $M$ to a fresh varaible $z$. Then, $[x := y]\lambda z.\ (x\ z) = \lambda z.\ (y\ z)$.

We replaced $M = \lambda y.\ (x\ y)$ by $M' = \lambda z.\ (x\ z)$ which is obtained by renaming. Such a pair $M$ and $M'$ are called $\alpha$-equivalent.

## Problems with raw $\lambda$-terms (cont.)

A second problem with raw $\lambda$-terms is this:

$$\Lambda \ni M, N, P ::= x \mid (M\ N) \mid \lambda x.\ M$$

In this defintion, in order to generate a closed $\lambda$-term, say, $\lambda x.\ x$, one must first generate a term $x$ which is not a closed $\lambda$-term.

I think this is the most problematic point of the definition of raw $\lambda$-terms.

## Problems with raw $\lambda$-terms (cont.)

A third problem with raw $\lambda$-terms is that one cannot develop the $\lambda$-calculus on the structure of raw $\lambda$-terms.

One has to define $\lambda$-terms by quotiating raw $\lambda$-terms by the $\alpha$-equivalence relation.

Thus, we have the set of $\lambda$-terms

$$[\Lambda] := \Lambda / \equiv_\alpha$$

and this forces us to always deal with two kinds of terms, namely, raw $\lambda$-terms and $\lambda$-terms.

This might be OK in informal mathematics, but it becomes a very subtle issue when one works in a proof assistant (which is a computer program to support formal mathematical reasoning).

## Problems with raw $\lambda$-terms (cont.)

A fourth problem that the notion of immediate subterm becomes obscure on raw $\lambda$-terms.

For example what is the immediate subterm of

$$\lambda x.\ \lambda y.\ (x\ y)?$$

One may say the answer is $\lambda y.\ (x\ y)$ (with $x$ free).

But, then what about

$$\lambda y.\ \lambda x.\ (y\ x)?$$

Your answer should be $\lambda x.\ (y\ x)$ (with $y$ free).

Since initial two terms are $\alpha$-equivalent, the answers must also be $\alpha$-equivalent. But, this is not the case here.

## Problems with raw $\lambda$-terms (cont.)

All of thse problems boil down to the following problem.

1. The raw $\lambda$-terms $\lambda x.\ x$ and $\lambda y.\ y$ are two distinct raw $\lambda$-terms (since they are syntactically different).
2. However, we somehow wish to identify them. And we do this by quotienting $\Lambda$ by the $\alpha$-equivalence relation.

My aim in this talk is to solve this prblem without using the notions of substitution or equivalence relation.

## Structure of raw $\lambda$-terms

To see the essence of the $\alpha$-equivalence relation, we make the following observation.
Recall that:

$$\Lambda \ni M, N, P ::= x \mid (M\ N) \mid \lambda x.\ M$$

Any $\lambda$-term can be uniquely written in one of the following two forms.

1. $\lambda x_0 x_1 \cdots x_{n-1}.\ y$.
2. $\lambda x_0 x_1 \cdots x_{n-1}.\ (M\ N)$.

## Structure of raw $\lambda$-terms (cont.)

1. (Type I) $\lambda x_0 x_1 \cdots x_{n-1}.\ y$.
2. (Type II) $\lambda x_0 x_1 \cdots x_{n-1}.\ (M\ N)$.

The above classification gives us the following 3 invariants under $\alpha$-equivalence. For each raw $\lambda$-term, we can assign the following 3 values.

1. height is $n$ (as shown above).
2. thickness is 1 if it is of type I; otherwise, it is the sum of the thickness of its two direct sub-terms $\lambda x_0 x_1 \cdots x_{n-1}.\ M$ and $\lambda x_0 x_1 \cdots x_{n-1}.\ N$.
3. shape is defined in the next slide.

The last invariant shape completely characterize raw $\lambda$-terms up to $\alpha$-equivalence.

## Shape of raw $\lambda$-term

We define the shape of a term as follows. We write $\bar{x}$ for $x_0 x_1 \cdots x_{n-1}$.

1. $\mathrm{shape}(\lambda \bar{x}.\ y) := \mathsf{I}^n_y$ if $y \notin \bar{x}$.
2. $\mathrm{shape}(\lambda \bar{x}.\ y) := \mathsf{I}^n_i$, if $y \in \bar{x}$ and $i$ is the largest index such that $y = x_i$.
3. $\mathrm{shape}(\lambda \bar{x}.\ (M\ N)) := \mathsf{II}^n(\mathrm{shape}(\lambda \bar{x}.\ M), \mathrm{shape}(\lambda \bar{x}.\ N))$.

That shape is invariant under $\alpha$-equivalence is intuitively obvious. We use this intuitive idea to define the notion of $\alpha$-equivalence formally.

## Definition of $\alpha$-equivalence relation

$$\frac{\mathrm{shape}(\lambda \bar{x}.\ u) = \mathrm{shape}(\lambda \bar{y}.\ v)}{\lambda \bar{x}.\ u \equiv_\alpha \lambda \bar{y}.\ v}$$

$$\frac{\lambda \bar{x}.\ M \equiv_\alpha \lambda \bar{y}.\ M' \quad \lambda \bar{x}.\ N \equiv_\alpha \lambda \bar{y}.\ N'}{\lambda \bar{x}.\ (M\ N) \equiv_\alpha \lambda \bar{y}.\ (M'\ N')}$$

## An example

$$\dfrac{\dfrac{}{\lambda xy.\ x \equiv_\alpha \lambda yx.\ y}\ \mathsf{I}_0^2 \quad \dfrac{}{\lambda xy.\ y \equiv_\alpha \lambda yx.\ x}\ \mathsf{I}_1^2}{\lambda xy.\ (x\ y) \equiv_\alpha \lambda yx.\ (y\ x)}\ \mathsf{II}^2$$

This derivation is essentially the same as the following derivation tree.

$$\dfrac{\mathsf{I}_0^2 \quad \mathsf{I}_1^2}{\mathsf{II}^2}$$

Or:

$$\mathsf{II}^2(\mathsf{I}_0^2, \mathsf{I}_1^2)$$

## Definition of $\alpha$-equivalence relation (cont.)

$$\dfrac{\mathrm{shape}(\lambda\bar{x}.\ u) = \mathrm{shape}(\lambda\bar{y}.\ v) = \mathsf{I}_u^n}{\lambda\bar{x}.\ u \equiv_\alpha \lambda\bar{y}.\ v}\ \mathsf{I}_u^n$$

$$\dfrac{\mathrm{shape}(\lambda\bar{x}.\ u) = \mathrm{shape}(\lambda\bar{y}.\ v) = \mathsf{I}_i^n}{\lambda\bar{x}.\ u \equiv_\alpha \lambda\bar{y}.\ v}\ \mathsf{I}_i^n$$

$$\dfrac{\lambda x.\ M \equiv_\alpha \lambda y.\ M' \quad \lambda x.\ N \equiv_\alpha \lambda y.\ N'}{\lambda\bar{x}.\ (M\ N) \equiv_\alpha \lambda\bar{y}.\ (M'\ N')}\ \mathsf{II}^n$$

Here, we can observe that all the $\alpha$-equivalent closed $\lambda$-terms are generated without using the first rule.

This observation naturally led me to define a new definition of closed $\lambda$-terms which is completely name-free. No variables are used in the definition.

## $\mathbb{L}$-terms

We write $\mathbb{N}$ for the set of natural numbers 0, 1, 2, $\cdots$.

$$\dfrac{n \in \mathbb{N} \quad i \in \mathbb{N}}{\mathsf{J}_i^n \in \mathbb{L}} \qquad \dfrac{n \in \mathbb{N} \quad M \in \mathbb{L} \quad N \in \mathbb{L}}{(M\ N)^n \in \mathbb{L}}$$

Height

1. $\mathrm{ht}(\mathsf{J}_i^n) := n$.
2. $\mathrm{ht}((M\ N)^n) := n$.

Thickness.

1. $\mathrm{th}(\mathsf{J}_i^n) := 1$.
2. $\mathrm{th}((M\ N)^n) := \mathrm{th}(M) + \mathrm{th}(N)$.

## Admissible $\mathbb{L}$-terms: $\{M\}_k$

We will write $\{M\}_k^n$ for $\{M\}_k \wedge n \leq \mathrm{ht}(M)$.

1. $\{\mathsf{J}_i^n\}_k := i < n + k$.
2. $\{(M\ N)^n\}_k := \{M\}_k^n \wedge \{N\}_k^n$.

Example (Admissible $\mathbb{L}$-terms)

1. $\mathsf{I} := \mathsf{J}_0^1 \quad (\lambda x.\ x)$.
2. $\mathsf{K} := \mathsf{J}_0^2 \quad (\lambda xy.\ x)$.
3. $\mathsf{S} := ((\mathsf{J}_0^3\ \mathsf{J}_2^3)^3\ (\mathsf{J}_1^3\ \mathsf{J}_2^3)^3)^3 \quad (\lambda xyz.\ ((x\ z)\ (y\ z)))$.

Admissible terms $\{M\}_k$ exactly correspond to raw $\lambda$-terms (moudlo $\alpha$-equivalence) whose free variables are among $x_0, \ldots, x_{k-1}$.

## Thread

An $\mathbb{L}$-term $\mathsf{J}_i^n$ is called a thread.

Definition (Projection)

A thread $\mathsf{J}_i^n$ is called a projection if $i < k$. It represents a projection function of arity $n$. Given $n$ arguments, it returs the $i$-th argmuent (counting from 0).

Definition (Parameter)

A thread $\mathsf{J}_i^0$ is called a parameter, and it is sometimes written $\mathsf{X}_i$. It is a function of arity 0, and hence it is a constant.

## Ground level instantiation function

We put $\mathbb{L}^n := \{M \in \mathbb{L} \mid \{M\}_0^n\}$ and $\mathbb{L}_k^n := \{M \in \mathbb{L} \mid \{M\}_k^n\}$.

Definition (Lifting: $\Uparrow^l \cdot : \mathbb{L}^k \to \mathbb{L}^{k+n}$)

1. $\Uparrow^n \mathsf{J}_i^k := \mathsf{J}_{i+n}^{k+n}$.
2. $\Uparrow^n (M\ N)^k := (\Uparrow^n M\ \Uparrow^n N)^{k+n}$.

Definition (Ground level insantiation $\triangledown : \mathbb{L}^{k+1} \times \mathbb{L} \to \mathbb{L}^k$)

1. $\mathsf{J}_0^k \triangledown P := \Uparrow^{k-1} P$.
2. $\mathsf{J}_{i+1}^k \triangledown P := \mathsf{J}_i^{k-1}$.
3. $(M\ N)^k \triangledown P := (M \triangledown P\ N \triangledown P)^{k-1}$.

## Instantiation function

$$\langle \cdot \quad \cdot \rangle_k^n : \mathbb{L}_k^{n+1} \times \mathbb{L}_k^n \to \mathbb{L}_k^n$$

1. $\langle M\ P \rangle_k^0 := M \triangledown P$.
2. $\langle M\ P \rangle_k^{n+1} := [\langle (M)_k\ (P)_k \rangle_{k+1}^n]_k$.

Instantiation is more fundamental than substitution. We can define substitution using instantiation, but the converse is impossible.

## Opening and Closing

Definition (Opening function: $(\ \cdot\ )_m : \mathbb{L}_m^{k+1} \to \mathbb{L}_{m+1}^k$)

We understand that $i - j$ is 0 when $j > i$.

1. $(\mathsf{J}_i^k)_m := \begin{cases} \mathsf{J}_{k-1}^{k-1} & \text{if } i = 0, \\ \mathsf{J}_{i-1+m}^{k-1} & \text{o.w.} \end{cases}$
2. $((M\ N)^k)_m := ((M)_m\ (N)_m)^{k-1}$.

Definition (Closing function: $[\ \cdot\ ]_m : \mathbb{L}_{m+1}^k \to \mathbb{L}_m^{k+1}$)

1. $[\mathsf{J}_i^k]_m := \begin{cases} \mathsf{J}_0^{k+1} & \text{if } i = k + m, \\ \mathsf{J}_{i+1}^{k+1} & \text{o.w.} \end{cases}$
2. $[(M\ N)^k]_m := ([M]_m\ [N]_m)^{k+1}$.

## $\mathbb{L}_\beta$-calculus

$$\frac{\{M\}_0^{n+1} \quad \{N\}_0^n}{(M\ N)^n \to_{\mathrm{B}} \langle M\ N\rangle_0^n}\ \mathsf{B}$$

$$\frac{M \to_{\mathrm{B}} M'}{(M\ N)^n \to_{\mathrm{B}} (M'\ N)^n}\ \mathsf{L} \qquad \frac{N \to_{\mathrm{B}} N'}{(M\ N)^n \to_{\mathrm{B}} (M'\ N)^n}\ \mathsf{R}$$

$$\frac{}{M \to_{\mathrm{B}} M}\ \mathsf{Rfl} \qquad \frac{M \to_{\mathrm{B}} N \quad N \to_{\mathrm{B}} P}{M \to_{\mathrm{B}} P}\ \mathsf{Trn}$$

$\mathbb{L}_\beta$-calculus is name-free and computes closed terms. In cotrast to this, $\lambda_\beta$-calculus mentions variables:

$$\frac{}{(\lambda x.\ M\ N) \to_\beta [x := N]M}\ \beta \qquad \frac{M \to_\beta N}{\lambda x.\ M \to_\beta \lambda x.\ N}\ \xi$$

## Church-Rosser Theorem

We are now working on the Church-Rosser Theorem for $\mathbb{L}_\beta$ in Minlog.

Theorem
$M \to_{\mathrm{B}} N, M \to_{\mathrm{B}} N' \Longrightarrow \exists P.\ \ N \to_{\mathrm{B}} P \wedge N' \to_{\mathrm{B}} P.$

# FORMALIZATION OF DIRECT SUM DECOMPOSITION OF GROUPS IN MIZAR

December 3rd, 2014
TPP2014 @ Kyushu University

Kazuhisa Nakasho, Hiroshi Yamazaki,
Hiroyuki Okazaki and Yasunari Shidama
@ Shinshu University

## INDEX



## 1. DIRECT PRODUCT VS. DIRECT SUM



1. DIRECT PRODUCT VS. DIRECT SUM

## I . WHAT IS THE DIFFERENCE?

Let I be set, $\{G_i \mid i \in I\}$ be family of groups:

- Direct Product $\prod_{i \in I} G_i$
  - Set of Elements: $\{(g_i)_{i \in I} \mid g_i \in G_i\}$
  - Binary Operation: $(g_i)_{i \in I} \cdot (h_i)_{i \in I} = (g_i \cdot h_i)_{i \in I}$
- Direct Sum $\bigoplus_{i \in I} G_i$
  - Set of Elements: $\{(g_i)_{i \in I} \mid g_i \in G_i,$ $\boldsymbol{g_i = 1_{G_i}}$ **for all but finitely many i**$\}$
  - Binary Operation: $(g_i)_{i \in I} \cdot (h_i)_{i \in I} = (g_i \cdot h_i)_{i \in I}$

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅰ. WHAT IS THE DIFFERENCE?

- The direct sum $\oplus_{i\in I} G_i$ is a subgroup of the direct product $\prod_{i\in I} G_i$.
- If I is finite, the direct product $\prod_{i\in I} G_i$ and the direct sum $\oplus_{i\in I} G_i$ are equal.

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅱ. EXAMPLES

- A Banach space is the completion of the direct sum of 1-dimensional normed vector spaces, and a subset of the direct product.
  - A norm of an infinite sum $\sum_i x_i \mathrm{v_i}$ cannot be defined directly.
  - A finite sum $\sum_i x_i \mathrm{v_i}$ has a norm by definition.
  - The completion of the direct sum is a subset of direct product whose elements satisfies $\lim_{\mathrm{i}\to\infty} \|\sum_i x_i \mathrm{v_i}\| < \infty$
  - $\oplus_{i\in I} V_i \subseteq$ (the completion of $\oplus_{i\in I} V_i$) , $\prod_{i\in I} V_i$

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅲ. FORMALIZATION

**:: The Direct Product of Group**

```
definition
  let I be set, F be multMagma-Family of I;
  func product F -> strict multMagma means
:: GROUP_7:def 2
  the carrier of it = product Carrier F
& for f, g being Element of product Carrier F,
      i being set st i in I
   ex Fi being non empty multMagma, h being Function
   st Fi = F.i & h = (the multF of it).(f,g)
    & h.i = (the multF of Fi).(f.i,g.i);
end;
```

$\{(g_i)_{i\in I} \mid g_i \in G_i\}$

$(g_i)_{i\in I} \cdot (h_i)_{i\in I} = (g_i \cdot h_i)_{i\in I}$

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅲ. FORMALIZATION

**:: The Direct Sum of Group**

```
definition
  let I be set, F be associative Group-like multMagma-Family of I;
  func sum F -> strict Subgroup of product F means
:: GROUP_7:def 9
  for x being object holds x in the carrier of it
  iff ex g being Element of product Carrier F,
        J being finite Subset of I,
        f being ManySortedSet of J
      st g = 1_product F
       & x = g +* f
       & for j being set st j in J
          ex G being Group-like non empty multMagma
          st G = F.j & f.j in the carrier of G & f.j <> 1_G;
end;
```

$\oplus_{i\in I} G_i$ is a subgroup of $\prod_{i\in I} G_i$

Only a finite number of f.j are not $1_G$

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅲ. SOME USEFUL TOOLS

```
definition
  let I be set;
  let F be Group-Family of I;
  let a be Function;
  func support(a,F) -> Subset of I means
:: GROUP_19:def 1
  for i be object holds i in it iff (a.i <> 1_F.i & i in I);
end;

theorem :: GROUP_19:8
  for I be set,
      F be Group-Family of I,
      a be Element of product F
  holds
    a in sum F
  iff
    support(a,F) is finite;
```

$\boxed{\text{i in support(a, F)} \Leftrightarrow \text{a.i} \neq 1}$

$\boxed{\text{a in } \oplus_i F_i \Leftrightarrow \text{support(a,F) is finite}}$

1. DIRECT PRODUCT VS. DIRECT SUM

## Ⅲ. SOME USEFUL TOOLS

```
theorem :: GROUP_19:11
  for I be non empty set,
      F be Group-Family of I
  holds
  support(1_product F,F) is empty;

theorem :: GROUP_19:17
  for I be non empty set,
      F be Group-Family of I,
      x be Element of product F,
      i be Element of I,
      g be Element of F.i
  st x = 1_product F +* (i,g)
  holds support(x,F) c= {i};

theorem :: GROUP_19:22
  for I be non empty set,
      F be Group-Family of I,
      x be Function,
      i be Element of I,
      g be Element of F.i
  holds
    support(x,F) is finite
    implies support(x+*(i,g),F) is finite;

theorem :: GROUP_19:27
  for I be non empty set,
      F be Group-Family of I,
      i be object,
      a,b be Function
  st i in I & dom a = I & b = a +* (i,1_F.i)
  holds support(b,F) = support(a,F) ¥ {i};

theorem :: GROUP_19:31
  for I be non empty set,
      F be Group-Family of I,
      a,b be Element of product F
  st support(a,F) misses support(b,F)
  holds a +* (b|support(b,F)) = a * b;

theorem :: GROUP_19:32
  for I be non empty set,
      F be Group-Family of I,
      a,b be Element of product F
  st support(a,F) misses support(b,F)
  holds a * b = b * a;
```

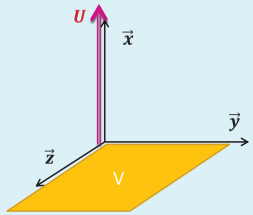
## 2. EXTERNAL DIRECT SUM DECOMPOSITION VS. INTERNAL DIRECT SUM DECOMPOSITION

I. Example
II. What is the difference?
III. Formalization
IV. Other equivalent expressions of the internal direct sum decomposition
V. Equivalence between internal and external direct sum decomposition

2. EXTERNAL DIRECT SUM DECOMPOSITION VS. INTERNAL DIRECT SUM DECOMPOSITION

## Ⅰ. EXAMPLE

Vector Space




$W = \{a\vec{x} + b\vec{y} + c\vec{z} \mid a, b, c \in \mathbb{R}\}$
$U = \{a\vec{x} \mid a \in \mathbb{R}\}$
$V = \{b\vec{y} + c\vec{z} \mid b, c \in \mathbb{R}\}$

$W = U \oplus V$
$\Leftrightarrow U \oplus V$ is the direct sum decomposition of W

## Ⅱ. WHAT IS THE DIFFERENCE?

Let $\{H_i\}$ be a direct sum decomposition of $G$

- It is external
  $\Leftrightarrow \quad \exists f\colon \oplus_{i\in I} H_i \to G \quad$ (isomorphism)
- It is internal
  $\Leftrightarrow$ 1. $\{H_i\}$ are subgroups of G
  2. natural mapping $\oplus_{i\in I} H_i \to G$ is isomorphism
- internal $\Rightarrow$ external



## Ⅲ. FORMALIZATION

**:: External Direct Sum Decomposition**

```
definition
  let I be non empty set;
  let G be Group;
  mode DirectSumComponents of G,I -> Group-Family of I
means
:: GROUP_19:def 8
  ex h be Homomorphism of sum it, G st h is bijective;
end;
```

$\{F_i\}$ *is* **a direct sum decomposition of** $G$

$h\colon \oplus F_i \to G$ **is isomorphism**



## Ⅲ. FORMALIZATION

**:: Internal Direct Sum Decomposition**

```
definition
  let I be non empty set;
  let G be Group;
  let F be DirectSumComponents of G,I;
  attr F is internal means
:: GROUP_19:def 9
   (for i be Element of I holds F.i is Subgroup of G)
  & ex h be Homomorphism of sum F, G
     st h is bijective
     & for x be finite-support Function of I,G
        st x in sum F holds h.x = Product x;
end;
```

$h\colon \oplus F_i \to G$ **is isomorphism**

$h\colon (x_i) \mapsto \prod_i x_i$ **(natural mapping)**



## Ⅲ. FORMALIZATION

**:: Product**

```
definition
  let A be finite set;
  func canFS A -> FinSequence of A equals
:: FINSEQ_1:def 19
  the one-to-one onto FinSequence of A;
end;
```

**canFS(A): {1,2,...,n}->A (not unique!)**

```
definition
  let G be non empty multMagma,
  I be finite set,
  b be (the carrier of G)-valued total I -defined Function;
  func Product b -> Element of G means
:: GROUP_17:def 1
  ex f being FinSequence of G st it = Product f & f = b*canFS(I);
end;
```

**Product f = f(1) * f(2) * ... * f(N)**
**→ Not unique for non-commutative group!**

## Ⅲ. FORMALIZATION

```
theorem :: GROUP_19:32
  for I be non empty set,
      F be Group-Family of I,
      a,b be Element of product F
  st support(a,F) misses support(b,F)
  holds a * b = b * a;

definition
  let I be non empty set;
  let G be Group;
  let F be DirectSumComponents of G,I;
  attr F is internal means
:: GROUP_19:def 9
   (for i be Element of I holds F.i is Subgroup of G)
  & ex h be Homomorphism of sum F, G
     st h is bijective

     & for x be finite-support Function of I,G
        st x in sum F holds h.x = Product x;
end;
```

This theorem and homomorphism h assures the uniqueness of Product x



## Ⅳ. OTHER EQUIVALENT EXPRESSIONS OF THE INTERNAL DIRECT SUM DECOMPOSITION

```
theorem :: GROUP_19:54
  for I be non empty set,
      G be Group,
      F be Group-Family of I
  holds
   F is internal DirectSumComponents of G,I
  iff
   (for i be Element of I holds F.i is Subgroup of G)
  &(for i,j be Element of I, gi,gj be Element of G
    st i <> j & gi in F.i & gj in F.j
    holds gi * gj = gj * gi)
  &(for y be Element of G
    ex x be finite-support Function of I,G
    st x in sum F & y = Product x)
  &(for x1,x2 be finite-support Function of I,G
    st x1 in sum F & x2 in sum F & Product x1 = Product x2
    holds x1 = x2);
```

1. $F_i$ is a subgroup of G
2. Elements of $F_i, F_j$ are commutative
3. $h: \oplus F_i \to G$
   $((x_i) \mapsto \prod_i x_i)$ is isomorphism



## Ⅳ. OTHER EQUIVALENT EXPRESSIONS OF THE INTERNAL DIRECT SUM DECOMPOSITION

```
theorem :: GROUP_20:16
  for I be non empty set,
      G be strict Group,
      F be Group-Family of I
  holds
   F is internal DirectSumComponents of G,I
  iff
      (for i be Element of I holds F.i is normal Subgroup of G)
   & (ex UF be Subset of G
       st UF = Union Carrier F & gr(UF) = G)
   & for i be Element of I holds
       ex UFi be Subset of G
       st UFi = Union((Carrier F) | (I ¥ {i}))
        & [#](gr(UFi)) /¥ [#](F.i) = {1_G};
```

1. $\forall i \in I,\ H_i \lhd G$
2. $< H_i >_{i \in I} = G$
3. $\forall i \in I,\ H_i \cap < H_j >_{j \neq i} = \{e\}$



## Ⅴ. EQUIVALENCE BETWEEN INTERNAL AND EXTERNAL DIRECT SUM DECOMPOSITION

```
theorem :: GROUP_20:20
  for I be non empty set,
      G be Group,
      M be DirectSumComponents of G,I
  holds
  ex f be Homomorphism of sum M,G,
     N be internal DirectSumComponents of G,I
  st f is bijective
   & for i be Element of I holds
     ex qi be Homomorphism of M.i,N.i
     st qi = f * 1ProdHom(M,i)
      & qi is bijective;
```

$$\begin{tikzcd} \bigoplus_{i \in I} M_i \arrow[r, "f"] & G \\ M_i \arrow[u] \arrow[r, "\text{isomorphism}"'] \arrow[ur, phantom, "\circlearrowright"] & N_i \arrow[u, "\text{embedded}"'] \end{tikzcd}$$

$$G \simeq_{external} \oplus M_i$$
$$\Downarrow$$
$$G \simeq_{internal} \oplus N_i$$

## 3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION (UNDER CONSTRUCTION)

I. Commutative property
II. Associative property
III. Universal property

### 3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION
## I . COMMUTATIVE PROPERTY

- Basic pattern

$$G \simeq G_1 \oplus G_2 \quad \Rightarrow \quad G \simeq G_2 \oplus G_1$$

- Generalized

$G \simeq \oplus_{i \in I} G_i$

$\Rightarrow$ For any p be permutation of I,

$G \simeq \oplus_{i \in I} G_{p(i)}$

### 3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION
## I . COMMUTATIVE PROPERTY

```
theorem ThDS141:
  for G be Group,
      I be non empty set,
      F be DirectSumComponents of G,I,
      w be Permutation of I
  holds
    F * w is DirectSumComponents of G,I;

theorem ThDS142:
  for G be Group,
      I be non empty set,
      F be internal DirectSumComponents of G,I,
      w be Permutation of I
  holds
    F * w is internal DirectSumComponents of G,I;
```

### 3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION
## II . ASSOCIATIVE PROPERTY

- Basic pattern

$$G \simeq (G_1 \oplus G_2) \oplus G_3 \Rightarrow G \simeq G_1 \oplus (G_2 \oplus G_3)$$

- Generalized

$G \simeq \oplus_i F_i,\ F_i \simeq \oplus_j H_{ij} \ \Rightarrow G \simeq \oplus_{ij} H_{ij}$

$G \simeq \oplus_{ij} H_{ij}, F_i \simeq \oplus_j H_{ij} \ \Rightarrow \ G \simeq \oplus_i F_i$

3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION

## Ⅱ. ASSOCIATIVE PROPERTY

```
theorem ThDS162:
 for G  be Group,
    I  be non empty set,
    F  be internal DirectSumComponents of G,I,
    II be non-empty ManySortedSet of I,
    FF be non-empty ManySortedSet of I
 st II is disjoint_valued
  & for i be Element of I holds
   FF.i is internal DirectSumComponents of F.i,II.i
 holds
  union FF is internal DirectSumComponents of G,Union II;
```

$$\boldsymbol{G \simeq \oplus_i F_i,\ F_i \simeq \oplus_j H_{ij}\ \Rightarrow G \simeq \oplus_{ij} H_{ij}}$$

3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION

## Ⅱ. ASSOCIATIVE PROPERTY

```
theorem ThDS172:
 for G  be Group,
    I  be non empty set,
    F  be Group-Family of I,
    II be non-empty ManySortedSet of I,
    FF be non-empty ManySortedSet of I
 st II is disjoint_valued
  & union FF is internal DirectSumComponents of G,Union II
  & for i be Element of I holds
   FF.i is internal DirectSumComponents of F.i,II.i
 holds
  F is internal DirectSumComponents of G,I;
```

$$\boldsymbol{G \simeq \oplus_{ij} H_{ij}, F_i \simeq \oplus_j H_{ij}\ \Rightarrow\ G \simeq \oplus_i F_i}$$

3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION

## Ⅲ. UNIVERSAL PROPERTY

Assume $\forall \lambda \in \Lambda, \exists f_\lambda : G \to G_\lambda$,
then $\exists_1 f : G \to \prod_{\lambda\in\Lambda} G_\lambda\ \ s.t.\ f_\lambda = \pi_\lambda \circ f$.

$$\begin{tikzcd}
\prod_{\lambda\in\Lambda} G_\lambda \arrow[r, "\pi_\lambda\ :\ \text{surjective}"] & G_\lambda \\
G \arrow[u, "f"] \arrow[ur, "f_\lambda"'] \arrow[ur, phantom, shift left=3ex, "\circlearrowright"] &
\end{tikzcd}$$

3. BASIC PROPERTIES OF DIRECT SUM DECOMPOSITION

## Ⅲ. UNIVERSAL PROPERTY

Assume $\forall \lambda \in \Lambda, \exists f_\lambda : G_\lambda \to G$,
then $\exists_1 f : \oplus_{\lambda\in\Lambda} G_\lambda \to G\ \ s.t.\ f_\lambda = f \circ i_\lambda$.

$$\begin{tikzcd}
\bigoplus_{\lambda\in\Lambda} G_\lambda \arrow[d, "f"'] & G_\lambda \arrow[l, "i_\lambda\ :\ \text{injective}"'] \arrow[dl, "f_\lambda"] \arrow[dl, phantom, shift left=3ex, "\circlearrowright"] \\
G &
\end{tikzcd}$$

## 4. CONCLUSION

- Accomplished (5,200 lines)
  - Definition of the direct sum decomposition of groups
  - A few equivalent expressions of the direct sum decomposition
- Under Construction
  - Basic properties of direct sum decomposition
- Future Plan
  - Apply to the fundamental theorem of finite abelian groups
  - Extend to modules

## ACKNOWLEDGMENT


- This research is supported by KAKENHI 22300285.


## REFERENCES


1. Rotman, J. J.: An introduction to the theory of groups. Vol. 148. Springer. 40-42 (1995)
2. Robinson, D: A Course in the Theory of Groups. Vol. 80. Springer. 159-191 (1996)
3. Grabowski, A., Kornilowicz, A., Naumowicz, A.: Mizar in a Nutshell. Journal of Formalized Reasoning 3(2), 153-245 (2010)
4. Wiedijk, F.: Writing a Mizar article in nine easy steps. http://mizar.org/project/mizman.pdf
5. Bandini, S., Federici, M. L., Vizzari, G.: An Overview of the Mizar Project. Cybernetics and Systems: An International Journal. 38(7), 729-753 (2007)
6. Wojciech A. Trybulec.: Groups. Formalized Mathematics. 1(5), 821-827 (1990)
7. Kornilowicz, A.: The Product of the Families of the Groups. Formalized Mathematics. 7(1), 125-132 (1998)
8. Okazaki, H., Arai, K. and Shidama, Y.: Normal Subgroup of Product of Groups. Formalized Mathematics. 19(1), 23-26 (2011)
9. Okazaki, H., Yamazaki, H. and Shidama, Y.: Isomorphisms of Direct Products of Finite Commutative Groups. Formalized Mathematics. 21(1), 65-74 (2013)

# Formalization of Polynomially Bounded Functions in Mizar

Hiroyuki Okazaki *

平成 27 年 1 月 10 日

## 1 Introduction

A security proof of cryptographic systems is very important. We are trying to formalize many topics about cryptology, such as computational complexity, finite probability[4, 5, 6, 7, 8, 9], algorithms[10], number theory[11, 12], etc. In this report, we introduce our formalization of the polynomially bounded sequences and prove the correctness of the formalization using the Mizar proof checking system as a formal verification tool[2]. Polynomially bounded sequences play an important role in practical computer complexity theory, such as, computer simulations and cryptology, and is very important in proving the security of cryptographic systems. A security proof is performed by proving that an attack on a target cryptosystem is more difficult than a complexity problem. The class P is a fundamental computational complexity class that contains all polynomial-time decision problems[1]. It takes polynomially bounded amount of computation time to solve polynomial-time decision problems by a deterministic Turing machine. Moreover we formalize polynomial sequences and Negligible Functions.

## 2 Preparation

### 2.1 Mizar

Mizar[2], which formalizes mathematics, is an advanced project of the Mizar Society led by Andrzej Trybulec. The Mizar project was developed to describe mathematical proofs formally in the Mizar language. The

*Institute of Engineering, School of Science and Technology, Shinshu University, 4–17–1 Wsakasato, Nagano–City, Nagano 380–8553 Japan, okazaki@cs.shinshu-u.ac.jp

Mizar proof checker operates in both Windows and UNIX environments and registers proven definitions and theorems in the Mizar Mathematical Library(MML). The objective of this study is to prove the security of cryptographic systems using the Mizar proof checker.

## 2.2 Asymptotic notation

In this section, we briefly review an asymptotic notation $\mathcal{O}$, and we then introduce related formal definitions available in Mizar mathematical library.

**Definition: 1** *Let $f(\cdot)$ and $g(\cdot)$ be functions from $\mathbb{N}$ to $\mathbb{R}$. $g(\cdot) \in \mathcal{O}(f(\cdot))$ iff $\exists N$ be a natural number and $c$ be a real number s.t.C $\forall n s t N \leq n$ holds $0 \leq g(n) \leq c \cdot f(n)$.*

The $\mathcal{O}$ notation is defined in Mizar as follows:

**Definition: 2** `(ASYMPT_0:def 9)`

```
definition
let f be eventually-nonnegative Real_Sequence;
 func Big_Oh(f) -> FUNCTION_DOMAIN of NAT,REAL
 equals
{ t where t is Element of Funcs(NAT, REAL)
  : ex c,N st c > 0 &
  for n st n >= N holds t.n <= c*f.n &
  t.n >= 0};
end;
```

In Mizar, Real_Sequence ia an alias for a function from $\mathbb{N}$ to $\mathbb{R}$C“eventually-nonnegative” is an attribute for Real_Sequence defined as follows:

**Definition: 3** `(ASYMPT_0:def 2)`

```
definition
  let f be Real_Sequence;
  attr f is eventually-nonnegative means
  ex N being Nat st for n being Nat
  st n >= N holds f.n >= 0;
end;
```

The sequence of monomial $a^n$ is defined in Mizar as follows:

**Definition: 4** `(ASYMPT_1:def 3)`

```
definition
  let a be Real;
```

```
  func seq_n^(a) -> Real_Sequence means
  it.0 = 0 &
  for n st n > 0 holds
    it.n = n to_power a;
end;
```

The sequence of monomial $a^{b \cdot n+c}$ ($a, b, c$ are given constant numbers) is defined in Mizar as follows:

**Definition: 5** `(ASYMPT_1:def 1)`

```
definition
  let a,b,c be Real;
  func seq_a^(a,b,c) -> Real_Sequence means
  it.n = a to_power (b*n+c);
end;
```

Note that for a given real number $x$, $x^n$ is represented as "seq_a^(x,1,0)" in Mizar.

## 2.3 Polynomially-bounded Functions

In this section, we briefly review polynomially-bounded functions.

**Definition: 6** *A real valued function $f(\cdot)$ is polynomially-bounded if $\exists n$ be an nutural numbet such that*

$$f(x) \in \mathcal{O}\left(x^n\right)$$

## 2.4 Negligible Functions

In this section, we briefly review negligible functions[3]D

**Definition: 7** *Let $\mu(\cdot)$ be a function from $\mathbb{N}$ to $\mathbb{R}$. $\mu(\cdot)$ is* (negligible function) *iff $\exists N$ be a natural number s.t.C $\forall n$ be a nutural number st $N \leq n$ holds $\forall p(\cdot)$ be a polynomial holds*

$$\mu(n) < \frac{1}{|p(n)|}$$

# 3 Formalization of Polynomially Bounded Functions and Polynomial Functions

In this section, we introduce our formal definition of polynomially bounded functions and polynomial functions.

## 3.1 Formalization of Polynomially Bounded Functions

**Definition: 8**

```
definition
  let p be Real_Sequence;
  attr p is polynomially-bounded means
  ex k be Element of NAT st
    p in Big_Oh(seq_n^(k));
end;
```

We then introduce some theorems about polynomially bounded functions.

**Theorem: 1**

```
theorem
  for a be Element of NAT st 1 < a holds
  seq_a^(a,1,0) is non polynomially-bounded;
```

This theorem shows that exponential $f(x) = a^x$ is not polynomially-bounded if $1 < a$.

**Theorem: 2**

```
theorem
for a,b be Element of INT st b <> 0 holds
  ex A,B be sequence of NAT,
    C be Real_Sequence,
     n be Element of NAT st
  A.0 = |.a.| & B.0 = |.b.| &
  (for i be Nat holds
  A.(i+1) = B.i &
  B.(i+1) = A.i mod B.i) &
  n = (min*{i where i is Nat: B.i = 0} ) &
a  gcd b  = A.n
   & n <= C.(|. b .|)
   &  C is polynomially-bounded;
```

This theorem shows that Euclidean division algorithm calculates the GCD of two integers in polynomially-bounded repetitions.

## 3.2 Formalization of Polynomial Functions

**Definition: 9**

```
definition
```

```
  let c be XFinSequence of REAL;
  func seq_p(c) -> Real_Sequence
  means
  for x be Element of NAT holds
  it.x = Sum(c (#) seq_a^(x,1,0));
end;
```

We then introduce a theorem about polynomial functions.

**Theorem: 3**

```
theorem
  for k be Nat, c be XFinSequence of REAL
  st len c = k+1 & 0 < c.k
  holds seq_p(c) in Big_Oh( seq_n^(k) );
```

This theorem shows that any polynomial function whose most significant coefficient is positive is polynomially bounded.

# 4 Formalization of Negligible Functions

In this section, we introduce our formal definition of negligible functions.

**Definition: 10**

```
definition
let f be Function of NAT,REAL;
attr f is negligible
 means@for c be non empty
  positive-yielding XFinSequence of REAL  holds
 ex  N be Element of NAT st
  for x be Element of NAT
   st N <=x holds |. f.x .| < 1/((seq_p(c)).x) ;
end;
```

We then introduce some theorems about negligible functions.

**Theorem: 4**

```
theorem
for  f be Function of NAT,REAL
  st for x be Element of NAT holds
@@f.x = 1/ (2 to_power x)@holds
  f is negligible;
```

This theorem shows that $2^{-x}$ is a negligible function.

**Theorem: 5**

```
theorem
for  f,g be Function of NAT,REAL
  st f is negligible & g is negligible
    holds  f+g is negligible;
theorem
for  f be Function of NAT,REAL,
     a be Real
  st f is negligible
    holds  a(#)f is negligible;
theorem
for  f,g,h be Function of NAT,REAL
  st f is negligible & g is negligible &
   h =f(#)g
    holds  h is negligible;
```

These theorems show that negligible functions are closed with respect to addition, scalar multiplication and multiplication.

# 5 Conclusion

In this report, we introduce our formalization of polynomially-bounded functions, polynomial functions , and negligible functions in Mizar. We then show some rerated theorems about them. The definitions and theorems in this study have been verified for correctness using the Mizar proof checker.

# 参考文献


[1] Jon Kleinberg, Eva Tardos, “Algorithm Design”, Addison-Wesley, 2005.

[2] Mizar System: *Available at* http://mizar.org/.

[3] Goldreich, “Foundations of Cryptography : Volume: 1 Basic Tools”, Cambridge University Press, 2001.

[4] H. Okazaki, Y. Futa, Y. Shidama, “Formal definition of probability on finite and discrete sample space for proving security of cryptographic systems using Mizar”, Artificial Intelligence Research, 2(4), pp.37-48, 2013.

[5] H. Okazaki, Y. Shidama, “Random Variables and Product of Probability Spaces”, Formalized Mathematics, 21(1), pp.33-39, 2013.

[6] H. Okazaki, “Posterior Probability on Finite Set”, Formalized Mathematics, 20(4), pp.257-263, 2013.

[7] H. Okazaki, Y. Shidama, “Probability Measure on Discrete Spaces and Algebra of Real Valued Random Variables”, Formalized Mathematics, 18(4), pp.213-217, 2010.

[8] H. Okazaki, “Probability on Finite and Discrete Set and Uniform Distribution” , Formalized Mathematics, 17(2), pp.173-178, 2009.

[9] H. Okazaki, Y. Shidama, “Probability on Finite Set and Real-Valued Random Variables”, Formalized Mathematics, 17(2), pp.129-136, 2009.

[10] H. Okazaki, Y. Aoki, Y. Shidama, “Extended Euclidean Algorithm and CRT Algorithm”, Formalized Mathematics 20(2), pp-175-179, 2012.

[11] Y. Futa, D. Mizushima, H. Okazaki, “ Formalization of Gaussian integers, Gaussian rational numbers, and their algebraic structures with Mizar”, ISITA 2012: pp.591-595, 2012.

[12] Y. Futa, H. Okazaki, Y. Shidama, “Formalization of Definitions and Theorems Related to an Elliptic Curve Over a Finite Prime Field by Using Mizar”, J. Autom. Reasoning 50(2), pp.161-172 (2013).

# Formalization of Polynomially-bounded Functions in Mizar

Hiroyuki Okazaki
Shinhsu Univ.

## motivation

**Our Aim:**

**Subset of QED Project for cryptographers**

- Proving security of cryptographic protocols
- Formalizing and evaluating cryptographic primitives
- Formalizing performance evaluation of cryptographic algorithms

## motivation

- Proving the security of cryptographic systems by using Mizar
  - We want to achieve formal proof in a similar way cryptographers do,
  - Not on a specific model.

## Previous research

- automated proof checking
  - CertiCrypt (on Coq)
  - BPW(on Isabelle/HOL)
  - EasyCrypt（on Coq)

Our Aim
Add "Mizar" in this list!

- automated theorem proving
  - SCYTHER
  - Proverif
  - CryptoVerif
  - AVISPA

## motivation

**Our Aim:**

**Subset of QED Project for cryptographers**

- Proving security of cryptographic protocols
- Formalizing and evaluating cryptographic primitives
- Formalizing performance evaluation of cryptographic algorithms

## motivation

- Proving the security of cryptographic systems by using Mizar
  - We want to achieve formal proof in a similar way cryptographers do,
  - Not on a specific model.

Game-sequence proof

## motivation

**Our Aim:**

**Subset of QED Project for cryptographers**

- Proving security of cryptographic protocols
- Formalizing and evaluating cryptographic primitives
- Formalizing performance evaluation of cryptographic algorithms

## Related Topics we must formalize for cryptology

- Probability
- Computational Complexity
- Algorithms
- Number Theory
- Information Theory

etc.

## Computational Complexity

- Class P easy
- Class NP hard

## Class P

- Problem X is in class P

if it takes **polynomially bounded** computation time to solve problem X

- There are feasible algorithms to solve X efficiently

if problem X in P

## Polynomially-bounded Functions

$$f(x) \quad is \quad polynomially-boubded$$

$$iff$$

$$^{\exists}n \in \mathbf{N} \; s.t. \quad f(x) \in O(x^n)$$

## Asymptotic notation $O(*)$

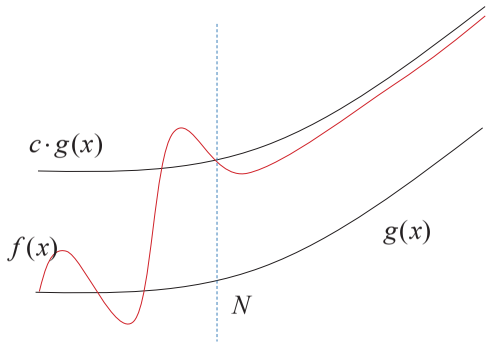
$$f \in O(g)$$

$$iff$$

$$\left( {}^{\exists}c, N \; s.t. \; 0 < c \; \& \right.$$

$$\left. {}^{\forall}x \; s.t. \; N \le x \; holds \; f(x) \le c \cdot g(x) \right)$$

## Asymptotic notation $O(*)$




## Related article in MML
## Asymptotic notation $O(*)$

```
definition
 let f be eventually-nonnegative Real_Sequence;
 func Big_Oh(f) -> FUNCTION_DOMAIN of NAT,
   REAL equals
:: ASYMPT_0:def 9
 { t where t is Element of Funcs(NAT, REAL) :
   ex c,N st c > 0 & for n st n >= N holds t.n <= c*f.n
   & t.n >= 0 };
end;
```

## Related article in MML
## Monomial sequence

$$0, 1^a, 2^a, \ldots, n^a, \ldots$$

```
definition
 let a be Real;
 func seq_n^(a) -> Real_Sequence means
:: ASYMPT_1:def 3
 it.0 = 0 & for n st n > 0 holds it.n = n to_power a;
end;
```

## Polynomially-bounded Functions in Mizar

```
definition
 let p be Real_Sequence;
 attr p is polynomially-bounded means
:: ASYMPT_2:def 1
 ex k be Element of NAT st p in
   Big_Oh(seq_n^(k));
end;
```

## Theorem
## $2^n$ is non polynomially-bounded

$$\forall x \in \mathbf{N}\ s.t.\ 1 < x\ holds$$
$$\neg\left(\exists c, N \in \mathbf{N}\ s.t.\ \forall n \in \mathbf{N}\ s.t.\ N \le n\ holds\ 2^n \le c \cdot n^x\ \right)$$

```
theorem :: ASYMPT_2:18
  for x be Element of NAT st 1 < x holds
  not ex N,c be Element of NAT st
  for n be Element of NAT st N <= n holds
  2 to_power n <= c * ( n to_power x);
```

## Application to computational complexity

```
Theorem :: Euclidean division is P
for a,b be Element of INT st b <> 0 holds
  ex A,B be sequence of NAT,
    C be Real_Sequence,
     n be Element of NAT st
  A.0 = |.a.| & B.0 = |.b.| &
  (for i be Nat holds
  A.(i+1) = B.i &
  B.(i+1) = A.i mod B.i) &
  n = (min*{i where i is Nat: B.i = 0} ) &
A gcd b  = A.n  &
n <= C.(|. b .|) & C is polynomially-bounded;
```

## Related article in MML
## Monomial sequence

$$0, 1^a, 2^a, \ldots, n^a, \ldots$$

```
definition
  let a be Real;
  func seq_n^(a) -> Real_Sequence means
:: ASYMPT_1:def 3
  it.0 = 0 & for n st n > 0 holds it.n = n to_power a;
end;
```

## Univariate Polynomial sequence

```
definition
  let c be XFinSequence of REAL;
  func seq_p(c) -> Real_Sequence
  means
:: ASYMPT_2:def 2
  for x be Element of NAT holds
  it.x = Sum(c (#) seq_a^(x,1,0));
end;
```

## EXAMPLE

$$f(x) = 5 + 4x^1 + 3x^2$$

```
c =<5,4,3>;    (c.0=5,c.1=4,c.2=3);
  c (#) seq_a^(x,1,0)
   =<(c.0)*x^0, (c.1)*x^1, (c.2)*x^2>
   =<5*x^0, 4*x^1, 3*x^2>
(seq_p(c)).x= 5+4*x^1+ 3*x^2;
```

## Polynomial is Polynomially-bounded

```
theorem :: ASYMPT_2:54
  for k be Nat,
  c be XFinSequence of REAL
  st len c = k+1 & 0 < c.k
  holds seq_p(c) in Big_Oh( seq_n^(k) );
```

## Negligible Functions

ある$\mathbf{N} \to \mathbf{R}$である関数$\mu(\cdot)$について
任意の多項式$p(\cdot)$に対して、
ある自然数$N$が存在し、
$N \le n$なる任意の自然数$n$について

$$\mu(n) < \frac{1}{|p(n)|}$$

であるとき$\mu(\cdot)$は無視できるほど小さい関数である

## Negligible Functions

```
definition
let f be Function of NAT,REAL;
attr f is negligible
 means
:defneg:
  for c be non empty  positive-yielding XFinSequence of REAL
   holds
   ex  N be Element of NAT
      st
    for x be Element of NAT
       st N <=x holds |. f.x .| < 1/((seq_p(c)).x) ;
end;
```

## $\frac{1}{2^x}$ is negligible

theorem

for f be Function of NAT,REAL st

for x be Element of NAT holds

f.x = 1/ (2 to_power x)

holds f is negligible

## Future Work using negligibility

- Roughly saying, a cryptosystem is secure if the "probability of attack against the cryptosystem succeeds" is negligible.
- Game sequence proof is based on "indistinguishability" , and "indistinguishability" is defined using negligibility

## Semantic Secure

- Secret key s
- dec(enc(m,s),s)= m
- Choice message m = 0 or 1
- C = enc(m,s)
- Adversary guesses the message is X
- |Prob(X=m) – Prob((X=m)|C)| is negligible

then {enc,dec} is Semantic Secure

## Indistinguishability

- Functions f and g

  are indistinguishable if

  |f-g| is negligible function

EXAMPLE

Pseudo-random number sequence is defined as a sequence indistinguishable from true-random number sequence

## Game based proof

- Design a protocol represented using ideal functionality : GAME0
- Replace Ideal functionality into Feasible algorithm :GAME 1
- Show GAME0 and GAME1 are indistinguishable

# Thank you for your listening

# Coq を使った デジタルデータ放送における ストリームデータ処理の形式化と検証

# Formalization and verification of stream data processing for datacasting in Coq

今井宜洋 Yoshihiro Imai

有限会社 IT プランニング



## 発表すること

- (有) IT プランニング について
- Coq を使用した案件例 (データ放送)
- 我々が Coq を使う理由



## IT プランニングについて

IT Planning, Inc.

- 有限会社 IT プランニング
- 3 人のエンジニアと 5 人のアルバイト
- http://www.itpl.co.jp/



## IT プランニングについて

- 独立系ソフトウェアベンダ
- 仕様言語: OCaml, Scala, Haskell, F#, Java, etc.

## IT プランニングについて

顧客



## 地上デジタル放送

2012 年、日本では地上デジタル放送へ完全移行した。



## 地上デジタル放送

緊急津波速報



## 地上デジタル放送

緊急地震速報

## 地上デジタル放送

開票速報

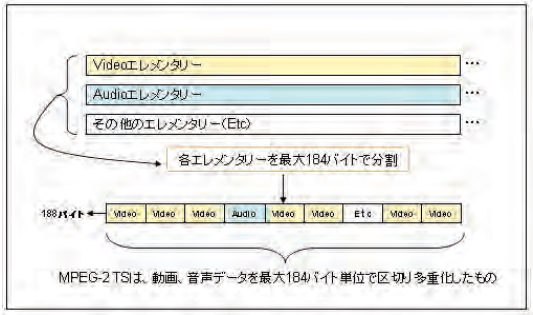


## 地上デジタル放送

マラソン



## Mpeg2-TS






## 受注案件

- 放送データを解析しデジタルデータコンテンツを取り出す
- データコンテンツを検証

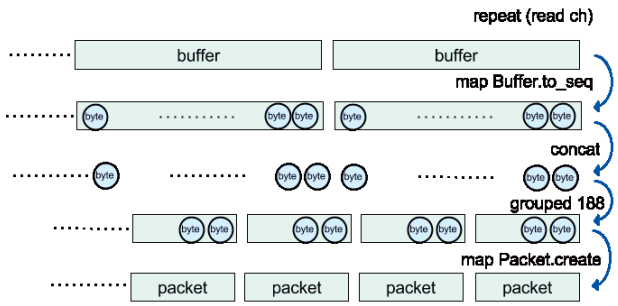
要求

- 高い処理スピード
- データの正確性

OCaml で開発した

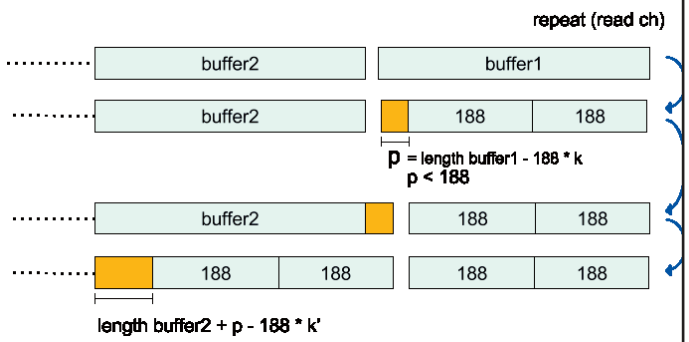
## take_packets

```
take_packets : seq bytes -> seq bytes
```






## take_packets'






```
CoInductive seq (A: Type) : Type :=
 | SNil : seq A
 | SCons : A -> seq A -> seq A.

Variables bytes : Set.
Variables sub : bytes -> nat -> nat ->
  bytes.
```



## 証明した性質(の一部)

```
Goal forall bufs: seq bytes,
  take_packets bufs == take_packets' bufs.

Goal forall buffers: seq bytes,
  ForAll
    (fun packet => length packet = 188)
    (take_packets' buffers).

Goal forall bufs: seq bytes,
  concat (take_packets' bufs) ==
    concat bufs.
```

## 証明

- 定義をあわせて数 100 行程度
- テストよりも楽だったかも



## 良かったこと

- 証明したことについては疑う必要がない前提でより複雑な部分の開発に望めた
- 全体のうち関数一つだけを抜き出して検証ができた



## Coq を使う理由

我々の考え方

- 信頼 → 営業活動
- ソフトウェアの品質 → 信頼

ここでいう品質とは、お客さんの前で致命的なエラーが起きない、実行時エラーが起きないということ



## Coq を使う理由

品質のために我々は静的型付き関数型言語を使っている

- null がない
- 型安全
- ユニットテストが書きやすい

## Coqの良い所

- 強力な仕様記述能力
- 証明済みのプログラムが得られる (Extraction)
- 自動証明 (tactic)



## 証明に向いている検証項目

”forall x: A, ...”

- ”A”はリスト型、関数型など無限の要素を持つ型

典型的な霊

- forall x, f x = g x
- forall x, g (f x) = x
- forall x, P x → P (f x)



## まとめ

- 民間企業では信頼関係が仕事を生む
- 納品物の品質が信頼関係を構築する
- Coqがもたらす品質はテストでは得られない

# The Application of VDM (Vienna Development Method) to the Industrial Development of Firmware for a Smart Card IC Chip

Taro.Kurita@jp.sony.com

December 4, 2014

Abstract

We applied the formal specification language VDM++ in the development of firmware of the "Mobile FeliCa" IC chip. As an outcome, we have achieved successful results and confirmed its effectiveness.

The objectives of applying a formal method were as follows: (1) Description of rigorous specifications; (2) Development and application of a scheme and processes for specification development, firmware implementation and testing; (3) Enhancing the quality of deliverables at the upper stream of development process; (4) Testing thoroughly with formal specifications for whole software development processes; (5) Improvement of communication between engineers.

## 1 Outline of Project

"FeliCa" is a contactless IC card technology widely used in Japan, developed and promoted by Sony Corporation. This FeliCa technology is utilized in the Mobile FeliCa IC chip which is embedded in a mobile phone.

Mobile phones embedded with the FeliCa IC chip are known as "Osaifu Keitai" (means of mobile wallet) by NTT DoCoMo, Inc., and today those chips are embedded in over 50 million mobile phones which can be used as electric money, train tickets, identifications, door keys and so on.

The mobile FeliCa system is comprised of mobile phones with the FeliCa IC chip, FeliCa servers connected to the mobile telecom network and FeliCa reader/writers.

The mobile FeliCa system is shown in Figure 1.

The characteristics of the Mobile FeliCa IC chip firmware are as follows:

- Contains the secure file system and the communications protocol, which are the basis of the FeliCa technology;
- Possesses firewall functions that enable the multiple services in the Mobile FeliCa IC chip such as electric money and train tickets.

We must ensure the extremely high quality of the software in order to avoid serious problems related to social infrastructure, and so the stakeholders will not be affected. It would be impossible to recall all the FeliCa IC chips which are embedded in mobile phones.

The project duration was three years and three months. There were 50-60 members within this project, and the average age was about 30 years old. There were no members who had the knowledge of or the experience with formal methods at the time of project launch.

We have employed several chip manufacturers in order to reduce risks in manufacturing and sales. It was necessary that the firmware on the different ASICs and firmware development environments behave exactly the same so that the compatibility was retained.

C/C++ and assembler languages are used in the implementation of the Mobile FeliCa IC chip firmware.

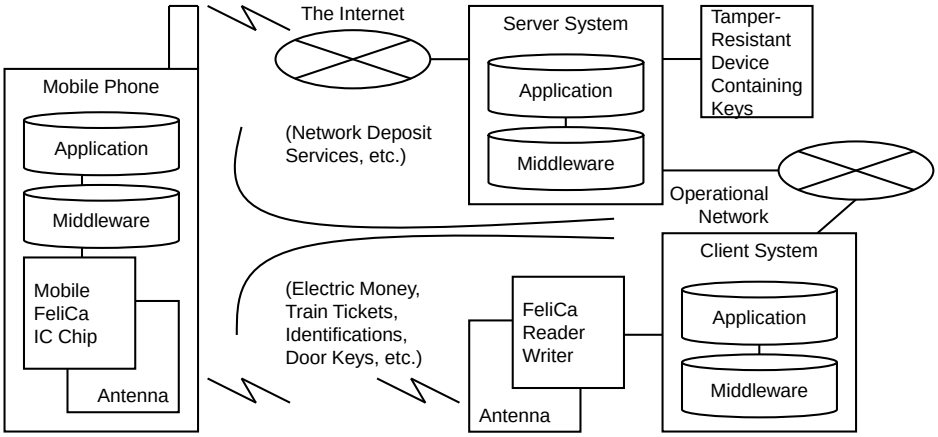



Fig. 1 Mobile FeliCa System

## 2 Objectives

After a consideration to the characterization of the development of the Mobile FeliCa IC chip, we decided to focus on improvements at the upper stream of development process related to software development and mutual understanding between engineers, and we have taken on the challenge of describing formal specifications.

The objectives of applying a formal method were as follows:

1. Description of rigorous specifications and defining functions;
2. Development and application of a scheme and processes for specification development, firmware implementation and testing;
3. Enhancing the quality of deliverables at the upper stream of development process;
4. Testing thoroughly with formal specifications for whole software development processes;
5. Improvement of communications between engineers.

## 3 Approach

We have decided to use the formal specification language VDM++[1] and VDMTools [3], since they support describing and executing large-scale models.

The obtained results of the overall development process are shown in Figure 2. We have developed and tested external specifications using VDM++.

The process of specification development is as follows:

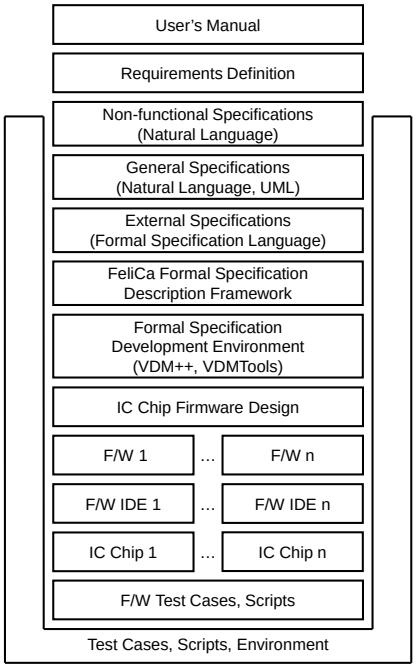



Fig. 2 Obtained Results

1. Discussed requirements with stakeholders and wrote the general specifications in a natural language with various diagrams based on UML notation, such as state transition diagrams and sequence diagrams;
2. Modeled the FeliCa file system and designed and implemented a framework in VDM++;
3. Described command and security specifications using the framework;
4. Tested specifications using a unit testing framework.

The main components or functions of the formal specifications are as follows:

- The FeliCa file system specification that defines the basic data structure;
- The framework for describing and testing specifications that are based on the basic data structure;
- Command specifications which are the basis of the FeliCa technology;
- Security specifications.

Non-functional specifications such as performance and reliability were written in the natural language separately from the formal specifications.

VDM++ is a multi-purpose formal specification language that is primarily an object-oriented extension of VDM-SL [2] which is a formal specification language standardized under the International Organization for Standardization (ISO).

VDMTools is an integrated tool that supports model analysis, specification description and testing. We have used following functions; (1) syntax checking, (2) type checking, (3) sequential implementation of executable specification and debugging support, (4) measurement of the code coverage of executable specification.

VDMTools has the conversion function of executable specification to C++ or Java code, however because the generated code was not suitable for the embedded development environment for secure IC cards, this function was not used.

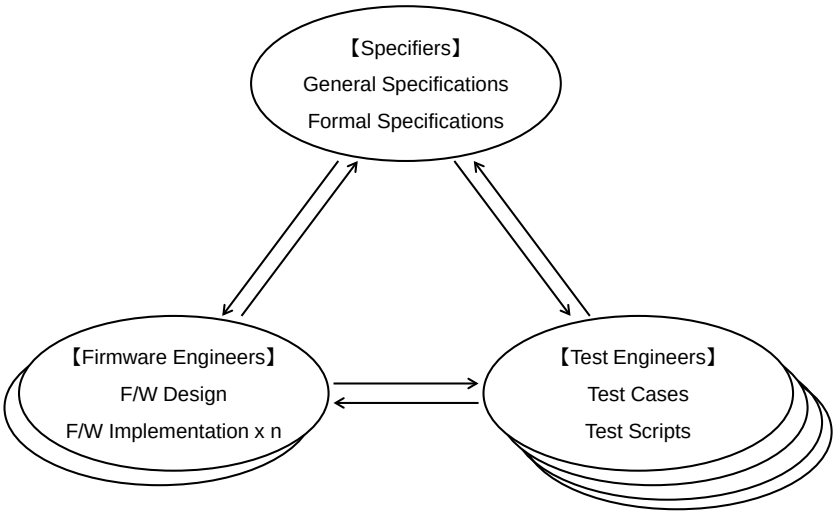



Fig. 3 Development Teams

In the project, we organized three teams; specification team, firmware implementation team and testing team. There were 5-20 members, 15-20 members and 25-35 members respectively.

The outline of the development teams is shown in Figure 3.

The outline of the overall development process is shown in Figure 4.

Considering a single iteration starts from describing specifications to testing the firmware on development board, one iteration consumes a few weeks, and in total 30 iterations were carried out.

## 4 Test Scheme

Test scheme for the overall development is shown in Figure 5.

From the formal specifications, test engineers designed black-box test specifications and then implemented test scripts. Executable formal specifications, firmware on development board and IC chips were automatically tested using the test environment and the test scripts.

From the results of the tests, we were able to confirm whether the test cases and scripts were consistent with the specifications. In addition, from measuring the coverage of the executable formal specifications, we have confirmed that the test cases covered all the defined specifications.

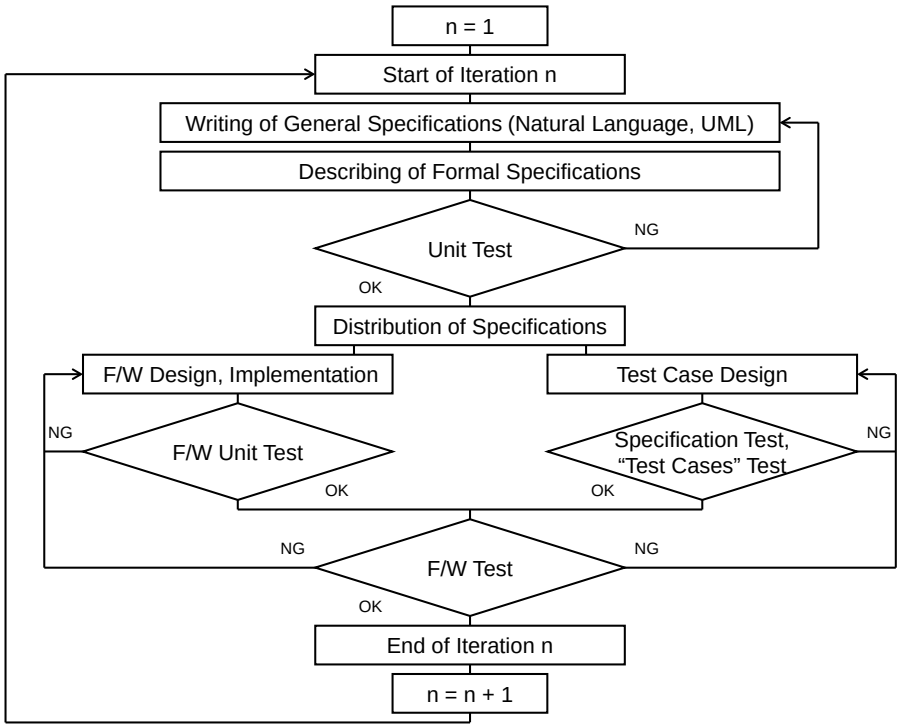



Fig. 4 Development Process

# 5 Results and Discussion

## 5.1 Specification Development

At first, we discussed requirements with stakeholders and wrote the general specification in a natural language with various diagrams based on UML notation, such as state transition diagrams and sequence diagrams. Secondly, we modeled the FeliCa file system and, designed and implemented the framework. Finally, we described command specifications and security specifications on framework.

Furthermore, we developed class methods for unit testing of formal specification, and tested executable formal specifications.

The results related to specifications are as follows:

- 383 pages of a protocol manual written in the natural language (user's manual for other departments within the company and for outside customers);
- 677 pages of an external specification document written in the formal specification language.

Our formal specifications are about 100,000 steps including test cases (about 60,000 steps) and comments written in the natural language. Using this specifications, we implemented the C/C++ code of about 110,000 steps, inclusive of comments, as firmware of a single IC chip.

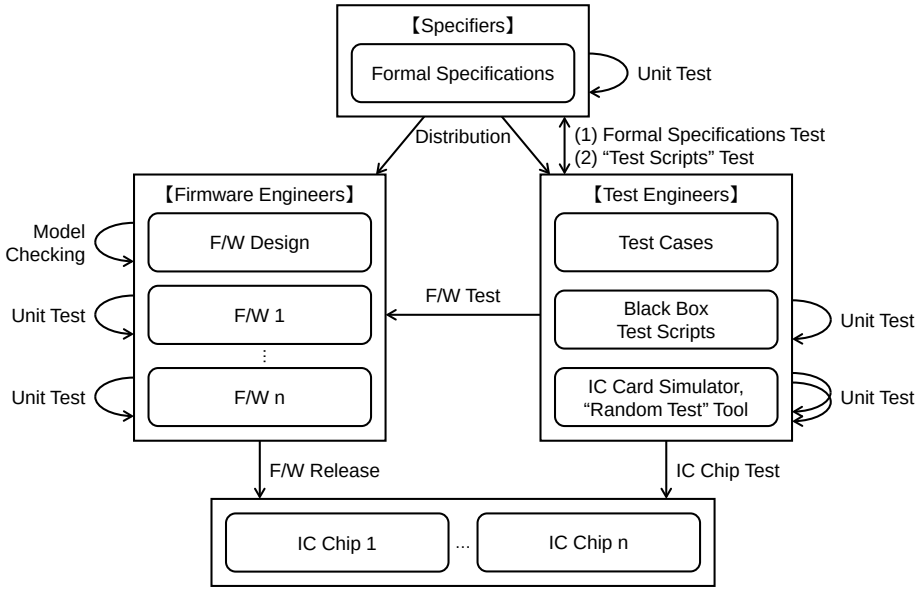



Fig. 5 Test Scheme

Table 1 Percentages of Errors in Firmware Implementation

| Reason for Errors | Percentage |
|---|---|
| Missing description | 0.2% |
| Erroneous description | 0% |
| Unclear description | 1.8% |
| Oversight | 5.6% |
| Insufficient understanding | 10.7% |
| Insufficient confirmation | 0% |
| Failure of change propagation | 0.2% |
| Others (reasons unrelated to specifications) | 81.5% |

## 5.2 Percentages of Errors in Firmware Implementation

The percentages of errors in firmware implementation related to specifications for the overall project are as shown in Table 1.

The formal methods are useful for finding errors in the early stages of development. The formal specification can be syntax and type checked. In addition, since the specification is executable, testing can be carried out.

From the above results, it can be said that we have successfully described the specifications in a precise way. On the other hand, the total percentage of “oversight” errors and “insufficient understanding” errors was 16.3%. This was due to the fact that the separations between the actual specifications and the code required to execute the

Table 2 Number of Changes

| Trigger of Modification | Number | |
|---|---|---|
| Additions and modifications by specifiers themselves | 84 | (46%) |
| Requests, indications and proposals from firmware engineers | 25 | (14%) |
| Requests, indications and proposals from test engineers | 23 | (13%) |
| Others | 9 | ( 4%) |
| Requests and proposals from outside the project | 41 | (23%) |
| Total | 182 | |

Table 3 Number of Errors Discovered before Integration Test

| Phase of Development Process | Number |
|---|---|
| Describing Specifications | 162 |
| Executing and Unit Testing Specifications | 116 |
| Reviewing Specifications | 93 |
| Communicating with Firmware Engineers | 69 |
| Total | 440 |

specifications was unclear.

## 5.3 Efficiency of Specification Development

The average productivity of VDM code for the formal specifications was about 1,900 lines per engineer per month (approximately 160 hours). This number is equal to that for firmware implementation. It can be said that there were no particular disadvantages by using the formal specification language from scratch.

As for the skill level of the engineers, although all has learned the basic information technology, there was a wide gap in the number of years that the engineers had experienced in software development. There were even some members who had no experience.

## 5.4 Changes of Specifications

182 modifications were made to the formal specifications. Specifiers added, modified and debugged 84 times since formal specification version 1.0 was released. The number of changes are as shown in Table 2.

## 5.5 Tests and Reviews of Specifications

The line coverage rate of the formal specifications by unit testing was 82%. We were able to enhance the coverage rate of unit test cases by coverage analysis. As a result of unit testing, we were able, for example, to discover an incorrect path in postconditions. This kind of inconsistency is generally difficult to discover by review.

Discovered formal specification errors before integration tests are as shown in Table 3. The formal method contributes to enhancing the quality of deliverables at the upper stream of development process.

The line coverage rate of the formal specifications by black-box testing and visual inspection was 100%.

“Random Test” is an aging test. The test tool sends random commands continuously to the test target and checks whether the test target sends back correct responses.

By carrying out about 7,000 black-box tests and 100 million random tests, the high quality of IC chips was achieved.

Table 4 Results of Questionnaire

| Responses | Specifiers | Firmware Engineers | Test Engineers | Overall |
|---|---|---|---|---|
| I frequently refer to it | 87% | 13% | 27% | 24% |
| I refer to it | 13% | 27% | 46% | 35% |
| It is easy to read | 29% | 13% | 12% | 12% |
| It is not easy to read, but I can read | 71% | 53% | 67% | 59% |
| It was very good that we applied it | 43% | 20% | 8% | 14% |
| It was good that we applied it | 57% | 7% | 69% | 43% |
| There was no effect observed | 0% | 14% | 4% | 6% |
| A specification language is necessary | 87% | 40% | 81% | 65% |
| I would like to utilize it in the future as well | 0% | 13% | 19% | 18% |
| I would like to utilize it depending on the case | 100% | 40% | 69% | 59% |

### 5.6 Formal Specifications and Communication

We have analyzed all the questions related to the specifications from firmware engineers, test engineers and stakeholders. We have divided questions into three categories: “Comprehension”, “Intent” and “Error”.

As compared with the formal specifications, there are more requests for clarification on the general specifications and the manuals written in the natural language. On the other hand, there are more questions related to background comprehension of the formal specifications.

This result shows that the specifications in the natural language were not precise. For the formal language, the backgrounds of the specifications were unclear to readers. Therefore, it is preferable that the background of the specifications written in the natural language are included as comments in the formal specifications.

### 5.7 Results of Questionnaires and Interviews with Project Members

A questionnaire survey was conducted towards all members of this project. At the same time, an approximately 40 minutes interview was held individually to 12 members who wrote and tested the formal specifications or referred to them frequently.

A portion of the results of the questionnaire is summarized in Table 4.

In briefly summarizing the results of the above questionnaire and interviews, we have found that there was no major adverse reaction to the application of the formal method. A commonly found opinion was that while they realized its efficiency as a communication tool, the cost for initial training was necessary.

**Specifiers** Generally have a favorable impression of the formal method and are positive about the effect of its application in the project. Within this project, they can confidently take on duties other than specification development.

**Firmware Engineers** While the specification itself is clear, it is hard to identify the boundary of the external specification and the formal specification program required execution.

**Test Engineers** Since the specification is clear, it is possible to test with confidence.

## 6 Conclusion

The application of the formal method was highly effective for the success of our project on schedule.

The formal method contributes to the quality of deliverables at the upper stream of development process.

And, the formal method appear to have encouraged communications between engineers, which is vital for software development.

Additionally, the fact that the executable formal specifications are resembled to program codes is an substan-

tial advantage because the know-how accumulated through program development can be applied (for example, configuration management, filter programs, batch programs, object-oriented analysis and design technology, unit testing and so on).

It is necessary to pay attention to not only executable features but also the readability of specifications. Specifications which are referred to by all project members need to be simple, so that it can be read without stress.

## 7 Difficulties

In our project, the capability for abstraction required by formal specification engineers did not go beyond that required by usual programmers.

In the case of VDM++, coding and testing formal specifications are not difficult for engineers who are familiar with the object-oriented design and the implementation of C++/Java languages. They would need only about a month training.

However, it is difficult to describe of the formal specifications than using general programming languages. Although a lack of VDM libraries, templates and advices from experts could be a disadvantage for beginners.

## 8 Future Issues

Future issues are listed below:

- Validating whether specifications fulfill requirements;
- Negotiating with stakeholders who do not read formal specifications;
- Testing user's manuals that is based on formal specifications;
- Defining effective combinations of formal and informal specifications;
- Description of formal specifications suitable for embedded systems;
- Validation and testing of the formal specifications; for example validation of whether a security specification is logically consistent;
- Framework for describing specifications that are easy-to-read and executable;
- Specifications that firmware engineers and test engineers feel close to, familiar with and comfortable browsing.

## Acknowledgments

We would like to thank Professor Peter Gorm Larsen of Engineering College of Aarhus, Professor Keijiro Araki of Kyushu University, Shin Sahara of CSK Systems Corporation and Hiroshi Sako of Designers' Den Corporation for their great assistance in the application of the formal method.

## References

[1] J. Fitzgerald, P.G. Larsen, P. Mukherjee, N. Plat, M. Verhoef, *Validated Designs for Object-oriented Systems*, Springer, 2005

[2] J. Fitzgerald, P.G. Larsen, *Modelling Systems: Practical Tools and Techniques in Software Development*, Cambridge University Press, 1998

[3] http://www.vdmtools.jp/en/

FeliCa Networks

SONY

# The Application of VDM to the Industrial Development of Firmware for a Smart Card IC Chip

December 4, 2014

Taro KURITA

Sony Corporation and FeliCa Networks, Inc.

## Agenda

- What is "Mobile FeliCa"
- Development of First Generation ("The Stone Age")
- Development of Second Generation
  - Goal
  - Approach
  - Result
- Summary and Current/Future Issues

FeliCa Networks

- What is "Mobile FeliCa"
- Development of First Generation ("The Stone Age")
- Development of Second Generation
  - Goal
  - Approach
  - Result
- Summary and Current/Future Issues

FeliCa Networks

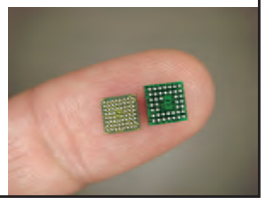
## What is "Mobile FeliCa"

- "FeliCa" is a contactless IC card technology widely used in Japan.
- FeliCa is developed and promoted by Sony Corporation.
- FeliCa uses Near Field Communication (NFC) technology.
- FeliCa is used for electric money, train tickets, identification, door keys and so on.
- Today, "Mobile FeliCa" IC chips were embedded in over 300 million mobile phones.

FeliCa Networks

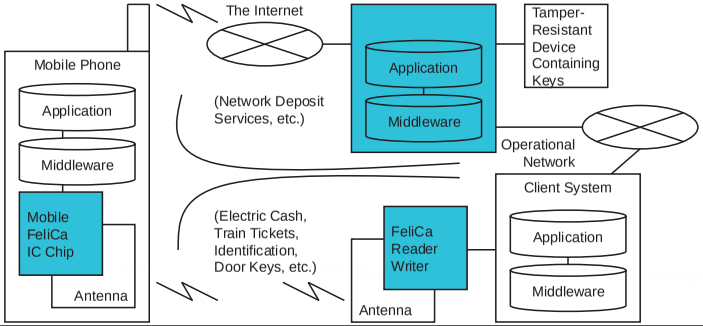
## 2nd Mobile FeliCa IC Chip

- **Hardware Specification:**
  - 32 bit CPU (ARM7 and H8SX, etc.)
  - 12 KByte RAM
  - 240 KByte ROM (for Firmware)
  - 72 KByte E2PROM
  - 4+ Common Criteria Security Level (ISO/IEC 15408)
- **C/C++ and assembler languages are used in the implementation of the Mobile FeliCa IC chip firmware manually.**
- **We have employed several chip manufacturers in order to reduce risk in manufacturing and sales.**



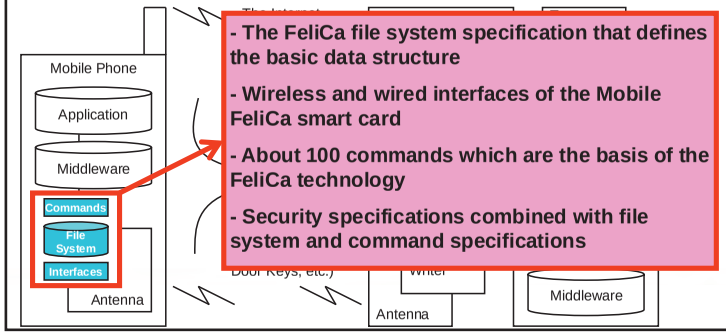
## Mobile FeliCa System

The mobile FeliCa system is comprised of
- mobile phones with a FeliCa IC chip,
- FeliCa servers connected to the mobile telecom network,
- and FeliCa reader/writers.




## Mobile FeliCa IC Chip Firmware (1/2)

The mobile FeliCa system is comprised of
- mobile phones with a FeliCa IC chip,
- FeliCa servers connected to the mobile telecom network,
- and FeliCa reader/writers.




**- The FeliCa file system specification that defines the basic data structure**

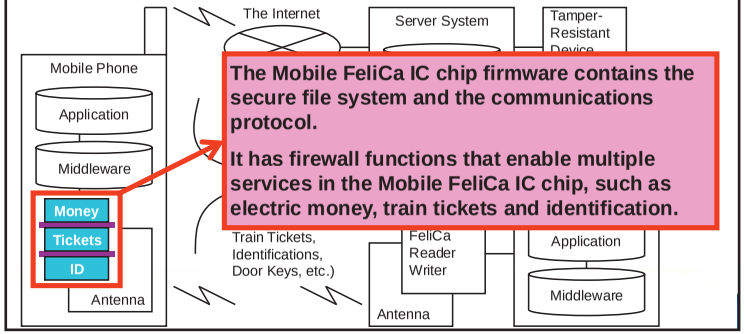
**- Wireless and wired interfaces of the Mobile FeliCa smart card**

**- About 100 commands which are the basis of the FeliCa technology**

**- Security specifications combined with file system and command specifications**

## Mobile FeliCa IC Chip Firmware (2/2)

The mobile FeliCa system is comprised of
- mobile phones with a FeliCa IC chip,
- FeliCa servers connected to the mobile telecom network,
- and FeliCa reader/writers.




**The Mobile FeliCa IC chip firmware contains the secure file system and the communications protocol.**

**It has firewall functions that enable multiple services in the Mobile FeliCa IC chip, such as electric money, train tickets and identification.**

- What is "Mobile FeliCa"
- Development of First Generation ("The Stone Age")
- Development of Second Generation
  - Goal
  - Approach
  - Result
- Summary and Current/Future Issues

FeliCa Networks

## Development of First Generation ("The Stone Age")

- 2001 - 2004
- Death march project with vague specifications
- Then, we decided to study formal methods and describe of rigorous specifications.

FeliCa Networks

- What is "Mobile FeliCa"
- Development of First Generation ("The Stone Age")
- Development of Second Generation
  - Goal
  - Approach
  - Result
- Summary and Current/Future Issues

FeliCa Networks

## Development of Second Generation

- 2004 - 2006 (First Release)
- 2006 - 2010 (Modified Release)
- We have developed the "Mobile FeliCa" IC Chip firmware specifications with a formal method using VDM++ and VDMTools.
- We use "Level 0" Formal Methods. (Level 0: Formal Specification)
- We described and tested formal specifications without proof.
- Thanks to the formal methods, there are no problems related to the software specifications since the first release in 2006.

FeliCa Networks

## Goals for Applying the VDM

- Creating precise specifications
- Enhancing the quality of deliverables at the design phase
- Improving development processes
- Testing completely with different approaches
- Activating communication between engineers

FeliCa Networks

## Project Duration and Members

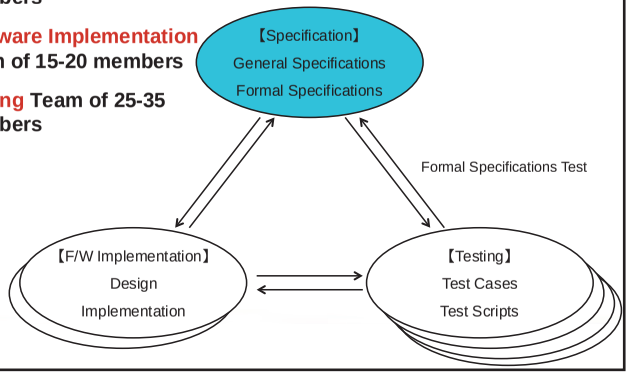
- The project duration was three years and three months. It finished on schedule.
- There were 50-60 members. The average age was about 30 years old.
- There were no members who had knowledge of or experience with formal methods.

FeliCa Networks

## Teams for Overall Development

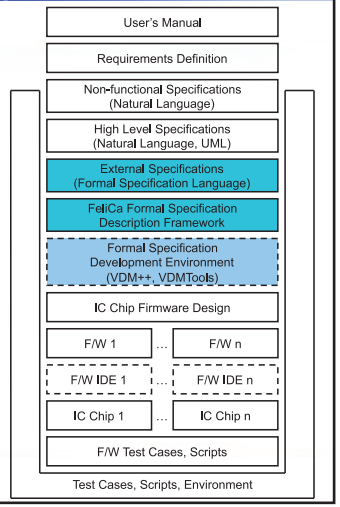
We organized three teams:

- Specification Team of 5-20 members
- Firmware Implementation Team of 15-20 members
- Testing Team of 25-35 members




## Artifacts for Overall Development

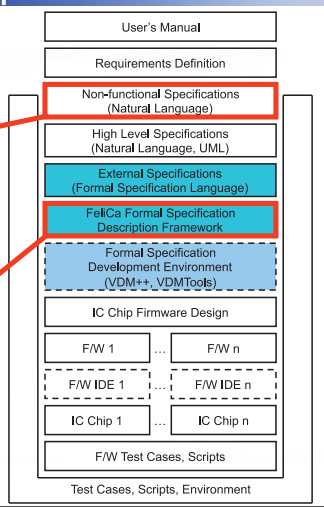
- Developed products are shown with solid lines and tools used in development are shown with dashed lines.
- We described and tested specifications of external behavior using VDM++.
- We developed executable formal specifications.



## Artifacts for Overall Development

Non-functional specifications such as performance and reliability were written in the natural language separately from the formal specifications.

This is the framework for describing and testing specifications that are based on the basic data structure, interfaces and templates.

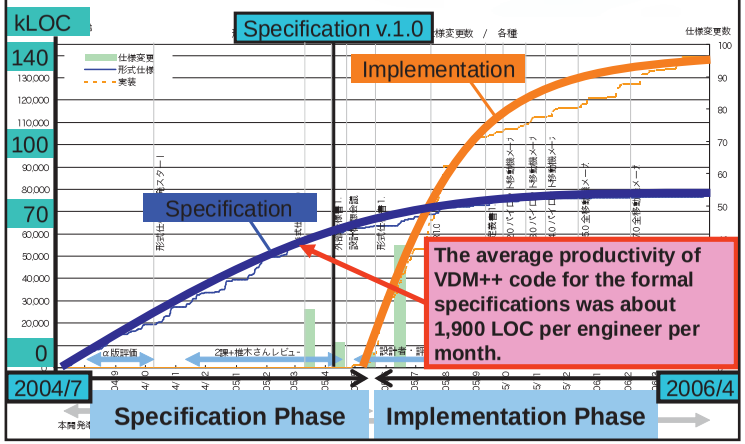



## Results

- 383 pages of a protocol manual written in the natural language
- 677 pages of an external specification document written in the formal specification language
- Our formal specifications are about 100,000 LOC including test cases (about 60,000 LOC) and comments.
- Using this specifications, we implemented the C++ code of about 110,000 LOC, inclusive of comments.

FeliCa Networks

## Specification and Implementation Growth




The average productivity of VDM++ code for the formal specifications was about 1,900 LOC per engineer per month.

## Formal Specification Errors in Specification Phase

| Phase of Development Process | Number |
|---|---|
| Describing Specifications | 162 |
| Executing and Unit Testing Specifications | 116 |
| Reviewing Specifications | 93 |
| Communicating with Firmware Engineers | 69 |
| Total | 440 |

Debug Density = 440/40,000 = about 11 errors/kLOC

The formal method contributes to enhancing the quality of deliverables at the early stage of the development process.

FeliCa Networks

## Errors in Firmware Implementation Phase

| Reason for Errors | Percentage |
|---|---|
| Missing description | 0.2% |
| Erroneous description | 0% |
| Unclear description | 1.8% |
| Oversight | 5.6% |
| Insufficient understanding | 10.7% |
| Insufficient confirmation | 0% |
| Failure of change propagation | 0.2% |
| Others (reasons unrelated to specifications) | 81.5% |



## Errors in Firmware Implementation Phase

| Reason for Errors | Percentage |
|---|---|
| Missing description | 0.2% |
| Erroneous description | 0% |
| Unclear description | 1.8% |
| Oversight | 5.6% |
| Insufficient understanding | 10.7% |
| Insufficient confirmation | 0% |
| Failure of change propagation | 0.2% |
| Others (reasons unrelated to specifications) | 81.5% |

**Errors in Firmware Implementation Related to Description of Specifications**

- **It can be said that we have successfully described the specifications in a precise way.**
- **The formal methods are useful for finding errors in the early stages of development.**



## Errors in Firmware Implementation Phase

| Reason for Errors | Percentage |
|---|---|
| Missing description | 0.2% |
| Erroneous description | 0% |
| Unclear description | 1.8% |
| Oversight | 5.6% |
| Insufficient understanding | 10.7% |
| Insufficient confirmation | 0% |
| Failure of change propagation | 0.2% |
| Others (reasons unrelated to specifications) | 81.5% |

**Errors in Firmware Implementation Related to Misreading**

- **On the other hand, the total percentage of "oversight" errors and "insufficient understanding" errors was 16.3%.**
- **This was due to the fact that the separations between the actual specifications and the code required to execute the specifications was unclear.**

**Future Issue: Easy-to-read**



- **What is "Mobile FeliCa"**
- **Development of First Generation ("The Stone Age")**
- **Development of Second Generation**
  - Goal
  - Approach
  - Result
- **Summary and Current/Future Issues**

## Conclusions

- The application of the level 0 formal method was highly effective for the successful completion of our project on schedule.
- The formal method contributes to the quality of deliverables at the early stage of the development process.
- And, the formal method appears to have activated and encouraged communication between engineers, which is vital for software development.
- It is necessary to pay attention to not only executable features, but also the readability of specifications.
- Specifications which are referred to by all project members need to be simple, so that it can be read without stress.

FeliCa Networks

## Current Issues

- Negotiating with stakeholders who do not use formal methods.
- Translating and testing user's manuals that is based on formal specifications.
- Defining effective combinations of formal and informal specifications.
- Description of formal specifications suitable for embedded systems and code generation.
- Framework for describing specifications that are easy-to-read and executable.
- Specifications that firmware engineers and test engineers feel familiar with and comfortable reading.

FeliCa Networks

# Correct-by-Construction Program Synthesis in Coq

**Adam CHLIPALA**
Massachusetts Institute of Technology, USA

Fiat is a Coq library supporting deriving efficient programs from specifications. We think of it as enabling a new style of modularity in programming, where it is possible to separate functionality from performance, with language-enforced encapsulation keeping the latter from interfering with the former. More concretely, the programmer starts with a program written to be as easy to understand as possible. From here, optimization scripts are applied to do stepwise refinement, gradually replacing nondeterministic constructions with deterministic ones, and replacing slow algorithms with fast ones. As the implementation language of optimization scripts, we reuse Ltac, Coq's language for coding proofs and decision procedures. We inherit Ltac's support for generating proof trails that vouch for the soundness of all operations. With highly expressive functional programming languages for both the functionality and performance parts of programs, we develop new abstraction and modularity patterns on each side of the divide. Our main case study so far deals with specifications that resemble SQL query and update operations. A set of operations over a particular schema are packaged together as an abstract data type, hiding the concrete representation of a relational database. We refine these nondeterministic descriptions of operations into efficient functional programs, which can be extracted to OCaml and run with good performance. In ongoing work, we are extending the refinement process to produce efficient assembly code using imperative data structures (via the Bedrock library), retaining proof trails to relate those programs to the original specifications. We are also exploring several new specification domains with associated refinement strategies, including for parsing, graph algorithms, and stencils in scientific computing.

Joint work with Benjamin Delaware, Clément Pit–Claudel, Jason Gross, and Peng Wang

## References

[1] B. Delaware, C. Pit–Claudel, J. Gross, A. Chlipala. Fiat: Deductive Synthesis of Abstract Data Types in a Proof Assistant. In *Proceedings of POPL'15*, to appear, 2015.

**TPP2014**

Theorem Proving and Provers for Reliable Theory and Implementations

**December 3-5, 2014, Nishijin Plaza, Kyushu University, Fukuoka, Japan**

# Formalization of Error-correcting Codes using SSReflect

**Reynald AFFELDT**

National Institute of Advanced Industrial Science and Technology

(joint work with Jacques Garrigue, Nagoya University)

By adding redundant information to transmitted data, error-correcting codes (ECCs) make it possible to communicate reliably over noisy channels. Minimizing redundancy and coding/decoding time has driven much research, culminating with Low-Density Parity-Check (LDPC) codes. Hard-disk storage, wifi communications, mobile phones, etc.: most modern devices now rely on ECCs and in particular LDPC codes. Yet, correctness guarantees are only provided by research papers of ever-growing complexity. One solution to improve this situation is to provide a formalization of ECCs. A first difficulty to achieve this goal has been lifted by the SSReflect extension [GMT08] of the Coq proof-assistant, that provides in particular a substantial formalization of linear algebra. Using SSReflect, we have been able to formalize the main properties of the celebrated Hamming codes and also the properties of the more difficult LDPC codes.

## 1. A Formal Setting for Linear ECCs

Let us first give a brief overview of the bottom layer of our library for linear ECCs. This layer itself relies on previous work on formalization of information theory [AHS].

We are concerned with linear ECCs that manipulate bit-vectors as data. In SSReflect, bit-vectors are appropriately modeled as row vectors over $\mathbb{F}_2$ and we start by providing their properties in terms of Hamming weight (the number of bits 1) or Hamming distance (the number of bits that are different).

The simplest definition of a linear ECC one can find in textbooks is as a set of bit-vectors (called *codewords*) closed by addition. In practice, a linear ECC is rather defined as the kernel of a matrix called the *parity check matrix* [MS77]. This view of a linear ECC as a set of vectors makes it already possible to prove several properties of linear ECCs such as the characterization of the codewords of a given weight or the minimum distance between any two codewords.

The view of a linear ECC as a set of codewords leaves out the (noisy) channel on which the transmission of codewords occurs, the coding procedure from messages to codewords, and the decoding procedure from channel outputs back to input messages. From a practical standpoint, a linear ECC is rather a pair of coding/decoding functions used in conjunction with a channel modeled as a stochastic matrix. In this setting, it is possible to compare different encoding/decoding schemes according to whether they perform, e.g., minimum distance decoding (decoding decodes to the closest codeword in terms of Hamming distance) or maximum likelihood decoding (decoding decodes to the message that is the most likely to have been encoded according to the channel definition). More precisely, for an encoding function $f$, a maximum likelihood decoding function $\phi$ is such that $W^n(y|f(\phi(y)) = \max_{x\in\mathbb{F}_2^n} W^n(y|f(x))$ where $W^n(y|y_0)$ is the

probability for a channel $W$ that an input $y_0$ of length $n$ is output as $y$. SSReflect's library about big operators came in handy to formalize these notions.

## 2. Formalization of Hamming Codes and Their Properties

Hamming codes are linear ECCs on an alphabet $\Sigma$, where messages of length $|\Sigma|^n - n - 1$ are encoded by codewords of length $|\Sigma|^n - 1$, *i.e.* one adds only $n$ extra bits for error checking. Assuming bit-vectors, the codewords are defined by the parity check matrix whose columns are the binary representations of non-null words of length $n$ (there are $2^n - 1$ such words).

Let $[u]_i$ be the $i^{\text{th}}$ bit of the word representing $u$ in binary notation. Then codewords $y$ satisfy the equations $\sum_{i=1}^{2^n-1}[i]_{j-1}[y]_{i-1} \equiv 0 \pmod 2 \quad (1 \leq j \leq n)$. This provides us with a parity check matrix $H$, from which one can easily construct and encoding matrix $G$ to serve as an encoding function by permuting columns and using properties of block matrices (we of course have the relation $H \cdot G^T = 0$):

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \qquad G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

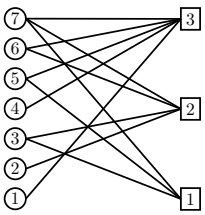
The most interesting property of Hamming codes is that there are no codewords of weights 1 and 2, *i.e.* by linearity the minimum distance between two distinct codewords is 3. A corollary of this property is that, by choosing the closest codeword, one is able to correct 1-bit errors (since there can be only one codeword that close). Another, more abstract property is that the above approach of choosing the word corresponding to the closest codeword, *i.e.* minimum distance decoding, also happens to be a maximum likelihood decoding when the error probability of the channel is less than $\frac{1}{2}$.

We were able to prove those properties in the binary case, for arbitrary size messages, making efficient use of the matrix and finset libraries of SSReflect. While computation on matrices can get tedious due to dependent types, we could avoid many problems by switching to sets of columns rather than concrete matrices where appropriate. Note that a large part of the effort was in finding the correct pre-conditions for each definition, something that is not insisted much on in coding theory textbooks.

## 3. Formalization of the Properties of LDPC Codes

Our main result is the formalization of the correctness property of the sum-product algorithm: a generic decoding algorithm that is particularly efficient for LDPC codes. This algorithm is better explained using *Tanner graphs* as an alternative representation of parity-check matrices.

A Tanner graph is a graph whose vertices correspond to the rows and columns of a parity-check matrix. For example, the figure below represents the Tanner graph corresponding to the Hamming code of the previous section:



Tanner graphs are instrumental in the formalization of the sum-product algorithm because the latter is better expressed as an algorithm that computes locally for each node "messages" that are exchanged with their neighbors.

Let us consider a noisy channel $W$, an output message $y$ we want to decode, and let us denote by $x$ the corresponding input codeword we are looking for. When decoding, we are concerned with finding a way to evaluate $P^W_{n_0}(x_{n_0}|y)$, the probability that the $n_0^{\text{th}}$ bit of $x$ is set knowing the output message $y$. The first result that we formalized is the fact that this probability is proportional to the following quantity:

$$P^W_{n_0}(x_{n_0}|y) \propto W(y_{n_0}|x_{n_0}) \prod_{m_0 \in F(n_0)} \alpha_{m_0,n_0}$$

where $F(n_0)$ is the set of neighbors of the node $n_0$ in the Tanner graph and $\alpha_{m_0,n_0}$ is a "message" (precise definition omitted here) computed using information from the subtree rooted at $m_0$ and $n_0$ (the above property holds when the Tanner graph is acyclic).

The property above provides a way to evaluate $P^W_{n_0}(x_{n_0}|y)$, and therefore to find out whether $x_{n_0}$ is likely to be set or not, but it does not provide an efficient algorithm because $\alpha$ messages cannot be computed locally. The second property that we formalized introduces a second type of messages $\beta$ such that both $\alpha$ and $\beta$ messages can be computed incrementally (starting from the leaves of the Tanner graph when it is acyclic):

$$\alpha_{m_0,n_0} = \sum_{t \# V(m_0) \setminus n_0} \delta(V(m_0), t) \prod_{n_1 \in `V(m_0) \setminus n_0} \beta_{m_0,n_1}$$

where $t\#V$ is a projection of the row-vector $t$ over the set of indices $V$ and $\delta$ is an indicator function (precise definition omitted here). This is the latter relation that justifies the correctness of the sum-product algorithm.

In this presentation, we explain how we formalized the above properties, focusing in particular on the key properties of Tanner graphs and the formalization of the *summary notation* that is an important technical device in the development of modern coding theory in general.

**Acknowledgments**. Taku Asai, Takafumi Saikawa, Kazuhiko Sakaguchi, and Yuto Takahashi made contributions to the Coq formalization. The project of formalization of modern coding theory is a collaboration with Manabu Hagiwara, Kenta Kasai, and Shigeaki Kuzuoka.

## References

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977. 7th impression: 1992.

[GMT08] Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A small scale reflection extension for the Coq system. Technical Report RR-6455, INRIA, 2008. Version 14 (March 2014).

[Hag12] Manabu Hagiwara. *Coding Theory: Mathematics for Digital Communication*. Nippon Hyoron Sha, 2012. `http://www.nippyo.co.jp/book/5977.html`. In Japanese.

[AHS] Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues. Formalization of Shannon's theorems. *J. Autom. Reasoning*, 53(1):63–103. Springer, 2014.

**TPP2014**

Theorem Proving and Provers for Reliable Theory and Implementations

**December 3-5, 2014, Nishijin Plaza, Kyushu University, Fukuoka, Japan**

# Formalization of Matrix Representation of Direction Relations with Application to the Superposition of Rectangles

**Fadoua Ghourabi**

Kwansei Gakuin University, Japan

(joint work with Kazuko Takahashi)

Our perception of the space and the objects around us is basically qualitative. For instance, in most cases we can manage without recourse to the exact positions of objects in a coordinate system. As an attempt to imitate this cognitive aspect, qualitative spatial representation emerged as an area of spatial knowledge representation. The foundation in qualitative spatial representation is to treat objects of the space qualitatively, i.e. what matters is how objects are related. Positional knowledge about objects in the space is one of the relevant problems that is addressed by the field of qualitative spatial representation. The direction relations describe where an object is positioned w.r.t. a reference [Frank, 1991, Clementini et al., 1997]. In this work, we use the direction relations in a smaller scale problem. We examine a practically oriented problem, namely software window allocation, from the qualitative spatial representation point of view.

The window allocation involves superposition under conditions of visibility. Namely, a superfluous information in a window can be hidden and superposed by a relevant information in another window. The spatial object of interest is the rectangular window, called unit. The structure of a unit contains a must-be-visible white region and a might-be-hidden black region. For instance, Figs. 1(a) and 1(b) depict two units that are divided into white and black regions. The superposition of the unit in Fig. 1(b) onto the unit in Fig. 1(a) while keeping all white regions visible have several solutions as shown in Figs. 1(c) - 1(f). We use relative directions to qualitatively represent the positions of the white and black regions inside the unit. To encode the direction relations between the objects, matrices are commonly used data-structure, e.g. the *object interaction matrix* (OIM) [Chen et al., 2010]. By extending the edges of the black and white regions, the plane is divided into tiles of direction relations. The entries of the matrices are computed from the intersections of those tiles and the black and white regions.

We formalize a matrix theory in Isabelle/HOL [Nipkow et al., 2002]. Operation of rotation is defined to establish an equivalence relation between the matrices. Then, the superposition of units is computed from the superposition of their matrix representations. We define properties to answer questions about a superposition, for instance, whether it is successful or whether the result is a valid unit. Checking these properties

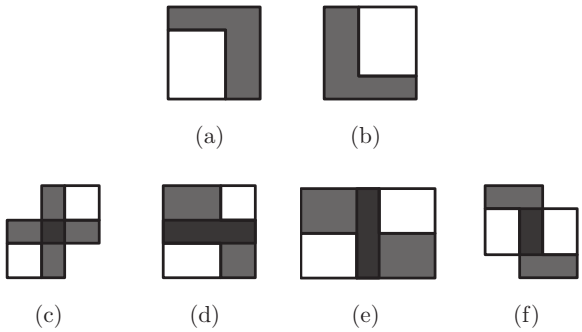



FIGURE 1. Units divided into black and white regions and the solutions of their superposition

formally is a challenge due to the number of the pairwise combinations of rectangles to be superposed. In an attempt to optimize the proofs, we group the results of superposing the rectangles into equivalence classes. Given an equivalence relation $r$ and two functions $f$ and $g$, we introduce the two (infix) predicates "preserves" and "respects" that are defined as follows.

$$f \text{ preserves } r \triangleq \forall x\ y.\ (x, y) \in r \Rightarrow (f\ x, f\ y) \in r$$

$$g \text{ respects } r \triangleq \forall x\ y.\ (x, y) \in r \Rightarrow gx = gy$$

We furthermore prove that the superposition function respects the rotation relation and that the properties of success and validity preserves the rotation relation. We therefore show that it is enough to check a property for a representative element of an equivalence class.

Part of this work will be published in [Ghourabi and Takahashi, 2015]. For a closer look at the proofs, our Isabelle/HOL theory files are available at `http://ist.ksc.kwansei.ac.jp/~ktaka/SuperpositionTheory/`.

## REFERENCES

[Chen et al., 2010] Chen, T., Schneider, M., Viswanathan, G., and Yuan, W. (2010). The Objects Interaction Matrix for Modeling Cardinal Directions in Spatial Databases. In *Database Systems for Advanced Applications*, volume 5981 of *LNCS*, pages 218–232. Springer Berlin Heidelberg.

[Clementini et al., 1997] Clementini, E., Felice, P. D., and Hernándes, D. (1997). Qualitative Representation of Positional Information. *Artificial Intelligence*, 95(2):317 – 356.

[Frank, 1991] Frank, A. U. (1991). Qualitative Spatial Reasoning about Cardinal Directions. In *Proceedings of the International Symposium on Computer-Assisted Cartography*, pages 148–167. ACSM-ASPRS.

[Goyal and Egenhofer, 2001] Goyal, R. and Egenhofer, M. J. (2001). Similarity of Direction Relations. In *Seventh International Symposium on Spatial and Temporal Databases*, volume 2121 of LNCS, pages 36–55. Springer-Verlag.

[Nipkow et al., 2002] Nipkow, T., Paulson, L. C., and Wenzel, M. (2002). *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS Tutorial*. Springer.

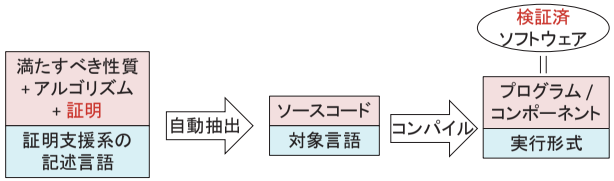
[Ghourabi and Takahashi, 2015] Ghourabi, F. and Takahashi, K. (2015). Formalizing the Qualitative Superposition of Rectangles in Proof Assistant Isabelle/HOL. In *Proceedings of the 7th International Conference on Agents and Artificial Intelligence (ICAART)*. (accepted, to appear).

# CoqからScalaへのコード抽出とその妥当性
Coq code extraction to Scala and its correctness

TPP2014
2014年12月4日

逸見港* 田辺 良則** 今井 宜洋*** 萩谷 昌己*
Henmi, Ko Tanabe, Yoshinori Imai, Yoshihiro Hagiya, Masami
* 東京大学 ** 国立情報学研究所 *** ITプラニング

## コード抽出

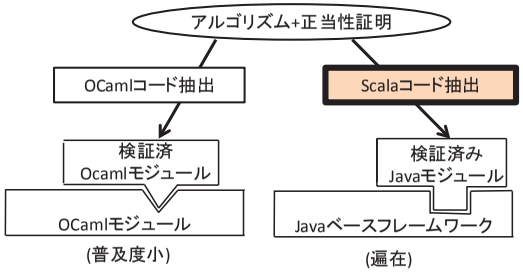



- Coqでの対象言語: OCaml, Haskell, Scheme

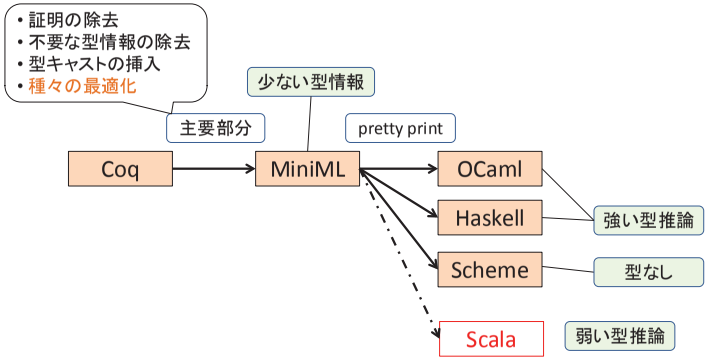
Scalaを対象言語としたコード抽出を実現したい

## モジュールの抽出

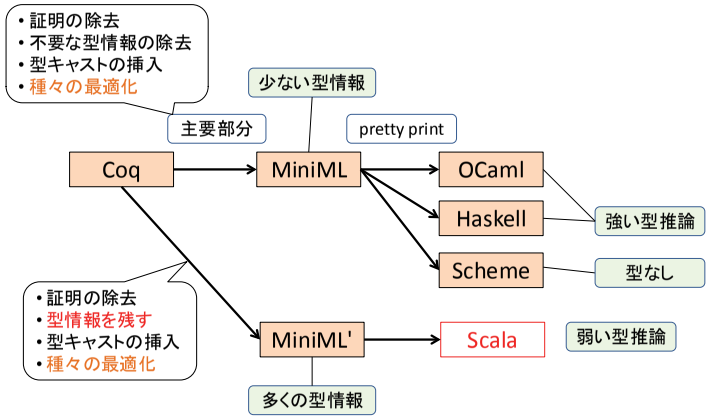
- 全プログラムをCoqで記述するのは非現実的
- キーとなる一部のモジュールだけを検証したい



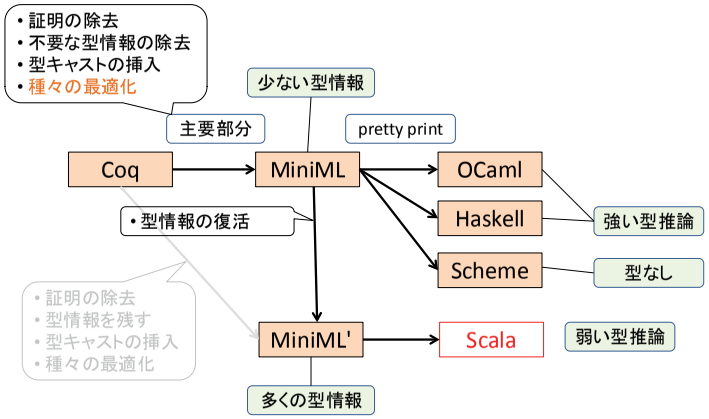

## 既存機能



## Scala抽出 (案1)




## Scala抽出 (案2)




## CoqからMiniMLへの変換

- Coq → MiniML

  e : T の抽出

  1. eをMiniMLの項に変換する
  2. TをMiniMLの型に変換する
  3. 1の項が2の型が持つように型キャストを挿入する

     型推論アルゴリズムM[Lee and Yi 1998]を元にした

     アルゴリズムM'による

     M 推論した型をもとに型検査する

     M' 推論した型をもとに型キャストを挿入する



## MiniMLの構文

MiniMLの項(ml_ast)、MiniMLの型(ml_type)の構文

| | |
|---|---|
| ml_ast $e \equiv$ | |
| $\mid x$ | 変数参照 |
| $\mid e\ e$ | 関数適用 |
| $\mid \text{fun}\ x \to e$ | 関数 |
| $\mid c$ | グローバル参照 |
| $\mid \text{let}\ x = e\ \text{in}\ e$ | let 式 |
| $\mid \text{match}\ e\ \text{with}$ | |
| $\quad c\vec{x} \to e \mid \cdots \mid c\vec{x} \to e$ | match 式 |
| $\mid \text{fix}\ x = e\ \text{with}\ \cdots\ \text{with}$ | |
| $\quad x = e\ \text{for}\ x$ | 再帰関数 |
| $\mid \text{magic}\ e$ | magic 式 |

| | |
|---|---|
| ml_type $T \equiv$ | |
| $\mid \alpha$ | (型変数) |
| $\mid C[\vec{T}]$ | (定数型) |
| $\mid T \to T$ | (関数型) |
| ml_type_scheme $\tau \equiv$ | |
| $\mid \forall \alpha.\tau$ | |
| $\mid T$ | |

## 型キャスト

- 例: 下の const1 を抽出する.

```
Definition nat_or_bool : forall b:bool, Set :=
  fun b => if b then nat else bool.

Definition zero_or_false
                 : forall b:bool, nat_or_bool b :=
  fun b => if b then 0 else false.

Definition const1 : nat :=
  (zero_or_false true) + 1.
```

## 型キャスト

- Propソートは現れないので, 削除する部分はない.

```
def zero_or_false : Bool -> Any = {
  (b:Bool) => b match {
      case True  => 0()
      case False => False() }}

def const1 : Nat =
  (zero_or_false true) + 1.
```

- しかし, これでは型が合わないので....

## 型キャスト

```
def zero_or_false : Bool -> Any = {
  (b:Bool) => b match {
      case True  => 0().asInstanceOf[Any]
      case False => False().asInstanceOf[Any] }}

def const1 : Nat =
  (zero_or_false.asInstanceOf[Bool->Nat] true) + 1
```

## MiniML では不足する情報

```
def zero_or_false : Bool -> Any = {
  (b:Bool) => b match {
      case True  => 0().asInstanceOf[Any]
      case False => False().asInstanceOf[Any] }}

def const1 : Nat =
  (zero_or_false.asInstanceOf[Bool->Nat] true) + 1
```

## MiniML'

MiniMLの項に型情報を付加した言語MiniML'を考える
MiniML'の項（ml_ast'）からScalaコードを書き出せる

$$
\begin{array}{ll}
\text{ml\_ast}'\ e' \equiv & \\
\quad \mid x[\vec{T}] & \text{変数参照} \\
\quad \mid e'\ e' & \text{関数適用} \\
\quad \mid \text{fun}\ x : T \to e' & \text{関数} \\
\quad \mid c[\vec{T}] & \text{グローバル参照} \\
\quad \mid \text{let}\ x : \tau = e'\ \text{in}\ e' & \text{let 式} \\
\quad \mid \text{match}\ e'\ \text{with} & \\
\qquad c\vec{x} \to e' \mid \cdots \mid c\vec{x} \to e' & \text{match 式} \\
\quad \mid \text{fix}\ x : T = e'\ \text{with}\ \ldots\ \text{with} & \\
\qquad x : T = e' \text{for}\ x & \text{再帰関数} \\
\quad \mid \text{magic}\ (e', T) & \text{magic 式}
\end{array}
$$

MiniML'の型（ml_type)は
MiniMLの型と共通

$$
\begin{array}{ll}
\text{ml\_type}\ T \equiv & \\
\quad \mid \alpha & \text{(型変数)} \\
\quad \mid C[\vec{T}] & \text{(定数型)} \\
\quad \mid T \to T & \text{(関数型)} \\
\\
\text{ml\_type\_scheme}\ \tau \equiv & \\
\quad \mid \forall \alpha.\tau & \\
\quad \mid T &
\end{array}
$$



## アルゴリズムM''

推論した型をもとに型情報を挿入した項を返す
　　M　推論した型をもとに型検査する
　　M'　推論した型をもとに型キャストを挿入する

M'' : env × ml_ast × ml_type → subst × ml_ast'
　型環境　　代入
M''(Γ, e, T)　を実行すると
Γの下で e : T であるために必要な型変数への代入を返す
同時に e に型情報を付加したMiniML'の項を返す



$$
\begin{array}{l}
M''(\Gamma, x, T) = \\
\quad \text{let}\ \sigma = unify(T, Inst(\vec{b}, \Gamma(x)))\ \text{in} \\
\qquad (\text{with}\ \vec{b}\ \text{fresh}) \\
\quad (\sigma, x[\sigma\vec{b}]) \\
M''(\Gamma, a\ b, T) = \\
\quad \text{let}\ (\phi_a, a') = M''(\Gamma, a, \alpha \to T)\ \text{in} \\
\quad \text{let}\ (\phi_b, b') = M''(\phi_a\Gamma, b, \phi_a\alpha)\ \text{in} \\
\qquad (\text{with}\ \alpha\ \text{fresh}) \\
\quad (\phi_a \circ \phi_b, a'\ b') \\
M''(\Gamma, \text{fun}\ x \to a, T) = \\
\quad \text{let}\ \sigma = unify(T, \alpha \to \beta)\ \text{in} \\
\qquad (\text{with}\ \alpha\ \text{and}\ \beta\ \text{fresh}) \\
\quad \text{let}\ (\phi_a, a') = M''(\sigma\Gamma + \{x : \sigma\alpha\}, a, \sigma\beta)\ \text{in} \\
\quad \text{let}\ \phi = \sigma \circ \phi_a\ \text{in} \\
\quad (\phi, \text{fun}\ x : \phi\alpha \to a') \\
M''(\Gamma, c, T) = \\
\quad \text{let}\ \sigma = unify(T, Inst(\vec{b}, E(c)))\ \text{in} \\
\qquad (\text{with}\ \vec{b}\ \text{fresh}) \\
\quad (\sigma, c[\sigma\vec{b}]) \\
M''(\Gamma, \text{let}\ x = a\ \text{in}\ b, T) = \\
\quad \text{let}\ (\phi_a, a') = M''(\Gamma, a, \alpha)\ \text{in} \\
\qquad (\text{with}\ \alpha\ \text{fresh}) \\
\quad \text{let}\ \tau_a = Gen_{\phi_a\Gamma}(\phi_a\alpha)\ \text{in} \\
\quad \text{let}\ (\phi_b, b') = M''(\phi_a\Gamma + \{x : \tau_a\}, b, T)\ \text{in} \\
\quad (\phi_a \circ \phi_b, \text{let}\ x : \phi_b\tau_a = a'\ \text{in}\ b')
\end{array}
$$

$$
\begin{array}{l}
M''(\Gamma, \text{match}\ a\ \text{with}\ c_1\vec{x}_1 \to b_1 \mid \cdots \mid c_n\vec{x}_n \to b_n, T) = \\
\quad \text{let}\ (\phi_a, a') = M''(\Gamma, a, \tau_a)\ \text{in} \\
\qquad (\text{with}\ \tau_a\ \text{fresh}) \\
\quad \text{let}\ (\sigma_i, b_i') = \\
\qquad M''(S_{i-1}\Gamma + \{x_{i1} : S_{i-1}Inst(\vec{\beta}, T_{i1}), \ldots, x_{ik_i} : S_{i-1}Inst(\vec{\beta}, T_{ik_i}), b_i, S_{i-1}T)\ \text{in} \\
\qquad (S_{i-1} = \sigma_{i-1} \circ \cdots \circ \sigma_0 \circ \phi_a, \sigma_0 = id) \\
\qquad (\text{with}\ E(c_i) = \forall \vec{a}.(T_{i1} \to \cdots \to T_{ik_i} \to I[\vec{a}]), \vec{\beta}\ \text{fresh}) \\
\qquad \text{for}\ i = 1..n \\
\quad (S_n, \text{match}\ a'\ \text{with}\ c_1\vec{x}_1 \to b_1' \mid \cdots \mid c_n\vec{x}_n \to b_n') \\
M''(\Gamma, \text{fix}\ x_1 = a_1\ \text{with}\ \ldots\ \text{with}\ x_n = a_n\ \text{for}\ x_k, T) = \\
\quad \text{let}\ (\sigma_i, a_i') = \\
\qquad M''(S_{i-1}\Gamma + \{x_1 : S_{i-1}\alpha_1, \ldots, x_k : S_{i-1}T, \ldots, x_n : S_{i-1}\alpha_n\}, a_i, S_{i-1}\alpha_i)\ \text{in} \\
\qquad (S_{i-1} = \sigma_{i-1} \circ \ldots \sigma_0, \sigma_0 = id) \\
\qquad \text{for}\ i = 1..n \\
\qquad (\text{with}\ \alpha_i\ \text{fresh}) \\
\quad (S_n, \text{fix}\ x_1 : S_n\alpha_1 = a_1'\ \text{with}\ \ldots\ \text{with}\ x_k : S_nT = a_k'\ \text{with}\ \ldots\ \text{with}\ x_n : S_n\alpha_n = a_n' \\
\qquad \text{for}\ x_k) \\
M''(\Gamma, \text{magic}\ a, T) = \\
\quad \text{let}\ (\phi_a, a') = M''(\Gamma, a, \alpha)\ \text{in} \\
\qquad (\text{with}\ \alpha\ \text{fresh}) \\
\quad (\phi_a, \text{magic}\ (a', T))
\end{array}
$$



$$
\begin{array}{l}
(S_n, \text{match}\ a'\ \text{with}\ c_1\vec{x}_1 \to b_1' \mid \cdots \mid c_n\vec{x}_n \to b_n') \\
M''(\Gamma, \text{fix}\ x_1 = a_1\ \text{with}\ \ldots\ \text{with}\ x_n = a_n\ \text{for}\ x_k, T) = \\
\quad \text{let}\ (\sigma_i, a_i') = \\
\qquad M''(S_{i-1}\Gamma + \{x_1 : S_{i-1}\alpha_1, \ldots, x_k : S_{i-1}T, \ldots, x_n : S_{i-1}\alpha_n\}, a_i, S_{i-1}\alpha_i)\ \text{in} \\
\qquad (S_{i-1} = \sigma_{i-1} \circ \ldots \sigma_0, \sigma_0 = id) \\
\qquad \text{for}\ i = 1..n \\
\qquad (\text{with}\ \alpha_i\ \text{fresh}) \\
\quad (S_n, \text{fix}\ x_1 : S_n\alpha_1 = a_1'\ \text{with}\ \ldots\ \text{with}\ x_k : S_nT = a_k'\ \text{with}\ \ldots\ \text{with}\ x_n : S_n\alpha_n = a_n' \\
\qquad \text{for}\ x_k) \\
M''(\Gamma, \text{magic}\ a, T) = \\
\quad \text{let}\ (\phi_a, a') = M''(\Gamma, a, \alpha)\ \text{in} \\
\qquad (\text{with}\ \alpha\ \text{fresh}) \\
\quad (\phi_a, \text{magic}\ (a', T))
\end{array}
$$

magic式を扱えるようにMを拡張

## M''の性質

定理1

任意のMiniMLの項と型 e, T に対して、|- e : T ならば
M''([], e, T) は失敗せずにMiniML'の項e' を返して

|- e' : T

が成り立つ。

証明

Mの健全性、完全性の証明の拡張

17

## コード抽出の妥当性

抽出後のコードが

- 型の妥当性
  型検査を通る
- 計算の妥当性
  抽出前のコードと抽出後のコードが,「同じ計算」を実行する.

18

## 型の妥当性

[Letouzey, 2004]

標準のコード抽出は型の妥当性を満たす

定理2

任意のCoq上の定義 e : T から抽出した
MiniMLの項 $e_{MiniML}$ と型 $T_{MiniML}$ は

$|- e_{MiniML} : T_{MiniML}$

を満たす。

19

## 型の妥当性

本研究の方式によって抽出されるコードは型の妥当性を満たす

定理3

任意のCoq上の定義 e : T からこの方式によって得た
MiniML'の項 e' と型 T' は

|- e' : T'

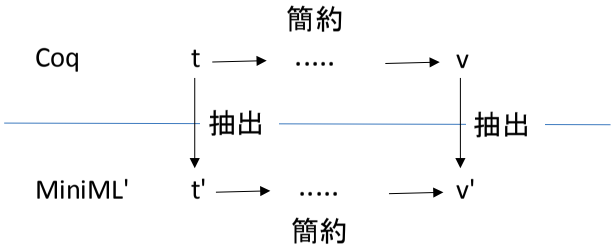
を満たす。

証明: 定理1, 2 の組み合わせ

20

## 実行の妥当性

- 抽出したコードが，抽出前のコードと「同じ計算」を実行する．




## 型キャストの問題

- 実行の妥当性の証明は，OCaml の場合を真似するだけではない部分がある．
- OCaml では，型キャストに用いる Obj.magic はエラーを起こさない(実行時は no-op).

  (Obj.magic true) + 1

  ここは通る　ここでエラー

- Scala では，asInstanceOf[] 自体がエラーを起こす．

  True().asInstanceOf[Nat] + 1

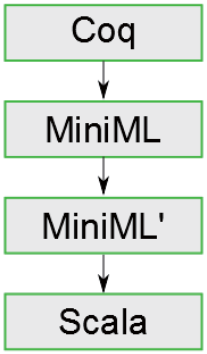
  BoolとNatが異なる帰納型なのでエラー

- 下のようになる例があると破綻する

  True().asInstanceOf[Nat].asInstanceOf[Bool] && b

## 実装

CoqからScalaへのコード抽出プログラム (OCaml)




Coqのソースコードに含まれる実装

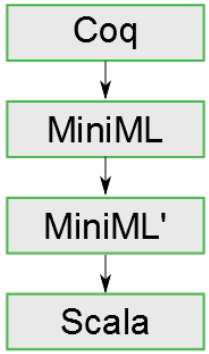
M''の実装

自然な対応に忠実に実装



## 実装1

実装済み



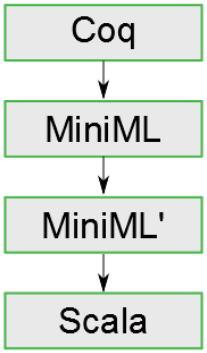

Coqのソースコードに含まれる実装

OCamlでM''を実装

自然な対応に忠実に実装

## 実装2

実装途中




Coqのソースコードに含まれる実装

Coq上でアルゴリズムM''の実装と定理1の証明(4000行)をした後、OCamlへ抽出して利用

自然な対応に忠実に実装



## 先行研究

W [Damas and Milner, 1982]の形式的証明
(Mのもととなった型推論アルゴリズム)

- Coq
  [C. Dubois and V. Menissier-Morain, 1999]
  [J. Garrigue, 2010]
- Isabelle/HOL
  [D. Nazareth and T. Nipkow, 1999]
  [C. Urban and T. Nipkow, 2009]



## Coq上のM''の実装

今回はCoq上で証明して終わりではなく
抽出後のコードを考慮する必要がある

- アルゴリズムの失敗の表現
- 新しい型変数の生成



## まとめ

- MiniMLの項に型の情報を付加するアルゴリズムM''を定義し、Scalaへのコード抽出を設計，実装
- Coq上でM''の定義と定理1の証明を実装
- 今後の課題
  – 計算の妥当性を満たすことの証明
  – 実際のコードへの適用
  – Scalaへのコード抽出全体の正当性が保証された実装

# Formalization of Featherweight Java by using weak HOAS

*Kentaro Okumura Atsushi Igarashi

Kyoto University

Dec 4th, 2014

1

## Featherweight Java (FJ) [Igarashi et al. 2001]

- A formal language for OO languages
- Extremely simplified Java
  - All of FJ programs are Java programs
- Some extensions
  - Featherweight GJ (FGJ) [Igarashi et al. 2001]
    - FJ+generics

2

## An example of FJ program

```
class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Object fst, Object snd) {
    super(); this.fst = fst; this.snd = snd;
  }
  Pair setfst(Object newfst) {
    return new Pair(newfst, this.snd);
  }
}
```

3

## About our work

- Goal
  - Formalization of FJ and FGJ, and proofs of type soundness properties of them
- Methods
  - Coq proof assistant
  - Weak Higher Order Abstract Syntax (Weak HOAS) [Despeyroux et al. 1994]
    - Methodology for formalization of variable bindings
- The Coq code is available: https://www.github.com/OKU1987/FJ_whoas

4

## Today's talk

- Background
- About FJ
- Formalization of FJ by using weak HOAS
- Related work and conclusion

5

## Featherweight Java (FJ)

- A formal language for OO languages
- Extremely simplified Java
  - All of FJ programs are Java programs
  - no assignments
- Semantics of FJ is defined by term rewriting like λ-calculus
- Pencil-and-paper proof of type soundness

6

## Syntax of FJ

constructors

$L ::= \text{class } C \text{ extends } C \ \{ \ \overline{C} \ \overline{f}; K \ \overline{M} \ \}$ (class decl.)

$M ::= C \ m(\overline{C} \ \overline{x}) \ \{ \text{ return } e; \ \}$ (method decl.)

$e ::= x \mid e.f \mid e.m(\overline{e}) \mid \text{new } C(\overline{e})$ (expression)

including "this"

$\overline{C} \ \overline{f}; = C_1 \ f_1; \ldots C_n \ f_n;$

$\overline{C} \ \overline{x} = C_1 \ x_1, \ldots, C_n \ x_n$

$\overline{e} = e_1, \ldots, e_n$

7

## The Pair example, again

```
class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Object fst, Object snd) {
    super(); this.fst = fst; this.snd = snd;
  }
  Pair setfst(Object newfst) {
    return new Pair(newfst, this.snd);
  }
}
```

binding

bound implicitly

8

## A typing rule of FJ

$$\frac{\boxed{\bar{x}:\bar{C}, \text{this}:C \vdash e_0 : E_0} \qquad E_0 <: C_0 \qquad \text{class } C \text{ extends D } \{\ldots\} \qquad override(m, D, C)}{C_0\ m(\bar{C}\ \bar{x})\ \{\ \text{return } e_0;\ \}\ \text{OK IN } C}\ \text{(T\_Method)}$$

Every type of FJ is a class name

9

## A reduction rule of FJ

- A reduction rule for method invocation

$$\frac{mbody(m, C) = \bar{x}.e_0}{(\text{new } C(\bar{e})).m(\bar{d}) \longrightarrow [\bar{d}/\bar{x}, \underline{\text{new } C(\bar{e})/\text{this}}]e_0}\ \text{(R\_Invk)}$$

"this" and parameters are replaced simultaneously

10

## Type soundness [Igarashi et al. 2001]

If $\varnothing \vdash$ e:C and e $\longrightarrow$* e' (where e' is normal), then e' is a value v such that $\varnothing \vdash$ v:D and D<:C

Type soundness is proved by using preservation and progress theorems

11

## Formalization of FJ

- How to represent variable bindings?
- How to represent methods?
  - Each method has different number of parameters.
- How to represent typing rules and typing contexts?
- How to represent "this"?
  - We want to
    - treat "this" as special but sometimes not
    - represent variables and "this" uniformly as much as possible

12

## Problem of variable bindings

- Definitions of substitution and renaming are troublesome
  - How to represent fresh variables?
- There are variable bindings in FJ
  - we use weak HOAS in this work

13

## Weak HOAS

- Methodology for formalizing of variable bindings
- Represent variable bindings of FJ by function abstractions of Coq
- We can utilize renaming of Coq

14

## Representation of variables and expressions by using weak HOAS

- Represent variables of FJ by using type V
  - Type V is a parameter

```
Parameter V : Set.
Inductive Exp : Set :=
| e_var : V → Exp
| fd_access : Exp → F → Exp    (* F is a field name *)
| m_call : Exp → M → list Exp → Exp
                                (* M is a method name *)
| new : C → list Exp → Exp.
```

15

## Representation of variable bindings by using weak HOAS

- Represent variable bindings by using function abstractions of Coq

$m(C_0\ x_0)$ { return $x_0$.m(); }

We ignore types of parameters for now

16

## Representation of methods

m() { return new C(); } new C [] : Exp

m($C_0$ $x_0$) { return $x_0$.f; }

(fun $x_0$ : V => ...$x_0$...) : V → Exp

m($C_0$ $x_0$, $C_1$ $x_1$) { return $x_1$.f; }

(fun $x_0$ : V =>

(fun $x_1$ : V => ...$x_1$...)) : V → (V → Exp)

...

- We want to represent these methods in a single type.



## Representation of method body

Inductive MB : Set :=

| mb_empty : Exp → MB (* method body *)

| mb_var : (V → MB) → MB. (* take a parameter *)

a variable binding



## Examples of method body (MB type)

mb_empty (new C []) : MB

mb_var (fun $x_0$ : V => mb_empty (...$x_0$...)) : MB

mb_var (fun $x_0$ : V =>

mb_var (fun $x_1$ : V =>

mb_empty (...$x_1$...))) : MB

...



## Representation of typing context

Regarded as binding

Represented by using an abstraction

$$\frac{\overline{x}:\overline{C}, \text{this}:C \vdash e_0 : E_0 \qquad E_0 <: C_0 \qquad \text{class } C \text{ extends D } \{...\} \qquad override(m, D, C_0)}{C_0\ m(\overline{C}\ \overline{x})\ \{\text{ return } e_0;\ \}\ \text{OK IN } C}\ (\text{T\_Method})$$

Represented by using an abstraction

## Representation of typing context

- Express typing contexts by using function abstractions
  - Parameterize V, which express FJ's variables, by Ty
  - V t is the set of variables which have type t

```
Parameter V : Ty → Set.
Inductive Exp : Set :=
| e_var : forall t, V t → Exp
| fd_access : Exp → F → Exp
...
```

21

## Modification of definition of method body

```
Inductive MB : Set :=
| mb_empty : Exp → MB            (* method body *)
| mb_var : (V → MB) → MB.  (* take a parameter *)
```

```
Inductive MB : Set :=
| mb_empty : Exp → MB
| mb_var : forall t : Ty, (V t → MB) → MB.
```

22

## Modification of definition of method body…

- … makes MB represent types of parameters

m() { return new C(); }  mb_empty (new [])

m($C_0$ $x_0$) { return $x_0$.f; }
  mb_var (fun $x_0$ : V $C_0$ => mb_empty (…$x_0$…))

m($C_0$ $x_0$, $C_1$ $x_1$) { return $x_1$.f; }
  mb_var (fun $x_0$ : V $C_0$ =>
    mb_var (fun $x_1$ : V $C_1$ => mb_empty (…$x_1$…)))
…

23

## Representation of type judgments of expressions

```
Inductive Exp_WF : Exp → Ty → Prop :=
| T_Var : forall t (v : V t), Exp_WF (e_var v) t
| T_Fields : …
| T_Invk : …
| T_New : ….
```

24

## Representation of type judgments of methods

Inductive Exp_WF_in_MB : MB → Ty → Prop :=
| wf_empty : forall e t, Exp_WF e t →
Exp_WF_in_MB (mb_empty e) t

empty contexts

| wf_var : forall t (f : V t → MB) ty,
(forall v : V t, Exp_WF_in_MB (f v) ty) →
Exp_WF_in_MB (mb_var f) ty.

other contexts

$$\frac{\bar{x} : \bar{C}, \text{this}:C \vdash e_0 : E_0 \quad \ldots}{C_0 \ m(\bar{C}\ \bar{x})\ \{\ \text{return } e_0;\ \}\ \text{OK IN } C}\ (\text{T\_Method})$$

25

## Binding of this

- Every method has an implicit "parameter" of this

```
class Pair extends Object {
  Object fst;
  Object snd;
  Pair(Objest fst, Object snd) {
   super(); this.fst = fst; this.snd = snd;
  }
 Pair setfst(Object newfst) {
  return new Pair(newfst, this.snd);
 }
```

bound implicitly and exactly once

26

## "this" as a usual variable

- We want to define followings uniformly:
  - substitution into arguments and "this"
  - typing rules

$$\frac{mbody(m, C) = \bar{x}.e_0}{(\text{new } C(\bar{e})).m(\bar{d}) \longrightarrow [\bar{d}/\bar{x}, \text{new } C(\bar{e})/\text{this}]e_0} \qquad \frac{(x : T) \in \Gamma}{\Gamma \vdash x : T}$$

27

## Modification of MB

- to enforce implicit bindings of "this"

Inductive preMB : Set :=
| mb_empty : E → preMB
| mb_var : forall t : Ty, (V t → preMB) → preMB.

Inductive MB : Set :=
| mb_this : forall t : Ty, (V t → preMB) → MB.

"this" is bound exactly once

28

## Modification of a typing rule

- following modification of MB

```
Inductive E_WF_in_MB (c:Ty) : MB → Ty → Prop :=
| wf_e_mb_this :
    forall (f : V c → preMB) ty,
    (forall v, E_WF_in_preMB c (f v) ty) →
    E_WF_in_MB c (mb_this _ f) ty.
```

$$\frac{\overline{x}:\overline{C}, \text{this}:C \vdash e_0 : E_0 \quad \ldots}{C_0\ m(\overline{C}\ \overline{x})\ \{\ \text{return}\ e_0;\ \}\ \text{OK IN}\ C}\ (\text{T\_Method})$$

29

## Proof of type soundness

- by using former definitions on Coq

The number of lines of Coq codes.

Definition: 316 lines
Proof: 707 lines
Utilities: 81 lines

30

## Related work

- Jinja is not Java [Klein et al. 2014]
- A Theory of Featherweight Java in Isabelle/HOL [Foster et al. 2006]
- AutoSubst [Schäfer et al. 2014]
  - Coq library for renaming and substitution
  - We can define substitution easily with some keywords
  - Using de Bruijn Index
  - Mutually recursive types are not supported

31

## Conclusion

- We formalize a calculus for OO languages by using higher order abstract syntax
- We show that we can apply a HOAS method even if each method of FJ has different number of parameters
- We treat problems of “this”
- The Coq code of our formalization is available: https://www.github.com/OKU1987/FJ_whoas
- On going work: Formalization of FGJ

32

# メモリモデルを考慮した 汎用型付中間言語設計に向けて

Towards Design of
Universal Typed Intermediate Languages
in Consideration of Memory Models

八杉 昌宏
九州工業大学 情報工学研究院

2014年12月4日 (午後のみ参加)



# はじめに(お詫び)

- 今回，このTPPにて発表していますが，現時点では定理証明にあまり結びついていません
- 正しい共有メモリ向け並列プログラムを作成するには，という点からの話題提供となります
- また，言語設計研究としても本当に萌芽段階にいます (発表内容は動機と整理のみ)
- 最終的な言語処理系では依存型等を用いた型検査で正しさを確認したいとも考えています
- 型システム，型検査の健全性を示すときには，定理証明と結びつくかとは思います



# 動機

- 複数の高水準言語からの共通の型付中間言語
  - これ自体が挑戦的
  - 最終的には処理系を作りたい: 既存の技術でよければ，それを使っていく
    - ごみ集めの研究，並列処理の研究などに使えるもの
- 複数の高水準言語間で，安全な並列処理
  - マルチコア時代
  - 単一言語における安全な並列処理でも課題
- 現状がまずそう
  - よく考えたつもりが，実はデータレース(data race)がある場合が多い
  - まずは，型付中間言語における data-race-free



# 背景

- 型付中間言語
  - メモリアクセスに対する型付け
    - 参照，オフセット，シングルトン型
- Data-race-free なプログラムはデータについて sequential consistency
  - C11 (C言語の比較的新しい仕様)
  - Java は data-race free でない場合も semantics
- 発表者が15年くらい前に考えていたこと (xccmem.h)
  - 順序制約の明示
    - それなしでは，アクセスの順序が入れ替わる
      - コンパイラによって，あるいは，ハードウェアによって

# Data Race (一般的な)

- メモリロケーションを同期用とデータ用に分ける
- 同期用のロケーションを使って様々な同期を実現
  - 例: バリア同期，データフロー同期
  - 例: mutual exclusion (lock)
  - 同期においては race あり (lock の獲得など)
- Data race: 同じロケーションに関して，順序関係のついていない2つのデータアクセスのすくなくとも一方が write
  - 順序関係: “happens before”
    - 言語仕様で定める
    - コンパイラやハードウェアによる最適化 (命令スケジューリング等) の限度



# C11では

- Any such data race results in undefined behavior.

(解釈案)

- データレースを認めると，コンパイラやハードウェアによる最適化の影響が出て，その範囲は容易に限定できないということかと
  - プログラム実行の「状態」すら不明確

(課題) (本研究はC11が直接の対象ではないが)

- Data race free かコンパイル時に検査されない
- 近年活発な研究領域
  - 例: プログラム論理 GPS [Turon et al. OOPSLA 2014]
    - Ghost state, protocols, and separation logic
    - Data race free に限らない内容
    - C11 仕様に関する健全性を Coq で証明しているとのこと



# Data race の等価な定義

- SC (sequential consistency)を仮定して，並行スレッドが interleaving で実行を続けたときに，決して並行読み書き，並行書き込みが生じないなら，data-race-free (なのでSC)
  - SCを仮定すれば最後に書き込んだ値が決まる
  - SCでないとスレッドによって順序が異なる値が読まれるか，もっとひどいことに.
- ただし，同期用ロケーションに関する，同期のための read や write については一般にはSC ではないため，全体としては等価とはいえないかもしれない.
  - GPS [Turon et al.] はこれらにも程よく対応
- Lock などに限定すればおそらく問題ない.



# 等価なデータレース定義

(さらに，ここでの整理案)

- SC を仮定してプログラムの並行実行中の状態 $s$ から $s'$ に遷移するとする.
  - $\forall s \in$ DRF. $\forall s' \in$ next ($s$). $s' \in$ DRF
  - $\forall s \in$ DRF. ($s$ が今まさにdata race ではない)
    - 「今まさにdata race」=「次のアクションとして，同じ場所に関して，書き込みを含む複数通りのアクセスが可能」

  を満たす最大の状態集合DRFを考える
- プログラムの初期状態が DRF に含まれるか?
  - DRFに含まれることを近似する程よい型システムとして判定したい
  - DRF を エラーフリーと読めば，型検査で検査したいことと同じ形 (よって，型安全性 Soundness of a type system は標準的)

## 既存技術: fractional permission/capability/ownership

- 順序関係に基づく定義の近似
- 例: [Terauchi, PLDI 2008]
- 全スレッド含むシステム全体の合計1以下 (を確認できる型システム)
  - 1を持っていないと，書き込めない
  - 0より多くを持っていないと，読めない
- Data race free
  - 1を持っていれば他に読み書きするスレッドはいない
- 持つ量が型の一部，持つ量に関する多相性は考えない範囲なら linear programming solvers で
- ロック以外のプロトコルに対応できるのか(調査中)



## 既存技術: Concurrent Separation Logic

- $$\frac{\{P_1\}\ e_1\ \{Q_1\} \quad \{P_2\}\ e_2\ \{Q_2\}}{\{P_1 * P_2\}\ e_1 || e_2 \{Q_1 * Q_2\}}$$
- メモリの所有者毎に分離
- $e_1$は $P_1$ が表す部分，$e_2$は $P_2$ が表す部分を所有
- Separating conjunction $P_1 * P_2$で分離
  - GPS もこれに基づく．GPS は SC でない場合にも対応という点で，Concurrent Separation Logic より進んでいる
  - ただ，concurrent reads に未対応
    - おそらくfractional な
      $l \hookrightarrow_f v * l \hookrightarrow_p v \Leftrightarrow l \hookrightarrow_{f+p} v$
      といったことで対応できるはず．(調査中)



## 並列プログラム例

- データ構造の生成と利用
  - 並列に木を構成，データレースが生じないようにしつつ
  - 利用フェーズでは，共有され read only
    - フェーズが変わったことを順序関係として示したい
- 並列行列計算
  - 未処理の部分を分離して更新
  - 処理済みの部分は共有され read only
  - 処理済みの部分を表すプロトコル(バリア同期などを含め)が必要
- これらの例では単なる lock では厳しい



## 難しい点

- Data race free をいうには，システム全体，あるいは将来の実行について確認する必要がある．(順序関係がグラフ)
- システム全体で「ない」ことを言うには，スレッド単位での確認を難しくする．

## 出発点として既存技術の組み合せ

- プロトコル等の実現
  - GPS の技術を利用できるか
- Concurrent reads を許す
  - Fractional で
- 足りない部分
  - 調査中
  - Fractional な所有権については一般の場合は，多相性(fractionalな所有権を半分にするなど)もほしい．推論しないなら可能か．



## 高水準でのメモリモデル (コンシステンシ)

- さらに挑戦的
- 型付中間言語レベルで data race free でも，高水準言語レベルでは consistency が不十分な可能性
  - そもそも consistency とは何か
  - 高水準 data race の概念?
  - 複数の高水準言語?



## おわりに

- 複数の高水準言語からの共通の型付中間言語
  - これ自体が挑戦的
  - 最終的には処理系を作りたい
- 複数の高水準言語間で，安全な並列処理
  - 単一言語における安全な並列処理でも課題
  - まずは，型付中間言語における data-race-free

# An Intuitionistic Set-theoretical Model of the Extend Calculus of Construction

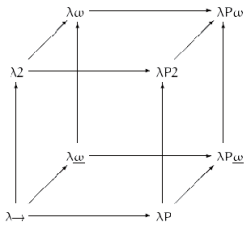
Masahiro Sato

December 4, 2014



## ECC

- ECC is a extention of CC($\lambda$P$\omega$)[6]
  - CC is the strongest type system in $\lambda$-Cube
  - ECC = CC + Universal Type (+ $\Sigma$-Type)
  - CIC = CC + Universal Type + (Co)Inductive Type
    - Coq's type system is CIC.




## A semantics of type system

| | $t : T$ | $f : A \to B$ | $f : \forall x : A.B$ |
|---|---|---|---|
| Set theory | $t$ is a element of Set(Class?) $T$ | $f$ is a function from $A$ to $B$ | $f$ is a function which maps $a$ into some element of $B(a)$ |
| Programming | $t$ is a variable whose type is $T$ | $f$ is a program with argument of type $A$ whose return value is type $B$ | $f$ is a dependent type function(program) |
| Proof theory | $t$ is a proof of a proposition $T$ | $f$ is a proof of a proposition $A \Rightarrow B$ | $f$ is a proof of a proposition $\forall x \in A, B(x)$ |

## Definition of Term and Context(ECC)

### Definition (Term)

- $\mathrm{Type}_i$ is a term $(i = 0, 1, 2, 3, 4, ...)$.
- $\mathrm{Prop}$ is a term.
- $x$ is a term for $x \in V$.
- If $t_1$ and $t_2$ are terms, then $t_1 t_2$ is a term.
- If $t$ and $T$ are terms, and $x \in V$ then, $\lambda x : T.t$ is a term.
- If $T_1$ and $T_2$ is a term, and $x \in V$ then $\forall x : T_1.T_2$ is a term.
  - If $x$ dosn't freely appear in $B$, $\forall x : A.B$ is denoted as $A \to B$.

### Definition (Context)

- $[]$ is a Context.
- If $\Gamma$ is a Context, $T$ is a term and $x \in V$, then $\Gamma; (x : T)$ is a Context.

## Typing-Rule1

$$\frac{\Gamma; (x : A) \vdash t : B \quad \Gamma \vdash \forall x : A.B : \mathrm{Type_i}}{\Gamma \vdash \lambda x : A.t : \forall x : A.B} \qquad \frac{\Gamma; (x : A) \vdash t : B \quad \Gamma \vdash \forall x : A.B : \mathrm{Prop}}{\Gamma \vdash \lambda x : A.t : \forall x : A.B}$$

$$\frac{\Gamma \vdash u : \forall x : A.B \quad \Gamma \vdash v : A}{\Gamma \vdash (uv) : B[x \backslash v]}$$

$$\frac{(x : A) \in \Gamma \quad \Gamma \vdash A : \mathrm{Type}_i}{\Gamma \vdash x : A} \qquad \frac{(x : A) \in \Gamma \quad \Gamma \vdash A : \mathrm{Prop}}{\Gamma \vdash x : A}$$

$$\frac{\Gamma \vdash x : A \quad A =_\beta B}{\Gamma \vdash x : B}$$

## Typing-Rule2

$$\Gamma \vdash \mathrm{Prop} : \mathrm{Type}_i \qquad \Gamma \vdash \mathrm{Type}_i : \mathrm{Type}_{i+1}$$

$$\frac{\Gamma \vdash A : \mathrm{Prop}}{\Gamma \vdash A : \mathrm{Type}_i} \qquad \frac{\Gamma \vdash A : \mathrm{Type}_i}{\Gamma \vdash A : \mathrm{Type}_{i+1}}$$

$$\frac{\Gamma \vdash A : \mathrm{Type}_i \quad \Gamma; (x : A) \vdash B : \mathrm{Type}_j}{\Gamma \vdash \forall x : A.B : \mathrm{Type}_{\max(i,j)}} \qquad \frac{\Gamma \vdash A : \mathrm{Prop} \quad \Gamma; (x : A) \vdash B : \mathrm{Type}_j}{\Gamma \vdash \forall x : A.B.\mathrm{Type}_j}$$

$$\frac{\Gamma \vdash A : \mathrm{Type}_i \quad \Gamma; (x : A) \vdash Q : \mathrm{Prop}}{\Gamma \vdash \forall x : A.Q : \mathrm{Prop}} \qquad \frac{\Gamma \vdash P : \mathrm{Prop} \quad \Gamma; (x : A) \vdash Q : \mathrm{Prop}}{\Gamma \vdash \forall x : P.Q : \mathrm{Prop}}$$

## Merit of "Prop"

- "Prop" enables to represent of higher order logic
  - The type of " Predicate on Type 'A' " is $A \to \mathrm{Prop}$
  - $\forall P : A \to \mathrm{Prop}, Q(P) : \mathrm{Prop}$
- "Prop" represents other logical symbols as follows:
  - $\bot := \forall P : \mathrm{Prop}.P$
  - $\neg A := A \to \bot$
  - $\exists x : A.B := \forall P : \mathrm{Prop}.(\forall x : A, (B(x) \to P)) \to P$
  - $A \wedge B := \forall P : \mathrm{Prop}.(A \to B \to P) \to P$
  - $A \vee B := \forall P : \mathrm{Prop}.(A \to P) \to (B \to P) \to P$
  - $A \leftrightarrow B := A \to B \wedge B \to A$
  - $x =_A y := \forall P : \mathrm{Prop}, Px \leftrightarrow Py$

### Definition (Propositional Term and Proof Term)

The term $P$ is called a propositional term for $\Gamma$ iff $\Gamma \vdash P : \mathrm{Prop}$ is derivable. The term $t$ is called a proof term for $\Gamma$ iff $\Gamma \vdash t : P$ is derivable for some $P$ which is a propositional term for $\Gamma$.

### Definition (Provable Propositional term)

Let $P$ be a propositional term for $\Gamma$. $P$ is called a provable propositional term if there exists the term $t$ such that $\Gamma \vdash t : P$ is derivable.



## Examples of Model of Type Theory

- Categorical Model
- Coherence Space
- Set Theoretical Model
  - Soundness of ECC is showed by B.Werner and M.Alexandre(1997, 2003).

## Dependent Function

### Definition (Dependent Function)

Let $A$ be a set, and $B(a)$ is a set with parameter $a \in A$.

$$\begin{aligned} \coprod_{a \in A} B(a) &:= \{p \in A \times \bigcup_{a \in A} B(a) \mid \forall a, b, (a, b) \in p \Rightarrow b \in B(a)\} \\ \prod_{a \in A} B(a) &:= \{f \subset \coprod_{a \in A} B(a) \mid \forall a \in A, \exists! b \in B(a), (a, b) \in f\} \end{aligned}$$

Dependent Function is a function with range $B(a)$ with a parameter $a \in A$, i.e.

$$f \in \prod_{a \in A} B(a) \Rightarrow \forall a \in A, f(a) \in B(a).$$

## Universe

### Definition (Universe)

Let $\mathcal{U}(i)$ be $i$-th Grothendieck Universe, i.e.

- $\mathcal{U}(i)$ is a Grothendieck Universe for each $i \in \mathbb{N}$.
- $\mathcal{U}(i) \in \mathcal{U}(i+1)$ for each $i \in \mathbb{N}$.

### Lemma

$$A \in \mathcal{U}(i) \quad \wedge \quad \forall a \in A, B(a) \in \mathcal{U}(i) \quad \Rightarrow \quad \prod_{a \in A} B(a) \in \mathcal{U}(i).$$

## Interpretation(Context)

### Definition (Interpretation of Context)

- $[\![ [] ]\!] := ()$
- $[\![ \Gamma; (x : A) ]\!] := \{(\gamma, \alpha) | \alpha \in [\![ \Gamma \vdash A ]\!](\gamma)\}$

## Interpretation(Judgement)

### Definition (Interpretation of Judgement)

- $[\![ \Gamma \vdash t ]\!](\gamma) := 0$ (If $t$ is a proof term for $\Gamma$)
- $[\![ \Gamma \vdash \mathrm{Type}_i ]\!](\gamma) := \mathcal{U}(i)$
- $[\![ \Gamma \vdash \mathrm{Prop} ]\!](\gamma) := \{0, 1\} = \{\phi, \{\phi\}\}$
- $[\![ \Gamma \vdash \lambda x : A.T ]\!](\gamma) := \{(\alpha, [\![ \Gamma; (x : A) \vdash T ]\!](\gamma, \alpha)) | \alpha \in [\![ \Gamma \vdash A ]\!](\gamma)\}$
- $[\![ \Gamma \vdash t_1 t_2 ]\!](\gamma) := [\![ \Gamma \vdash t_1 ]\!](\gamma)([\![ \Gamma \vdash t_2 ]\!](\gamma))$
- $[\![ \Gamma \vdash x_i ]\!](\gamma) := \gamma_i$
- $[\![ \Gamma \vdash \forall x : A.B ]\!](\gamma)$
  - $:= [\![ \Gamma \vdash A ]\!](\gamma) \ \vee \ \neg [\![ \Gamma; (x : A) \vdash B ]\!](\gamma, 0)$
    (when $A$ and $B$ are both Propositional terms)
  - $:= \min\{ \ [\![ \Gamma; (x : A) \vdash B ]\!](\alpha, \gamma) \mid \alpha \in [\![ \Gamma \vdash A ]\!](\gamma) \ \}$
    (when $B$ is a Propositional term)
  - $:= \prod_{[\![ \alpha \in \Gamma \vdash A ]\!](\gamma)} [\![ \Gamma; (x : A) \vdash B ]\!](\gamma, \alpha)$
    (Otherwise)

## Soundness

### Theorem (Soundness)

- *If $\Gamma \vdash t : T$ is derivable, then $[\![ \Gamma \vdash t ]\!](\gamma) \in [\![ \Gamma \vdash T ]\!](\gamma)$ for all $\gamma \in [\![ \Gamma ]\!]$*
- *If $\Gamma \vdash t_1 : T$ and $t_1 \to_\beta t_2$, then $[\![ \Gamma \vdash t_1 ]\!](\gamma) = [\![ \Gamma \vdash t_2 ]\!](\gamma)$ for all $\gamma \in [\![ \Gamma ]\!]$*

*Where $[\![ \Gamma \vdash T ]\!]$ is a function whose domain is $[\![ \Gamma ]\!]$.*

Definition of ECC

Interpretation

Extended Interpretation

Future Works

## idea

- Above model is still 'Classical Model'.
  - In this model, PEM is true.
    - $[\![\Gamma \vdash \forall P : \mathrm{Prop}, P \vee \neg P]\!](\gamma) = 1$
  - Can we create the model which is PEM is NOT true?
- Expand the interpretation of $\mathrm{Prop}$ from $\{0, 1\}$ into general topological space.

## Notation of topological space

Let $(X, \mathcal{O}(X))$ be a topological space.

- $\mathbb{I} := X$
- $\mathbb{O} := \phi$
- $\bigsqcup S := \bigcup S$
- $\bigsqcap S := \bigsqcup\{t \mid \forall s \in S, t \subset s\} = \left(\bigcap S\right)^{\circ}$
- $b^a := \bigsqcup\{t \mid t \sqcap a \leq b\}$

## Point condition

Let $p$ be a point of $X$ such that following condition holds.

$$\bigcap \mathcal{N}(p) \; is \; open \; set. \tag{1}$$

where $\mathcal{N}(p)$ is open neighborhood, i.e.

$$\mathcal{N}(p) := \{O \in \mathcal{O}(X) | p \in O\}$$

We consider the model with a parameter $p(\in X)$ satisfying above condition.

## Interpretation(Context)

### Definition (Strict Interpretation of Judgement)

$$[\![\Gamma \vdash A]\!]'(\gamma) := \begin{cases} [\![\Gamma \vdash A]\!](\gamma) \cap \{p\} & (A \textit{ is a propositional term}) \\ [\![\Gamma \vdash A]\!](\gamma) & (\textit{otherwise}) \end{cases}$$

### Definition (Interpretation of Context)

- $[\![[]]\!] := ()$
- $[\![\Gamma; (x : A)]\!] := \{(\gamma, \alpha) | \alpha \in [\![\Gamma \vdash A]\!]'(\gamma)\}$

## Interpretation(Judgement)

### Definition (Interpretation of Judgement)

- $[\![\Gamma \vdash t]\!](\gamma) := p$ (If $t$ is Proof term for $\Gamma$)
- $[\![\Gamma \vdash \mathrm{Type}_i]\!](\gamma) := \mathcal{U}(i)$
- $[\![\Gamma \vdash \mathrm{Prop}]\!](\gamma) := \mathcal{O}(X)$
- $[\![\Gamma \vdash \lambda x : A.T]\!](\gamma) := \{(\alpha, [\![\Gamma; (x : A) \vdash T]\!](\gamma, \alpha)) | \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}$
- $[\![\Gamma \vdash t_1 t_2]\!](\gamma) := [\![\Gamma \vdash t_1]\!](\gamma)([\![\Gamma \vdash t_2]\!](\gamma))$
- $[\![\Gamma \vdash x_i]\!](\gamma) := \gamma_i$
- $[\![\Gamma \vdash \forall x : A.B]\!](\gamma)$
  - $:= [\![\Gamma; (x : A) \vdash B]\!](\gamma, p)^{[\![\Gamma \vdash A]\!](\gamma)}$
    (when $A, B$ are both Propositional term)
  - $:= \bigsqcap \{[\![\Gamma; (x : A) \vdash B]\!](\alpha, \gamma) | \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}$
    (when $B$ is Propositional term)
  - $:= \prod_{\alpha \in [\![\Gamma \vdash A]\!]'(\gamma)} [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha)$
    (Otherwise)

## Soundness

### Theorem

*Following conditions hold:*

- $[\![\Gamma \vdash \bot]\!] = \phi$
- $[\![\Gamma \vdash A \wedge B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcap ([\![\Gamma \vdash B]\!](\gamma))$
- $[\![\Gamma \vdash A \vee B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcup ([\![\Gamma \vdash B]\!](\gamma))$
- $[\![\Gamma \vdash \exists x : A.Q]\!](\gamma) = \bigsqcup_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (x : A) \vdash Q]\!](\gamma, \alpha)$

### Theorem (Soundness)

- *If $\Gamma \vdash t : T$ is derivable, then $[\![\Gamma \vdash t]\!](\gamma) \in [\![\Gamma \vdash T]\!](\gamma)$ for all $\gamma \in [\![\Gamma]\!]$.*
- *If $\Gamma \vdash t_1 : T$ and $t_1 \to_\beta t_2$, then $[\![\Gamma \vdash t_1]\!](\gamma) = [\![\Gamma \vdash t_2]\!](\gamma)$ for all $\gamma \in [\![\Gamma]\!]$.*
  - *If $P$ is provable propositional term for $\Gamma$, then $p \in [\![\Gamma \vdash P]\!](\gamma)$.*

## Application

- Let $(X, \mathcal{O}(X)) = (1, \{0, 1\})$, and $p = 0$.
  - This is just the Classical Model.
- Let $(X, \mathcal{O}(X)) = (2, \{0, 1, 2\}), p = 1$.
  - If $P$ is true, then $1 \in [\![\vdash P]\!] = 2$.
  - $1 \notin [\![\vdash \forall P : \mathrm{Prop}, P \vee \neg P]\!]$ since $[\![\vdash \forall P : \mathrm{Prop}, P \vee \neg P]\!] = 1$.

## The reason of point condition

The reason of the condition "$\bigcap \mathcal{N}(p)$ is open set" is ...

- This condition is needed when proving the soundness.
- Let $P$ be a propositional term.
  - If $P$ is true, then $p \in [\![\Gamma \vdash P]\!](\gamma)$ should hold.

## The reason of point condition

The reason of the condition "$\bigcap \mathcal{N}(p)$ is open set" is ...

- This condition is needed when proving the soundness.
- Let $P$ be a propositional term.
  - If $P$ is true, then $p \in [\![\Gamma \vdash P]\!](\gamma)$ should hold.
  - The condition that

$$p \in [\![\Gamma; (x : A) \vdash P]\!](\gamma, \alpha) \Rightarrow p \in [\![\Gamma \vdash \forall x : A.P]\!](\gamma)$$

is needed.

## The reason of point condition

The reason of the condition "$\bigcap \mathcal{N}(p)$ is open set" is ...

- This condition is needed when proving the soundness.
- Let $P$ be a propositional term.
  - If $P$ is true, then $p \in [\![\Gamma \vdash P]\!](\gamma)$ should hold.
  - The condition that

$$p \in [\![\Gamma; (x : A) \vdash P]\!](\gamma, \alpha) \Rightarrow p \in [\![\Gamma \vdash \forall x : A.P]\!](\gamma)$$

is needed. However, it does not hold,

## The reason of point condition

The reason of the condition "$\bigcap \mathcal{N}(p)$ is open set" is ...

- This condition is needed when proving the soundness.
- Let $P$ be a propositional term.
  - If $P$ is true, then $p \in [\![\Gamma \vdash P]\!](\gamma)$ should hold.
  - The condition that

$$p \in [\![\Gamma; (x : A) \vdash P]\!](\gamma, \alpha) \Rightarrow p \in [\![\Gamma \vdash \forall x : A.P]\!](\gamma)$$

is needed. However, it does not hold, since

$$[\![\Gamma \vdash \forall x : A.P]\!](\gamma) = (\bigcap \{[\![\Gamma; (x : A) \vdash P]\!](\gamma, \alpha) | \alpha \in [\![\Gamma \vdash A]\!](\gamma)\})^{\circ}.$$

holds.

## Paradox

J.Reynolds showed following theorem.

**Theorem**

*If $P$ is a propositional term, $2 \leqq \sharp[\![\Gamma \vdash P]\!](\gamma)$ causes paradox.*

But we can avoid the paradox for the following reasons.

- Proof terms are interpreted to just one element in this model.
- Make the different interpretations of $\forall x : A.B$ for the case whether $A$ and $B$ are propositional terms or not.

## Question

- Remove the point condition.
  - Is the point condition really needed in proving the soundness?
  - Is there any counterexamples?
- Prove the completeness of this model.
  - For any topological space $(X, \mathcal{O}(X))$, $p(\in X)$ with point condition, if $P$ is propositional term, $p \in [\![\Gamma \vdash P]\!](\gamma)$ implies $P$ is provable propositional term?
    - If $\Gamma \vdash p_1 : P$ and $\Gamma \vdash p_2 : P$ where $P$ is Propositional term, then $[\![\Gamma \vdash p_1]\!](\gamma) = [\![\Gamma \vdash p_2]\!](\gamma)$
    - $\Gamma \vdash P_1 \leftrightarrow P_2$ implies $[\![\Gamma \vdash P_1]\!](\gamma) = [\![\Gamma \vdash P_2]\!](\gamma)$.
  - How close to completeness is this model?



## Reference


[1] Werner, Benjamin. "Sets in types, types in sets." Theoretical aspects of computer software. Springer Berlin Heidelberg, 1997.

[2] Miquel, Alexandre and Werner, Benjamin. "The not so simple proof-irrelevant model of CC." Types for proofs and programs. Springer Berlin Heidelberg, 2003. 240-258.

[3] Reynolds, John C. "Polymorphism is not set-theoretic." Springer Berlin Heidelberg, 1984.

[4] Pierce, Benjamin C. "Types and programming languages." MIT press, 2002.

[5] Barendregt, Henk, W. Dekkers, and RICHARD Statman. "Lambda calculus with types." Handbook of logic in computer science 2 (1992): 118-310.

[6] Coquand, Thierry, and Gerard Huet. "The calculus of constructions." Information and computation 76.2 (1988): 95-120.

[7] MacLane, Saunders, and Ieke Moerdijk. "Sheaves in geometry and logic: A first introduction to topos theory." Springer, 1992.

# Preparing formal specifications: uses of semi-formal methods in the concept phase of a system design

**Shunsuke YATABE**
Institute of Mathematics for Industry, Kyushu University, Japan / JST CREST

This is a joint work with Takashi MORI[1], Kenji TAGUCHI, Hideki Nishihara, Daisuke SOUMA[2].

West Japan Railway Company (JR West) and the National Institute of Advanced Industrial Science and Technology (AIST, Japan) are working together to develop radio based train control system named JRTC-W based on JRTC(Japan Radio Train Control System) standard. JRTC is a Japanese Industrial Standard and it is not defined systems specification but requires least requirements of train control system using radio communication. JRTC requires compatibility between various manufactures and other existing systems.

To insure these requirements, the railway operator (JR West) has to prepare various documents like specifications. Needless to say, system development is collaborative work with the operator and manufacturers and therefore the most important thing is to communicate without misunderstandings. However such misunderstandings often occur: we have a lot of seeds of misunderstandings such as logical ambiguity, implicit knowledge, difference of common senses, and difference of word definition. Preventing these seeds of misunderstandings is very important.

It is well-known that to fix formal languages and writing specifications in such formal language prevents these seeds of misunderstandings. However, they are not so useful for the railway operator: the operator concerns the concept phase, the system definition phase and the system requirement phase, but these formal languages are designed to be used in more detailed design phase like Design and implementation phase. The operator deals with very rough and naive concepts which are very difficult to be formalized. Therefore it is easier to write specifications in natural language for the operator.

In this sense, we need some sort of semi-formal methods, which are enough flexible to write rough ideas in the concept phase, and which are enough strict to exclude the possibility of misunderstandings in the later phase. Recently, the needs of such method is increasing day and day: it is getting more important for railway operators to write documents without logical ambiguity because we have to reference international standards such as IEC62278(RAMS) to manage railway signaling equipment development [3]. But these RAMS requirements are not easy to understand for non-English speakers. To help our understanding for RAMS, we do want to clarify objectives, inputs, outputs and requirements for each RAMS phase by using semi-formal methods.

[1]Train Control Systems Team, Technical Research and Development Department, Railway Operations HQ, West Japan Railway Company, Osaka, Japan

[2]Research Institute for Secure Systems, National Institute of Advanced Industrial Science and technology, Japan

In this talk, we introduce a few such methods, as meta-models, GSN (Goal Structuring Notation) chart for each phase and relationship chart for each requirement in RAMS [1], and traceability information models [2] to make sure the traceability among these RAMS documents. These charts help us to determine effectives for inputs and outputs between requirements in RAMS.

## References


[1] Kelly, T. Arguing Safety- A Systematic Approach to Safety Case Management *Dphil thesis*, Dept. Computer Science, U. of York, 1998.

[2] P. Mader, P. L. Jones, Y. Zhang, J. Cleland-Huang, Strategic Traceability for Safety-Critical Projects, *IEEE Software*, pp58-66, 2013

[3] IEC 62278: Ed.1.0:2002/EN 50126:1999, Railway applications — specification and demonstration of reliability, availability, maintainability and safety (RAMS)

# On formalization of basic geometric topology

Ken'ichi Kuga

joint work with

Manabu Hagiwara

Department of Mathematics and Informatics, Chiba University, Japan

Theorem Proving and Provers Meeting, December 5, 2014





## Contents







## Jordan Curve Theorem

The Jordan Curve Theorem asserts any simple closed curve in the plane separetes the interior from the outside region.
In more mathematical terms:

**Theorem (Jordan Curve Theorem)**

*Let $S^1 := \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 = 1\}$, the unit circle, and if $f : S^1 \to \mathbb{R}^2$ is a homeomrphism onto its image $f(S^1)$, then $\mathbb{R}^2 - f(S^1)$ consists of two arcwise connected components.*





## Jordan Curve Theorem

Although the Jordan curve theorem states seemingly obvious geometric facts, a mathematically rigourous proof may not be easy to create. The first proof of this theorem was obtained by C. Jordan and published in 1887. Many mathematicians thought, however, Jordan's proof was not quite satisfactory. Some mathematicians regard the proof given later by O. Veblen as the first mathematically rigorous proof.

The first formal proof of the Jordan Curve Theorem was created by T.C.Hales using the HOL Light proof system in January 2005:

Hales, Thomas C. "The Jordan curve theorem, formally and informally", The American Mathematical Monthly 114(10):882-894(2007)

## Jordan-Brouwer Separation Theorem

The higher-dimensional analogue of the Jordan curve theorem is called Jordan-Brower Separation Theorem, which was obtained independently by H. Lebesgue and L.E.J. Brouwer in 1911.

**Theorem**

*Let $n$ be a positive integer, $S^{n-1} := \{(x_1, \ldots, x_n) \in \mathbb{R}^n | x_1^2 + \cdots + x_n^2 = 1\}$, the $(n-1)$-dimensional sphere, and if $f : S^{n-1} \to \mathbb{R}^n$ is a homeomrphism onto its image $f(S^{n-1})$, then $\mathbb{R}^n - f(S^{n-1})$ consists of two arcwise connected components.*

The traditional proofs use basic machinery of singular homology theory, which doesn't seem to be formalized yet.





## Schoenflies Theorem (in dimension 2)

The Scheonflies Theorem claims that the interior region bounded by the simple closed curve in the plane is always a topological disk.
In other words:

**Theorem (Schenflies Theorem (in dimension 2))**

*Let $S^1 := \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 = 1\}$, the unit circle, and if $f : S^1 \to \mathbb{R}^2$ is a homeomrphism onto its image $f(S^1)$, then there is a homeomorphism of the plane $F : \mathbb{R}^2 \to \mathbb{R}^2$ whose restriction to $S^1 \subset \mathbb{R}^2$ is $f$: $F(p) = f(p)\ \forall p \in S^1$.*

This is not an easy therem. The original proof given by A. M. Schoenflies in 1906 seems to have some mathmatical gaps. L.E.J.Brouwer gave a complete proof in 1910. Caratheodory's theorem in complex analysis also proves Schoenflies theorem.





## Schoenflies Problem in dimensions greater than 2

Although, it might seem that higher-dimensional analogue of the 2-dimensional Schoenflies therem should surely hold, this is not the case:

**Theorem (Counter Example to Higher Dimensional Schoenflies Problem)**

*There are topological emmbeddings $f : S^2 \to \mathbb{R}^3$ such that there exist no homeomorphisms $F : \mathbb{R}^3 \to \mathbb{R}^3$ whose restriction to $S^2 \subset \mathbb{R}^3$ is $f$: $F(p) = f(p)\ \forall p \in S^2$.*

There are many "pathological" counter examples to the higher-dimensional Schoenflies problem including the **Alexander's horned sphere** shown in the next picture.





## Alexander's horned sphere

– This picture is taken from the book "Knots and Links" by D. Rolfsen, Publish or Perish (1976) –

## Generalized Schoenflies Theorem

If we assume embedded topological spheres are "topologically flat", then the higher-dimensional Schoenflies Theorem holds:

**Theorem (Generalized Schoenflies Theorem)**

*If $f : S^{n-1} \times (-1,1) \to \mathbb{R}^n$ is a homeomrphism onto its image $f(S^1) \times (-1,1)$, then there is a homeomorphism of the plane $F : \mathbb{R}^2 \to \mathbb{R}^2$ whose restriction to $S^1 \subset \mathbb{R}^2$ is $f$: $F(p) = f(p)\ \forall p \in S^1$.*

This theorem was first proved by Morton Brown in 1960
M. Brown: "A proof of the generalized Schoenflies theorem", Bull. Amer. Math. Soc. Vol 66, No. 2 (1960), 74-76.
We will show our formalization of The Bing Shrinking criterion which is a basic tool to make homeomorphisms in geometric topology.
R.H.Bing: "A homeomorphism between the 3-sphere and the sum of two horned spheres", Ann. of Math. (2) 56 (1952), 354-362.
The M. Brown's proof of the generealized Schoenflies theorem shares the same idea.





## Bing Shrinking Criterion

```
Hypothesis X_compact: compact Xt.
Hypothesis Y_compact: compact Yt.

Definition Bing_shrinkable (f:X→Y): Prop:=
  ∀ eps:R, eps>0 →
    ∃ h : point_set Xt → point_set Xt,
     homeomorφsm h ∧
    (∀ x:X, d' (f x) (f (h x)) < eps) ∧
    (∀ x1 x2:X,  (f x1) = (f x2) → d (h x1) (h x2)  < eps).
```





## Bing Shrinking Theorem

```
Definition approximable_by_homeos (f:X→Y): Prop:=
  ∀ eps:R, eps>0 →
    ∃ h:point_set Xt → point_set Yt,
     homeomorφsm h ∧
    (∀ x:X, d' (f x) (h x) < eps).

Theorem Bing_Shrinking_Theorem:
 ∀ f: point_set Xt → point_set Yt,
continuous f → surjective f →
 (Bing_shrinkable f → approximable_by_homeos f).
```





## Formal Proof

We formalized the proof of the Bing Shrinking Theorem following the line of argument of R.D.Edwards' ICM talk in 1978. (This part was added just after the TPP 2014 meeting held in December 5. Our formalization was completed in December 14, 2014.)

(Our formalization uses The Coq/Ssreflect proof system. We used Daniel Schepler's Topology library in coq user-contributions http://www.lix.polytechnique.fr/coq/pylons/contribs/ besides the Coq standard libraries.)

```
Let CMap :=
  { f:X→Y | bound (Im Full_set
              (fun x:X⇒ d' (y0 x) (f x)))∧
              @continuous Xt Yt f }.
```

With uniform metric, CMap becomes a complete metric space.

## Formal Proof

Suppose $\mathrm{f}\colon \mathrm{CMap}$ satisfies the Bing shrinking criterion.

```
set fH : Ensemble (point_set CMapt) :=
  fun gP: CMap ⇒ ∃ hx: point_set Xt → point_set Xt,
              homeomorϕsm hx ∧
              ∀ x: point_set Xt, (proj1_sig gP) x = f (hx x).

set CfH := closure fH.
set CfHt := SubspaceTopology CfH.
```

Then by applying the Baire Category Theorem, $\mathrm{CfHt}$ is a baire space:

```
have CfHt_baire: baire_space CfHt.
apply BaireCategoryTheorem
          with um_restriction um_restriction_metric.
```





## Formal Proof

```
Let W (f: CMap) (eps:R):
 Ensemble (point_set CMapt) :=
 fun g:CMap ⇒  ∀ (x1 x2:X),
  (proj1_sig g x1) = (proj1_sig g x2) → d x1 x2 < eps.

Lemma W_is_open: ∀ (f: CMap) (eps:R),
                        eps > 0 → open (W f eps).
```

Bing shrinking criterion amounts to saying each such $\mathrm{W}$ is dense.

```
set Wn: IndexedFamily nat (point_set CfHt) := fun n:nat ⇒
   inverse_image (subspace_inc CfH)  (W fP (/INR (S n))).
have WnOD: ∀ n:nat, open (Wn n) ∧ dense (Wn n).
```

Then applying the baire property, intersection of $W_n'$s is dense.

```
have IWn_dense: dense (IndexedIntersection Wn).
apply CfHt_baire.
by apply WnOD.
```

This intersection consists of the desired homeomorphisms.





## Baire Spaces

As we needed Baire Category Theorem in our formalization of the Bing Shrinking Theorem, we formalized Baire Category Theory for compact metric spaces.

```
(* Definition of Baire Spaces *)
Definition baire_space : Prop :=
 ∀ V : IndexedFamily nat (point_set X),
  (∀ n: nat, (open (V n)) ∧ (dense (V n))) →
   dense (IndexedIntersection V).
```





## Baire Spaces

```
(* The Baire Category Theorem for complete metric spaces *)
Theorem BaireCategoryTheorem :
                  complete d d_metric → baire_space.

(* We need to assume the Axiom of Choice
                              to show this theorem*)
Axiom FDC : FunctionalDependentChoice_on
             (point_set X * { r:R | r > 0} * nat).
```

## Casson Handles are Topological Handle Theorem

**Theorem (Casson Handles are Topological Handles)**

*$\forall CH$ : Casson handle, $CH$ is homeomorphic to the standard 2-handle: $CH \approx D^2 \times \mathbb{R}^2$*

This is a key theorem in 4-dimensional topology proved in 1981 by M. Freedman:
M.H.Freedman " The topology of four-dimensional manifolds" Journal of Differential Geometry 1982.





## The last step of Freedman's Theorem

We are hoping to formalize its proof. One reason is that, just like Feit-Thompson Theorem in finite group theory, M. Freedman's proof is so complicated and exotic, even specialists in low-dimensional topology might not want to go over the details.

The proof of Lemma 9.2 is summarized in the following figure:

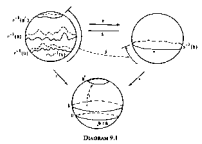


Diagram 9.1

In fact the last step is completed by applying a generalized version of Schoenflies theorem to a Casson Handle placed in the 4-dimensional sphere. This Shoenflies theorem produces the desired homeomorphism by the Bing shrinking method.

# Formalizing a coding theory

**Takefumi SAIKAWA**
Nagoya University

We are working on formalizing the core properties of Reed-Solomon codes. The presentation of the theory tightly follows the one given in Hagiwara's textbook[3]. As this is an unfinished work with almost no result yet, we can only present our objecives. First, we are trying to completely understand this basic theory. We shall be able to find every detail and necessary technicalities by formalizing it. After completing this first step, we expect that we can extend our formalization to more advanced coding theories such as BCH codes and Gabidulin codes. Second, we are benchmarking the applicability of the ssreflect/mathcomp library[1] which is provided for the proof assitant coq[2]. The mathcomp library contains many results of algebra, yet it was originally developed to deal with a specific result in group theory. We want to find mismatches between the library and our use, if there is any. And last, we are examining the effectiveness of a method to debug a book by formalizing it. So far our development have successfully pointed out in the textbook[3] the need of a few corrections which are more than typos, i.e., which actually affect the theory. We expect that this method can correct errors more consistently than usual reviewing process. Aside from these objectives, we propose some possible improvements in the graphical interface of coq. We plan to implement them in parallel to the development of our formalization.

## References

[1] `http://ssr.msr-inria.inria.fr/`
[2] `http://coq.inria.fr/`
[3] 萩原学, 「符号理論」, 2012, 日本評論社.

# Formalizing a coding theory

Takafumi Saikawa

December 18, 2014

## about the project

A part of infotheo project (Reynald's talk, yesterday).
Formalizing the final chapter of Hagiwara's book.

## motivation

General motivations for formalizing a theory: we want to

- rigorously understand the theory,
  we need details to really understand a theory
- build a good library, generalizing notations and patterns used in math,
  notations are not trivial
- and debug the textbook.

## motivation

I want to

- get used to ssreflect & mathcomp,
- understand a coding theory from scratch.

## theory

Reed-Solomon code

- is an error-correction code,
- takes code words in a polynomial ring $\mathbb{F}[X]$,
- is powerful and used everywhere.

## presentation in the textbook

Reed-Solomon code in Hagiwara's book:

- Presented mostly in terms of polynomials. Very little linear algebra.
- Employs the Euclidean decoding algorithm.
- Main theorem : the decoder corrects up to $(d-1)/2$ errors, where $d$ is the distance that appears in the definition of RS-code :

$$C(n, a, d) := \{f(X) | \mathrm{deg} f(X) < n \wedge \forall (1 \leq n_0 \leq d-1), f(a^{n_0}) = 0\}$$

## presentation in the textbook

The technical ingredients in the Euclidean decoding algorithm are:

- Euclid's gcd
- derivation

They are already given in `poly.v` and `polydiv.v` of mathcomp library.
It seems to be fairly easy to formalize the theory now...

Not finished yet (by this morning).

## interface

"interface matters!"

We want (I want) an improved ProofGeneral:

- Independent "`Search`" buffer, independent from `proof-response-buffer`.
- Graphical subterm selection by a pointing device.
- (Following the selection of a subterm,) automatic listing of immediately "`rewrite`"-able lemmas.
- Let a mouseover trigger a `Search`, `Locate`, ...
- Intermediate results
  - in a sequence of tactics at each semicolon,
  - and in a sequence of `rewrites` at each lemma,

  i.e. finer step-execution.

Maybe we need several concurrent coqtop processes; possible with much RAM and many CPUs.

## conclusion

work-in-progress; should be commited (hopefully) in some weeks.

# Formalization of geometric algebra with application to computational origami

Tetsuo Ida
University of Tsukuba, Japan





## Motivation and Background

- Recently, I became interested in 3D origami geometry as it has immediate applications in robotics and manufacturing using 3D printers.
- Instead of a fresh new start, I pursue to elaborate the basis of E-origami system *Eos*, that I have developed over the years.
- I have a rather ad hoc 3D model of origami implemented in Eos, and I will refine this 3D model to a fuller and more abstract 3D model.





## Motivating example 1

### Piano origami (intermediate steps omitted)

**Step 12**





## Motivating example 2

### Simulation of flapping crane (figure omitted)

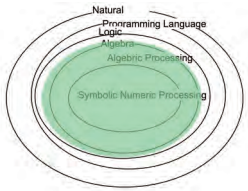
## Current version of Eos is unsatisfactory in that

- Restricted 3D modeling both for verification and visualization
- No theory for the motion





# Abstractions in Eos








## Geometric algebra (GA) - one definition

The geometric algebra (GA) is a Clifford algebra of a vector space over the field of real numbers endowed with quadratic form (Wikipedia).





## Geometric Algebra

### Origin

Grassmann in 19th century, then Clifford's development, and then wide-spread use of vectors, tensors in physics and mathematics (somehow distorted from the originators' idea)

### Period of synthesis and developments

Hestenes (1986), Doran 2003), ...
GA is applied successfully to computer vision and graphics in computer science, physics and mechanics.

## Geometric algebra GA

GA is an algebra R, where

**1.** R is an abelian group under addition (+)

**1.1.** + is associative

**1.2.** There is an element 0 in R such that a + 0 = a and 0 + a = a for all a in R (0 is the additive identity).

**1.3.** For each a in R there exists – a in R such that a + (–a) = (–a) + a = 0 (–a is the additive inverse of a).

**1.4.** + is commutative





**2.** R is equipped with multiplication (*)

**2.1.** * is associative

**2.2.** There is an element 1 in R such that 1 * a = a

**3.** Multiplication distributes over addition





**4.** R is endowed with quadratic form

"Endowment of quadratic form" is a link of vector to scalar. Usually it is realized by:

```
x * x = g (x) ∈ ℝ
```

**where** $g$ **(**$x1$, $x2$, **..) is a quadratic form,**
**e.g.** $g$ **(**$x1$, $x2$, $x3$**)** $= |x1|^2 + |x2|^2 + |x3|^2$





## For GA to be in action

- use the scripts/programs written for GA to be integrated into Eos.
- Isabelle/HOL for formalization and *Mathematica* for the rest of computation

## Geometric algebra applied to computational origami

We restrict to three-dimensional space, since we are mainly interested in physical and mechanical aspects of the objects.

### g3

g3 is a GA consisting of 4 blades (real, vector, bivector, trivector) over three dimensional space.





## GA in Isabelle/HOL

### Definition of g3

```
class g3 =  abelian_group +
fixes times :: "'a ⇒'a  ⇒'a" (infixl  " ⊗"  70)
fixes  quad_form:: "'a ⇒ bool"
and one::"'a"
assumes times_distr: "(x ⊕ y) ⊗ z  = x ⊗ z ⊕  y ⊗ z"
and times_distl: " x ⊗ (y ⊕ z)   =x  ⊗ y ⊕ x ⊗ z"
and times_assoc: " (x ⊗ y) ⊗ z  = x ⊗ (y ⊗ z)"
and times_left_identity: "one ⊗ x =x"
```





## Mvec as type constructor in g3

- g3 is inhabited by objects of datatype mvec consisting of four components, i.e. type scalar, vec, bivec and trivec.
- Datatype mvec is constructed by combining those components with Mvec as a constructor.
- The scalar is linked by the quadratic form axiom, and from i-vec to i+1-vec by single application of $\wedge$.





## Operations that connect to geometrical intuition

### Inner product

### Outer product

## Inner product

```
a · b := (1/2) * ((a * b) + (b * a));
```

connection to geometry by interpreting

```
a · b by | a || b | cos θ
```

when **a** and **b** are vectors

**when a ≠ 0 and b ≠ 0, a · b** = 0 if **a** and **b** are orthogonal.





## Outer product

```
a ∧ b := (1/2) * ((a * b) - (b * a));
```

connection to geometry by

```
| a ∧ b | := | a || b | sin θ
```

Object of the form **a ∧ b** , where **a** and **b** are vectors, is called bivector.

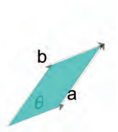
**when a ≠ 0 and b ≠ 0, a ∧ b** = 0 if **a** and **b** are collinear.





## Bivectors (2D)

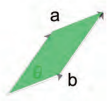
a ∧ b








a ∧ b = - b ∧ a



## Trivector - 3D

**related to volume**

unit trivector e1∧e2∧e3

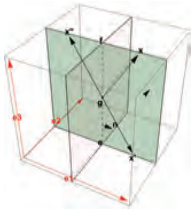




## Reflection

**x'** = - **n** * **x** * **n** (**x'** is the reflection of x along **ef**), where **n** is a unit vector.

x''= - n' * x * n' (x'' is the reflection across ef), where n' is a unit vector n' · n =0)







## Basic fold operations

H. Huzita (1989) : Axiomatic Development of Origami Geometry, Proceedings of the First International Meeting of Origami Science and Technology

His idea is based on the observation:

Fold along the line passing through two (different) points.

Superpose points and lines.





## Huzita's basic fold operations

1. **Given two distinct points, you can fold making the crease pass through both points (`ruler operation`).**
2. **Given two distinct points, you can fold superposing one point onto the other point (`perpendicular bisector`).**
3. **Given two distinct (straight) lines, you can fold superposing one line onto another (`bisector of the angle`).**
4. **Given one line and one point, you can fold making the crease perpendicular to the line and passing through the point (`perpendicular footing`).**
5. **Given one line and two distinct points not on this line, you can fold superposing one point onto the line and making the crease pass through the other point (`tangent from a point to a parabola`).**
6. **Given two distinct points and two distinct lines, you can fold superposing the first point onto the first line and the second point onto the second line at the same time.**

◄ | ►

## Huzita's basic folds restated (only O1 and O6 are shown)

- **(O1) Given two distinct points *P* and *Q* on *O*, we can fold *O* along the fold line that passes through *P* and *Q*.**
- **(O6) Given two distinct points *P* and *Q* on *O* and two distinct fold lines *m* and *n*, passing through *O* , it is decidable whether we can fold *O* to superpose *P* and *m*, and *Q* and *n*, simultaneously.**





## (O1)

### Fold line for O1

A line **x** through the origin is represented by

$$\mathbf{x} \wedge \mathbf{u} = \mathbf{0} \tag{1}$$

Vector **u** determines the slope of the line.

A line **x** through **p** with the slope specified by **u** is

$$(\mathbf{x} - \mathbf{p}) \wedge \mathbf{u} = \mathbf{0} \tag{2}$$

A line **x** through **q** with slope **u** is

$$(\mathbf{x} - \mathbf{q}) \wedge \mathbf{u} = \mathbf{0} \tag{3}$$

From (2) and (3) and using the distribution properties,

$$(\mathbf{q} - \mathbf{p}) \wedge \mathbf{u} = \mathbf{0}, \quad \text{and hence } \mathbf{u} = \alpha\,(\mathbf{q} - \mathbf{p}), \text{ where } \alpha\,(\in \mathbb{R}) \neq \mathbf{0} \tag{4}$$

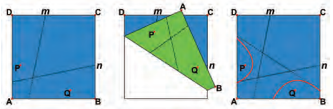
From (2) and (4),

$$(\mathbf{x} - \mathbf{p}) \wedge (\mathbf{q} - \mathbf{p}) = 0$$





## O6








## Fold line for O6

Given p, q, r, s, e and f, such that p ≠ q and r ≠ s

The desired fold line is
$(x - g) \wedge u = 0$

Let $\mathcal{U}e$ and $\mathcal{U}f$ be the reflections (across **u** with g as the origin) of e, and f, respectively.

$\mathcal{U}e = -\hat{u} * (e - g) * \hat{u} + g$
$\mathcal{U}f = -\hat{u} *_M (f - g) * \hat{u} + g$

By the definition of g, we can let $g = \frac{e + \mathcal{U}e}{2}$

The construction stipulates:
$(\mathcal{U}\mathbf{e} - \mathbf{p}) \wedge (\mathbf{q} - \mathbf{p}) = 0$ and $(\mathcal{U}\mathbf{f} - \mathbf{r}) \wedge (\mathbf{s} - \mathbf{r}) = 0$





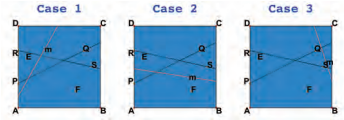
## Cartesian coordinate system





## Eos (O6)

```
HO["E", "PQ", "F", "RS"];
```








## Conclusion

- GA is a good model that lies between logic for geometry and Cartesian-coordinate based algebra.
- GA is *good* in the sense of abstraction and the power to express the geometric relations without explicitly mentioning the dimensionality.
- Its connection to physics and mechanical engineering is obvious advantage, where we can share the rich knowledge in these domains.
- I have not tackled the problem of the motion of the objects.

# Mathematical Components and Algebraic Numbers

**Cyril COHEN**

Inria Sophia Antipolis – Méditerranée, France

The Mathematical Components project spanned 6 years and involved 15 people, with the purpose of formalizing a proof of Feit-Thompson theorem [1] (also known as the odd order theorem), which is a milestone in the classification of finite groups. The proof relies on various areas in mathematics, including group theory, linear algebra, representation theory, Galois theory,... The project succeeded in formalizing the proof thanks to the development of a large set of libraries and methodologies covering the mathematics. One of the main achievements of the project, besides the formalization of the main theorem, is this set of libraries, which were designed to be reused. They were indeed reused in other developments such as, for example, in formalizations on homological algebra, elliptic curves, Shannon's information theory, error correcting codes,... My personal contribution to the Mathematical Components libraries was to build algebraic numbers and the tools to construct and use them. In this talk [2], I will give an overview of the project, its library and I will detail my own contributions.

## References

[1] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. Ould Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, L. Théry. A Machine-Checked Proof of the Odd Order Theorem. In *ITP 2013, 4th Conference on Interactive Theorem Proving*, pages 163–179, 2013.

[2] C. Cohen. `http://www.cyrilcohen.fr/papers/slides_TPP2014.pdf`

# 対話型幾何ソフトウエアと自動証明
# ーシンデレラとキッズシンディ

Interactive geometry software and automated theorem proving
- Cinderella and KidsCindy

阿原一志（明治大学）
Kazushi AHARA (Meiji university)

## 自己紹介

- 明治大学　総合数理学部　先端メディアサイエンス学科の　阿原一志です.
- 専門は位相幾何学に関する研究支援ソフトウエア開発です.
- これまで開発した主なソフトウエアはTeruaki（写像類群ソフト），KidsCindy（動的幾何学ソフト）です.
- 機械証明，グレブナー基底のどちらも専門外です.

## 大まかな内容

- IGS の紹介
- 自動定理証明機能
- シンデレラ・KidsCindyの紹介
- IGSに潜む問題点とその解決
- シンデレラ・KidsCindyにおける自動定理証明機能のアルゴリズム

## IGSとは

- 対話型幾何ソフトウエア（Interactive geometry system)の略です.
- 主にマウス操作によって平面（最近では3Dのものも多い）に作図していく機能を持つソフトウエアです.
- 作図手順が記録されているので，作図をした後から頂点などの幾何要素を動かすことができます（実演します）.

## IGSの特性

- IGS開発における大切なこと
  - 対話的な高校数学教材を作ることができます.
  - 作図後に図を動かすことにより，ユーザーは作図の中に普遍性を発見することができます.
  - 「コンパスと定規による作図問題」への試行錯誤が容易にできます.
  - 平面幾何の自動定理証明により，非自明な関係性を見つけるのに役立ちます.

## 主なIGS

- コレテンカンプ氏（シンデレラの作者）が数年前に数えたところによると100種類以上のIGSがあるそうです.
- 初期にはカブリとシンデレラが主流でした．阿原の近辺ではgeogebra，KSEGなどがありました.
- カブリは高価でしたが，日本に一定数のユーザーがいました．その後，シンデレラの日本語版作成のお手伝いをしましたが，日本での反応はいまひとつでした.

## KidsCindyを作った理由

- 日本語版シンデレラの普及が進まなかったのは，高校教員に紹介する機会があまり得られなかったためなのですが，当時は「有償ソフトなのがいけないのかもしれない」と考えていました.
- そのことから似た機能を持つフリーソフトを作ってみることにしたのがKidsCindyです.
- そこで障壁になったのが「自動定理証明機能」です.

## 自動定理証明機能

- 多くのIGSには自動定理証明機能が搭載されています.（実演します.）
- シンデレラはソースコードを公開しておらず，どのような方法で自動定理証明をしているかはわかりません.
- シンデレラの本には「十分いろいろな場合を試してみればその正しさがわかる」というような記述はあったのですが，数学的な裏付けは書かれておらず，どのようなアルゴリズムであるかはわかりません.（実は今でも知りません.）

# コンピュータ幾何学

- 「コンピュータ幾何学」という本を出版いたしました.（数学書房，2014年）
- 前半では，IGSにおける計算アルゴリズムについて紹介しています.（後半は別の話題について説明しています.）
- IGSにおける自動定理証明のアルゴリズムについても解説しています．今日はその内容をご紹介します.

# Wuの方法1

- 古典的には「Wuの方法」がよく知られています.
- 自由に動かせる点の座標を変数 $u_1, u_2, \ldots$ で表し，自由な点から作図によって求められる点の座標を変数$x_1, x_2, \ldots$であらわし，その関係式を

$$f_1\,(x_1, u_1, u_2, \ldots) = 0,$$
$$f_2\,(x_1, x_2, u_1, u_2, \ldots) = 0,$$
$$f_3\,(x_1, x_2, x_3, u_1, u_2, \ldots) = 0, \ldots$$

- で表します.
- 結論となる命題を

$$g(u_1, u_2, \ldots x_1, x_2, \ldots) = 0$$

の形に表現します.

# Wuの方法2

- 目標は

$$Q[u_1, u_2, \ldots, x_1, x_2, \ldots]/(f_1, f_2, \ldots)$$

のなかで，$g(u_1, u_2, \ldots, x_1, x_2, \ldots)$が0と等しいことを証明することです.
- もし，$x_1, x_2, \ldots$が

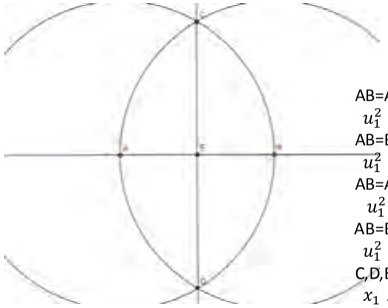
$$x_1 = h_1(u_1, u_2, \ldots),$$
$$x_2 = h_2(x_1, u_1, u_2, \ldots),$$
$$x_3 = h_3(x_1, x_2, u_1, u_2, \ldots), \ldots$$

という形で得られるのであれば，これらを順に$g$に代入すればよいわけですが，多くの作図ではそのような形で得られません.

# Wuの方法3

- たとえば，変数が$x_1, u_1, u_2$に限られているとして，$f_1$が
$h_1(u_1, u_2)x_1 + h_2(u_1, u_2) = 0$ --(*)
の形で得られていると仮定しましょう.
- $g(x_1, u_1, u_2)$の$x_1$を 消去するために，
$h_1(u_1, u_2)^s g(x_1, u_1, u_2)$ ---(**)
を計算して(ここで，$s$は$g(x_1, u_1, u_2)$における$x_1$の次数)，この式(**)に式(*)を代入することにより$x_1$を消去することができます.（擬剰余）
- すると，
$h_1(u_1, u_2)^s g(x_1, u_1, u_2) \equiv \exists g_0(u_1, u_2) \bmod. (f_1, f_2, \ldots)$
とあらわせます.
- 計算の結果，$g_0 = 0$であったとしましょう．すると$h_1 = 0$ または $g = 0$ が想定されます．前者は「図が崩れてしまう場合」，後者は「定理が成立」という形の結論を得ることができます.

# Wuの方法（実例）1



$A(0,0), B(u_1,0),$
$C(x_1,x_2), D(x_3,x_4), E(x_5,x_6)$

AB=AC
$u_1^2 - x_1^2 - x_2^2 = 0$
AB=BC
$u_1^2 - (x_1 - u_1)^2 - x_2^2 = 0$
AB=AD
$u_1^2 - x_3^2 - x_4^2 = 0$
AB=BD
$u_1^2 - (x_3 - u_1)^2 - x_4^2 = 0$
C,D,Eは同一直線上
$x_1\ x_4 + x_2x_5 + x_3x_6$
$-x_1x_6 - x_2x_3 - x_4x_5 = 0$
A,B,Cは同一直線上
$u_1x_6 = 0$

証明したい性質：AE=EB
$g = x_5^2 + x_6^2 - (x_5 - u_1)^2 - x_6^2$

# Wuの方法（実例）2

AB=AC
(1) $u_1^2 - x_1^2 - x_2^2 = 0$
AB=BC
(2) $u_1^2 - (x_1 - u_1)^2 - x_2^2 = 0$
AB=AD
(3) $u_1^2 - x_3^2 - x_4^2 = 0$
AB=BD
(4) $u_1^2 - (x_3 - u_1)^2 - x_4^2 = 0$
C,D,Eは同一直線上
(5) $x_1\ x_4 + x_2x_5 + x_3x_6$
$-x_1x_6 - x_2x_3 - x_4x_5 = 0$
A,B,Cは同一直線上
(6) $u_1x_6 = 0$

$f_1$ =(2)-(1)
$= 2x_1u_1 - u_1^2$
$f_2$ =(1)
$= u_1^2 - x_1^2 - x_2^2$
$f_3$ =(4)-(3)
$= 2x_3u_1 - u_1^2$
$f_4$ =(3)
$= u_1^2 - x_3^2 - x_4^2$
$f_5 = u\_1(5) - (x_3 - x_1)(6)$
$= u_1(x_1x_4 + x_2x_5$
$-x_2x_3 - x_4x_5)$
$f_6$ =(6)
$= u_1x_6$

# Wuの方法（実例）3

- $g = x_5^2 + x_6^2 - (x_5 - u_1)^2 - x_6^2$
  から始めて，$x_6, x_5, \ldots, x_1$の順番に文字を消去していく．その結果
  $$4u_1^3(x_2 - x_4)\ g(u_1, x_1, \ldots, x_6) \equiv 0$$
  $$mod.\ (f_1, f_2, \ldots, f_6)$$
  を示すことができる．
- $u_1 = 0$⇔A=B,
  $x_2 - x_4 = 0$⇔C=D
  であることを考えれば，A≠B,C ≠Dの仮定のもとにAE=EBが示されることがわかる．

# シンデレラはWuの方法を使っていません．

- シンデレラを説明した本で，著者（かつ，シンデレラの作者）であるコルテンカンプは自動定理証明機能について次のように言っています．
- Cinderella does not use symbolic methods to create a formal proof, but a technique called ‘Randomized Theorem Checking.’ First the conjecture “It seems that line *k* always passes through point *K*” is generated. Then the configuration is moved into many different positions and for each of these it is checked whether the conjecture still holds. It may sound ridiculous, but generating enough (!) random(!) examples where the theorem holds is at least as convincing as a computer-generated symbolic proof.

## 阿原から見たコルテンカンプの思想

- 「点Kは直線k上にある」という命題の証明をするという前提で考えます.
- 自由に動かせる点の座標を変数$x_1, x_2, \ldots$とし，従属的に決まる点の座標を変数を$u_1, u_2, \ldots$で表すことにすると，「点Kが直線k上にある」という関係性は$x_1, x_2, \ldots$の式（一般には根号を含むような式）で表現できます.
- もし$x_1, x_2, \ldots$に「十分に多くの個数のランダムな値を代入してみた結果，式の値が0になる」のならば，その式はそもそも0と等しいと結論できると考えます.

## この思想の根本的欠点１

- 「ランダムな値を代入してみた結果，式の値が0になる」ということをコンピュータのソフトウエアで実現しようとすると必ず計算誤差が発生する.「0になる」ことをわれわれが判定するためには「十分に0に近い（閾値の内側である）」という考え方をして判定する必要があり，数学的な厳密さを失ってしまう.

## この思想の根本的欠点２

- この思想に基づいて幾何の定理証明を試みるためには，作図によりきめられる「点」や「直線」が自由に動かせる要素の位置にかかわらず大域的に決定できなければならない. しかしそのようなことは不可能である.（依存幾何要素の決定不可能性をコルテンカンプ自身が論文の中で証明している.）
- 「円と直線の交点」「円と円の交点」「2直線の角の2等分線」は大域決定不可能である.

## IGSの静的問題

- 「作図手順を記述することにより，図は一意的に定まるか」という問題が静的問題である. 答えはNOであるが，作図を行う数学的枠組みを拡張することにより，多くの場合は解決可能である.
- コルテンカンプは射影平面$P^2$を用いることにより，無限大の問題（たとえば，「2直線の交点」というモジュールにおいて，平行な2直線が入力された場合にどのように出力するか）を解決した.

## IGSの動的問題

- 「自由に動かせる要素を連続的に動かした場合，作図全体も連続的に動かせるか」という問題が動的問題である．これも答えはNOである．しかし作図平面を複素射影平面($P^2(C)$)にすることにより，作図要素を代数多様体の上で考えることへと帰着し，その多くは解決することがコルテンカンプにより指摘された．

## KidsCindyにおける自動定理証明

- 思想としてはシンデレラに準ずる．
- 自由に動かせる幾何要素について「上下左右の微動」を独立に行い，それぞれの場合について作図を実行してみて，「点Kが直線k上に誤差付で載っているかどうか」を判定する．
- すべての微動について判定がOKであれば，数学の定理として非常に確からしいと判断する．

## KidsCindyにおける自動定理証明の有効性

- 平面幾何作図による定理を相当数準備して，KidsCindy が定理判定できるかどうかを確かめたところ（調査した範囲では）100パーセント正しく判定した．（正17角形のリッチモンドによる作図も含む．）
- 作図手順の個数から想定される累積誤差を想定した閾値を設定したため，このような結果が得られたと考えられる．（作図手順が10000～100000といった定理がもしあれば，判定に失敗することもありうる．一方で，IGSではこのような巨大作図は想定していない．）

## KidsCindyにおける自動定理証明の穴

- KidsCindyでは「正しくないものを正しいと判定してしまう」可能性も残っている．
- たとえば正17角形の作図においてそれは発生しうる．$\cos(2\pi/17)$の値を十分に近似する作図であれば，真の値でなくともKidsCindyは「正しい」と判定する．ただし，これは可能性を指摘しただけであって，実際に検証してみたわけではない．

# TPPmark2014

## 1 Question

Let $\mathbf{N} = \{0, 1, 2, 3, \cdots\}$ be the set of all natural numbers, $p \in \mathbf{N}$ and $q \in \mathbf{N}$. We denote $(p \,\mathbf{mod}\, q) = r$ if and only if there exist $k \in \mathbf{N}$ and $r \in \mathbf{N}$ such that $p = kq + r$ and $0 \leqq r < q$. Further, we denote $(q \mid p)$ if and only if $(p \,\mathbf{mod}\, q) = 0$. Prove the following questions:

(i) For any $a \in \mathbf{N}$, $(a^2 \,\mathbf{mod}\, 3) = 0$ or $(a^2 \,\mathbf{mod}\, 3) = 1$.

(ii) Let $a \in \mathbf{N}$, $b \in \mathbf{N}$ and $c \in \mathbf{N}$. If $a^2 + b^2 = 3c^2$ then $(3 \mid a)$, $(3 \mid b)$ and $(3 \mid c)$.

(iii) Let $a \in \mathbf{N}$, $b \in \mathbf{N}$ and $c \in \mathbf{N}$. If $a^2 + b^2 = 3c^2$ then $a = b = c = 0$.

## 2 List of Authors

All answers are stored in the github repository.
Cf. `https://github.com/KyushuUniversityMathematics/TPP2014/wiki`

1. Adam Chlipala (Coq (290 lines))
2. Cyril Cohen (Ssreflect (35 lines))
3. Daisuke Sato (Coq (250 lines))
4. Fadoua Ghourabi (Isabelle/HOL (282 lines))
5. Jaques Garrigues (Coq (190 lines))
6. Kazuhiko Sakaguchi (Ssreflect (55 lines))
7. Kazuhisa Nakasho (Mizar (175 lines))
8. Kei Tsujimoto (HOL Light (58,175,39 lines))
9. Keishi Suda (Ssreflect (55 lines))
10. Masahiro Sakai (Agda (528 lines))
11. Masahiro Sato (Coq (1153 lines))
12. Mitsuharu Yamamoto (Ssreflect (44 lines), HOL4 (69 lines))
13. Sosuke Moriguchi (Coq (503 lines))
14. Takashi Miyamoto (Coq (239 lines))
15. Yoichi Hirai (Ssreflect (64 lines))

# 3 Answers

1. **Author:** Adam Chlipala
   **System:** Coq 8.4pl4
   **Abstract:** I decided, somewhat arbitrarily, to only use those modules of Coq's standard library that I'm already familiar with, which turned out to encompass only facts that the tactics [omega] or [ring] could prove.
   I also specialized modular arithmeic to modulus 3.
   Finally, I'm going for high automation, especially in the proofs that are specified to the challenge problems and go beyond basic modular-arithmetic facts.

2. **Author:** Cyril Cohen
   **System:** Coq 8.4pl4 + ssreflect 1.5.0 + mathcomp 1.5.0
   **Abstract:** We prove Lemma 1, Lemma 2 (without using Lemma 1) and Lemma 3 (using Lemma 2, and not Lemma 1) directly, using the mathcomp libraries (mostly ssrnat, div and prime). The formal proofs follow closely the following informal proofs:
   Lemma 1 is a simple case analysis over the three possible values that a %% 3 can take, so we first introduce the proof that a %% 3 < 3 and we do three cases on a %% 3.
   For Lemma2 it suffices to show that a and b are divisible by 3. Indeed, then 3 divides $c^2$ and by Euclid's lemma, 3 divides c. Then, since $a^2 + b^2 = 0$ modulo 3, we prove by case analysis on (a %% 3) and (b %% 3) that the only possible values they can take is 0. (We do not use Lemma 1 as it seems shorter to redo directly the case analysis we did in the proof of Lemma 1).
   In Lemma 3, we begin with a strong induction on c (i.e. we prove the statement for c = 0 and then we prove it for any c smaller than n+1, with the induction hypothesis saying the statement is true for any c smaller than n). Indeed:

   - When c is 0, then $a^2 + b^2$ is 0, which means a is 0 and b is 0.
   - If c is smaller than n+1, we show that $(a / 3)^2 + (b / 3)^2 = 3 (c / 3)^2$, and apply the induction hypothesis to this identity.

   NB: The induction scheme is generated on the fly using the ssreflect elim command.

3. **Author:** Daisuke Sato
   **System:** Coq

4. **Author:** Fadoua Ghourabi
   **System:** Isabell/HOL
   **Abstract:** I use Isar (Wenzel, 1999), which is an extension of Isabelle/HOL, in an attempt to write structured proofs. The proofs use rules from Isabelle/HOL theories for semi-ring and ordering. In the proofs of (i) and (ii), I apply algebraic simplifications that are allowed in a semi-ring. In order to prove (iii), I first show that c = 0, then the rest is straightforward. I assume that c is not null. I define a set S = c. c > o /\ Exists a b. a^2+b^2 = 3c^2 and obtain its minimal element

cmin since relation < is well-founded over the natural numbers. Furthermore, using (ii), I prove the intermediate lemma asserting that if aˆ2+bˆ2=3cˆ2 then (a div 3)ˆ2 + (b div 3)ˆ2 = 3*(c div 3)ˆ2. Hence, (cmim div 3) is in S which leads to a contradiction ((cmim div 3) < cmim).

5. **Author:** Jacques Garrigue
   **System:** Coq-8.4pl3
   **Abstract:** This is a rather dumb proof written in plain Coq.

6. **Author:** Kazuhiko Sakaguchi
   **System:** Coq 8.4pl4, Ssreflect 1.5, MathComp 1.5
   **Abstract:**
   `https://github.com/pi8027/tppmark/blob/master/2014/solution.v`

7. **Author:** Kazuhisa Nakasho
   **System:** Mizar 8.1.03-5.25.1218
   **Abstract:** Those proofs are written in Mizar language.
   The followings are the proof sketches:
   EX1) 3 cases (a mod 3 = 0 or 1 or 2) are checked.
   EX2) 4 cases (aa mod 3 = 0 or 1) x (bb mod 3 = 0 or 1) are checked.
   EX3) Based on minimal counterexample technique. If c0 is assumed to be a minimum natural number that satisfies a0a0 + b0b0 = 3(c0c0), then there exists a natural number c1 = c0/3 that satisfies the equivalent by EX2. That is contradiction.
   A html-based Mizar proof can be obtained at the same repository. Please download the html file and open on your local computer. If you want to know the meaning of any terms, you can click them to jump to their definitions.

8. **Author:** Kei Tsujimoto
   **System:** HOL Light (r205)
   **Abstract:**
   20141115.ml : Simple proofs in “set_goal” and “expand” style.
   20141124.ml : Alternative proofs in “prove” style.
   20141130.ml : Towards automated theorem proving with Prover9.

9. **Author:** Keishi Suda
   **System:** Coq 8.4pl4, Ssreflect 1.5, MathComp 1.5
   **Abstract:** I tried to construct simple proof.

10. **Author:** Masahiro Sakai
    **System:**

    - Agda-2.4.2.1
    - agda-stdlib-0.9

    **Abstract:** Straightforward proofs using Agda and Agda standard library. We use ≡-Reasoning module to stylize equational reasoning, and use SemiringSolver module to avoid tedious reasoning where possible. In the proof for problem (iii),

well-founded induction on ($\mathbb{N}$, <) is used to show that "$\forall$ b c. $a^2 + b^2 = 3c^2 \rightarrow a = 0$" holds for all a$\in \mathbb{N}$.
See also Qiita post
(`http://qiita.com/masahiro_sakai/items/ecd83bad7ac8425f22a3`
, in Japanese).

11. **Author:** Masahiro Sato
**System:** Coq-8.4pl4
**Abstract:** I proved the problems in legacy coq, not using ssreflect. First, I proved four basic propositions, whose names are 'div_S', 'div_le', 'mod_plus' and 'mod_le'. Then, I prove the problems.

12. **Author:** Mitsuharu Yamamoto
**System:** [SSReflect Proof]
Coq-8.4pl5 + SSReflect 1.5 + MathComp 1.5
[HOL4 Proof]
HOL4, Kananaskis-10
**Abstract:**
[Coq Proof]
The lemma "forall a:nat, sqrmod3P a" is equivalent to "forall (a:nat) (b:bool), b = 3 %| a /\ m = $a^2$ %% 3 < − > (b = true /\ m = 0) \/ (b = false /\ m = 1)". But the (Co)Inductive form is more convenient to work with the case tactic.
[HOL4 Proof]
Actually, I mainly used the HOL system in 1990's (HOL90 at that time), and I'm not so familiar with its recent progress. So this proof may look rather awkward.

13. **Author:** Sosuke Moriguchi
**System:** Coq-8.4pl4
**Abstract:** No use of omega tactic. Question 3 can be proven by well-founded induction (or contradiction), but in this proof, I do not use the lemma for well-founded induction. (Sorry, but in this version, I mistook the assertion of question 2. The lemma to proof the correct assertion from the mistook assertion is already proven, so please pardon me :-) )

14. **Author:** Takashi Miyamoto
**System:** Coq. Used libraries are `Arith Ring Omega Wf_nat`.
**Abstract:**
induction principle for `mod3` calculation : assert

```
forall n, P n /\ P (S n) /\ P (S(S n))
```

and use it.

```
Theorem nat_ind_3 : forall P: nat -> Prop,
P 0 -> P 1 -> P 2 -> (forall n, P n -> P (S(S(S n)))) ->
forall n, P n.
```

define `mod3` function (also `div3`)

```
Fixpoint mod3(n:nat):nat :=
match n with
| S(S(S n')) => mod3 n'
| _ => n
end.
```

lemma for case analysis

```
Lemma mod3_012 : forall n,
  (mod3 n = 0) \/ (mod3 n = 1) \/ (mod3 n = 2).
```

operation on `mod3`

```
Lemma mod3_add_hom: forall p q, mod3 (mod3 p + mod3 q) = mod3 (p + q).
Lemma mod3_mul_hom: forall p q, mod3 (mod3 p * mod3 q) = mod3 (p * q).
```

lemma for `n/3` exists if `mod3 n = 0` using `div3` function.

```
Lemma mod3_0_q : forall n, mod3 n = 0 -> exists q, n = 3*q.
```

problem 1 : rewrite `mod3 (a*a)` with `mod3 a`, then case analysis.

```
Theorem mod3_square_01 : forall a, (mod3 (a*a) = 0) \/ (mod3 (a*a) = 1).
```

problem 2 : same as problem 1

```
Theorem all_mod3_0 : forall a b c, a*a + b*b = 3*c*c ->
  (mod3 a = 0) /\ (mod3 b = 0) /\ (mod3 c = 0).
```

lemma for shrinking `a`, `b`, and `c` : use result of problem 2, `mod3_0_q`, and `ring` and `omega` tactics.

```
Lemma shrink_by_3 : forall a b c, a*a + b*b = 3*c*c ->
  (exists a' b' c', a = 3*a' /\ b = 3*b' /\ c = 3*c' /\
  a'*a' + b'*b' = 3*c'*c').
```

lemma for infinite descent : by using `lt_wf_nat`

```
Theorem infinite_descent : forall P : nat -> Prop,
  (forall n : nat, P n -> exists2 m : nat, m < n & P m) ->
  forall n : nat, ~ P n.
```

conditions for `a`, `b`, and `c` : apply `infinite_descent` and use `shrink_by_3`

```
Lemma cond_of_a: forall a, ~( ~(a=0) /\ (exists b c:nat, a*a+b*b = 3*c*c)).
Lemma cond_of_b: forall b, ~( ~(b=0) /\ (exists a c:nat, a*a+b*b = 3*c*c)).
Lemma cond_of_c: forall c, ~( ~(c=0) /\ (exists a b:nat, a*a+b*b = 3*c*c)).
```

problem 3 : combine these 3 lemmas

```
Theorem abc_are_0 : forall a b c:nat, a*a+b*b = 3*c*c ->
  (a = 0 /\ b = 0 /\ c = 0).
```

15. **Author:** Yoichi Hirai
    **System:** Ssreflect 1.5
    **Abstract:** An attempt to do brute-force.

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」(H.15-19 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」(H.20-24 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。

平成 26 年 10 月
マス・フォア・インダストリ研究所
所長　福本康秀

# 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)

発　行　2015年3月6日
編　集　溝口佳寛 , Jacques Garrigue, 萩原学 , Reynald Affeldt
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 http://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<br>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<br>Masato KIMURA<br>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<br>Masato KIMURA<br>Tatsuyuki NAKAKI | Proceedings of Czech-Japanese Seminar in Applied Mathematics 2006 209pages | October 12, 2007 |
| COE Lecture Note Vol.7 | 若山　正人<br>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,<br>Kyushu University | Consortium “MATH for INDUSTRY” First Forum 87pages | September 16, 2008 |
| COE Lecture Note Vol.13 | 九州大学大学院<br>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」<br>— 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,<br>Tohru Tsujikawa<br>Shigetoshi Yazaki | Proceedings of Czech-Japanese Seminar in Applied Mathematics 2008<br>77pages | February 12, 2009 |
| COE Lecture Note Vol.15 | Faculty of Mathematics,<br>Kyushu University | International Workshop on Verified Computations and Related Topics<br>129pages | February 23, 2009 |
| COE Lecture Note Vol.16 | Alexander Samokhin | Volume Integral Equation Method in Problems of Mathematical Physics<br>50pages | February 24, 2009 |
| COE Lecture Note Vol.17 | 矢嶋　　徹<br>及川　正行<br>梶原　健司<br>辻　　英一<br>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<br>—Special Lecture in Functional Mathematics—　23pages | April 30, 2009 |
| COE Lecture Note Vol.20 | Yasuhide Fukumoto<br>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,<br>Kyushu University | Forum “Math-for-Industry”<br>Casimir Force, Casimir Operators and the Riemann Hypothesis<br>95pages | November 9, 2009 |
| COE Lecture Note Vol.22 | Masakazu Suzuki<br>Hoon Hong<br>Hirokazu Anai<br>Chee Yap<br>Yousuke Sato<br>Hiroshi Yoshida | The Joint Conference of ASCM 2009 and MACIS 2009:<br>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<br>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<br>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<br>数理学研究院 | Forum “Math-for-Industry” and Study Group Workshop<br>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<br>LECTURES AT KYUSHU UNIVERSITY 2010　62pages | November 26, 2010 |
| COE Lecture Note Vol.29 | 木田　雅成<br>原田　昌晃<br>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<br>松井　　卓<br>廣島　文生 | Mathematical Quantum Field Theory and Renormalization Theory<br>201pages | January 31, 2011 |
| COE Lecture Note Vol.31 | 若山　正人<br>福本　康秀<br>高木　　剛<br>山本　昌宏 | Study Group Workshop 2010 Lecture & Report　128pages | February 8, 2011 |
| COE Lecture Note Vol.32 | Institute of Mathematics for Industry,<br>Kyushu University | Forum “Math-for-Industry” 2011<br>“TSUNAMI-Mathematical Modelling”<br>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<br>福本　康秀<br>高木　　剛<br>山本　昌宏 | Study Group Workshop 2011 Lecture & Report　140pages | October 27, 2011 |
| COE Lecture Note Vol.34 | Adrian Muntean<br>Vladimír Chalupecký | Homogenization Method and Multiscale Modeling　72pages | October 28, 2011 |
| COE Lecture Note Vol.35 | 横山　俊一<br>夫　　紀恵<br>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<br>Masato Kimura<br>Shigetoshi Yazaki | Proceedings of Czech-Japanese Seminar in Applied Mathematics 2010<br>107pages | January 27, 2012 |
| COE Lecture Note Vol.37 | 若山　正人<br>高木　　剛<br>Kirill Morozov<br>平岡　裕章<br>木村　正人<br>白井　朋之<br>西井　龍映<br>栄　伸一郎<br>穴井　宏和<br>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” ～<br>154pages | February 20, 2012 |

# シリーズ既刊

| Issue | Author／Editor | Title | Published |
|---|---|---|---|
| COE Lecture Note Vol.38 | Fumio Hiroshima<br>Itaru Sasaki<br>Herbert Spohn<br>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<br>太田　泰広<br>筧　　三郎<br>梶原　健司<br>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<br>“Information Recovery and Discovery”　91pages | October 22, 2012 |
| COE Lecture Note Vol.42 | 佐伯　　修<br>若山　正人<br>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<br>“Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)”<br>121pages | February 19, 2013 |
| COE Lecture Note Vol.46 | 西井　龍映<br>栄　伸一郎<br>岡田　勘三<br>落合　啓之<br>小磯　深幸<br>斎藤　新悟<br>白井　朋之 | 科学・技術の研究課題への数学アプローチ<br>―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<br>脇　　隼人<br>平坂　　貢<br>谷口　哲至<br>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<br>小原　功任<br>濱田　龍義<br>横山　俊一<br>穴井　宏和<br>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<br>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<br>Hiroyuki Ochiai<br>Yoshinori Dobashi<br>Yoshihiro Mizoguchi<br>Shizuo Kaji | Symposium MEIS2013:<br>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<br>“The Impact of Applications on Mathematics”　97pages | October 30, 2013 |
| MI Lecture Note Vol.52 | 佐伯　　修<br>岡田　勘三<br>髙木　　剛<br>若山　正人<br>山本　昌宏 | Study Group Workshop 2013 Abstract, Lecture & Report<br>142pages | November 15, 2013 |
| MI Lecture Note Vol.53 | 四方　義啓<br>櫻井　幸一<br>安田　貴徳<br>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<br>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<br>～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<br>Hiroshi Fujiwara | Inverse problems for practice, the present and the future　93pages | January 30, 2014 |
| MI Lecture Note Vol.55 | 栄　伸一郎<br>溝口　佳寛<br>脇　　隼人<br>渋田　敬史 | Study Group Workshop 2013 数学協働プログラム Lecture & Report<br>98pages | February 10, 2014 |
| MI Lecture Note Vol.56 | Yoshihiro Mizoguchi<br>Hayato Waki<br>Takafumi Shibuta<br>Tetsuji Taniguchi<br>Osamu Shimabukuro<br>Makoto Tagami<br>Hirotake Kurihara<br>Shuya Chiba | Hakata Workshop 2014<br>~ 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:<br>“Applications + Practical Conceptualization + Mathematics = fruitful Innovation”　93pages | October 23, 2014 |
| MI Lecture Note Vol.58 | 安生健一<br>落合啓之 | Symposium MEIS2014:<br>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<br>岡田　勘三<br>梶原　健司<br>髙木　　剛<br>若山　正人<br>脇　　隼人<br>山本　昌宏 | Study Group Workshop 2014 数学協働プログラム<br>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）<br>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 |

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 http://www.imi.kyushu-u.ac.jp/