高信頼なソフトウェアの効率的な開発のための形式手法技術

Formal methods for reliable and efficient software development

本位田真一 Shinichi HONIDEN　田辺良則 Yoshinori TANABE　石川冬樹 Fuyuki ISHIKAWA　姜帆 Fan JIANG　小林努 Tsutomu KOBAYASHI

何がわかる？

これらの研究によって、我々の身の周りの至る所で動いているソフトウェアの安全性が、ひいては現代社会の安全性が向上します。また、ソフトウェアの開発において、間違いによって以前に行った手順のやり直しをすることが少なくなり、効率的な開発が可能になります。

どんな研究？

高信頼なソフトウェアを効率的に開発するための、数学に基づいた技術に関連する研究を行っています。具体的には、定理の証明による分散処理アプリケーションの挙動の正しさの保証や、自然言語の仕様を厳密な言語に落としこんで性質の検証を可能にする研究を行っています。

要素間の依存関係の分析による自然言語仕様からの Event-B 仕様の導出　（小林・石川）

背景・問題

Event-B での仕様検証

仕様を形式的な言語で記述

↓

段階的に詳細化をしながら検証

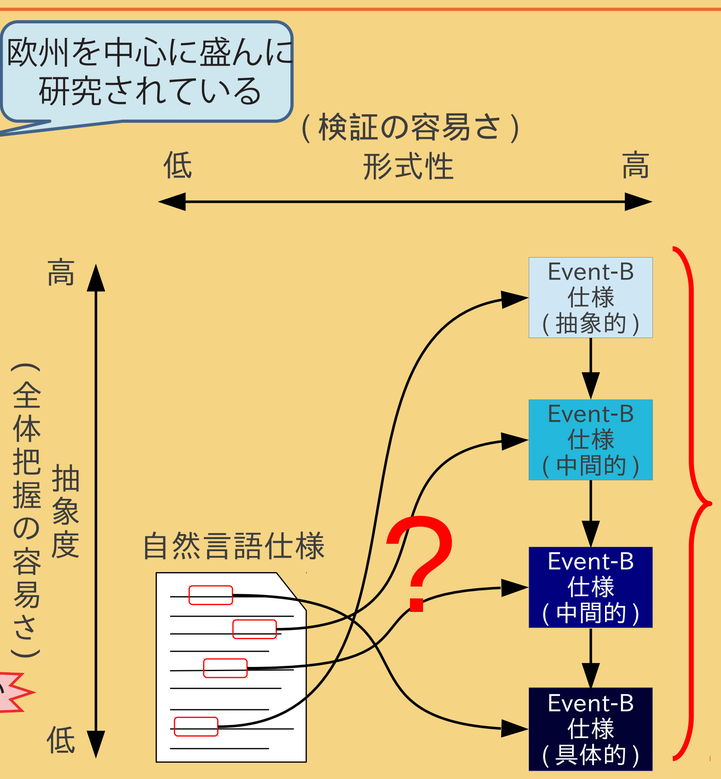
欧州を中心に盛んに研究されている

問題

詳細化の計画を
- 各段階で矛盾がない
- 各段階間で整合性を持つ
- 検証できる程度の粒度を持つ

ように行う必要がある

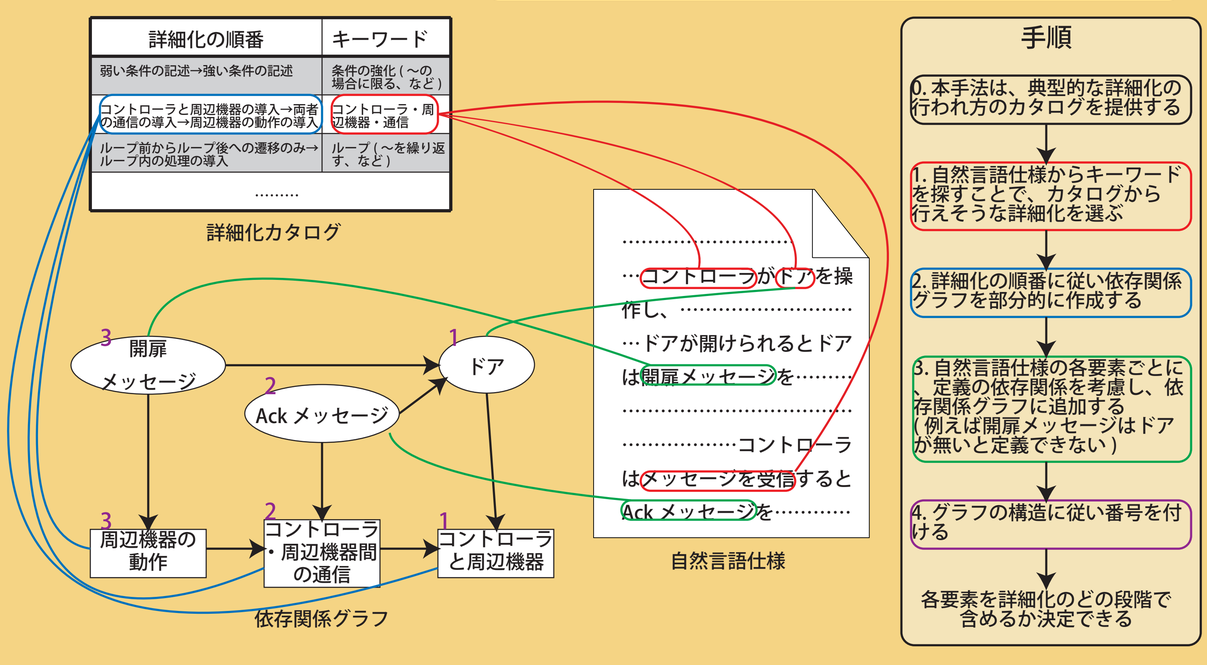
難しい



Event-B を用いる際は詳細化の各段階で導入される要素を正しく決める必要がある

アプローチ

アイデア：
要素 e に依存している要素 f は e より後に導入される必要がある
→　要素間の依存関係は導入段階の決定に重要



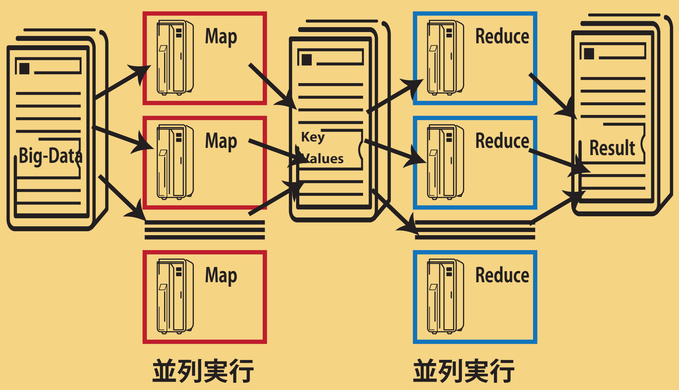
自然言語仕様を詳細化の典型例に当てはめた結果と各要素との依存関係から決定

定理証明による MapReduce アプリケーションの検証とコード生成　（姜・田辺）

背景・問題

MapReduce とは

MapReduce は大規模並列処理に適したソフトウェアパラダイムであり、
Hadoop MapReduce はそのオープンソース実装である



適切にそれぞれ同一の Map 処理と、同一の Reduce 処理を定義して分散アプリケーションを構築

問題

これらの Map 処理の定義と Reduce 処理の定義でアプリケーションが正しく記述されているかをどうやって検証するのか

アプローチ

定理証明（数学）による MapReduce のモデリング

「Map 処理と Reduce は関数で、MapReduce はこれらの関数の合成」

イメージ：　**MapReduce(** *input* **) ::= integReduce(Sort (integMap(** *input* **)))**

integReduce (x **) ::=** F_1 **(Reduce (**x**))**　（並列合成）

integMap(x **) ::=** F_2 **(Map (**x**))**　（並列合成）

数学的な操作・展開により性質の証明ができる：
- 「MapReduce 全体で**関数** g と同じ処理だろうか？」 ⇒　**アルゴリズムの正しさの検証**
- 「Map 処理ですべての入力に対して関 数 h を適用して出力しているだろうか？」
  ⇒　**計算の要所での正しさの検証**

実行効率の話

Q: Map や Reduce は本当に数学的な「**関数**」でいいの？
A: 実は不都合がある
　もし入力が巨大なデータのとき、
　出力をためるとメモリが足りなくなる
　そうならないように Java のイテレータを扱えて，
　証明もできる記述言語を設計した！

コードの生成

記述言語の設計：Scala と文の対応づけをしている
- **文法を小さく制限**
  - **５種類のみ (next, collect, put, if-then-else, while)**
  - **低学習コスト**
- **演算命令（関数適用）は型のみを制限**
  - **自由度の向上**
- **各命令の操作的意味論を用いた証明が可能**

NII
連絡先：　本位田真一　／ 国立情報学研究所　アーキテクチャ科学研究系　教授
TEL：03-4212-2513　　FAX：03-3556-1916　　Email：honiden@nii.ac.jp