自動プログラム検証を高速化するための要素技術

京都大学
KYOTO UNIVERSITY

ERATO　蓮尾メタ数理システムデザインプロジェクト
ERATO Metamathematics for Systems Design Project
国立情報学研究所 & 科学技術振興機構　National Institute of Informatics & Japan Science and Technology Agency

京都大学 大学院情報学研究科 准教授
NII 客員准教授
ERATO MMSD 京都サイトリーダー

末永 幸平

末永がお話しすること

- **形式検証の概説**
  - バグによって引き起こされた重大事例
  - 形式検証とは?
  - 自分で試せる形式検証器や，それらを採用している企業
- **形式検証のもう少し詳しい話**
  - 形式検証のコア技術である不変条件の話
- **末永の研究**
  - 形式検証を軽量化するための手法２つ

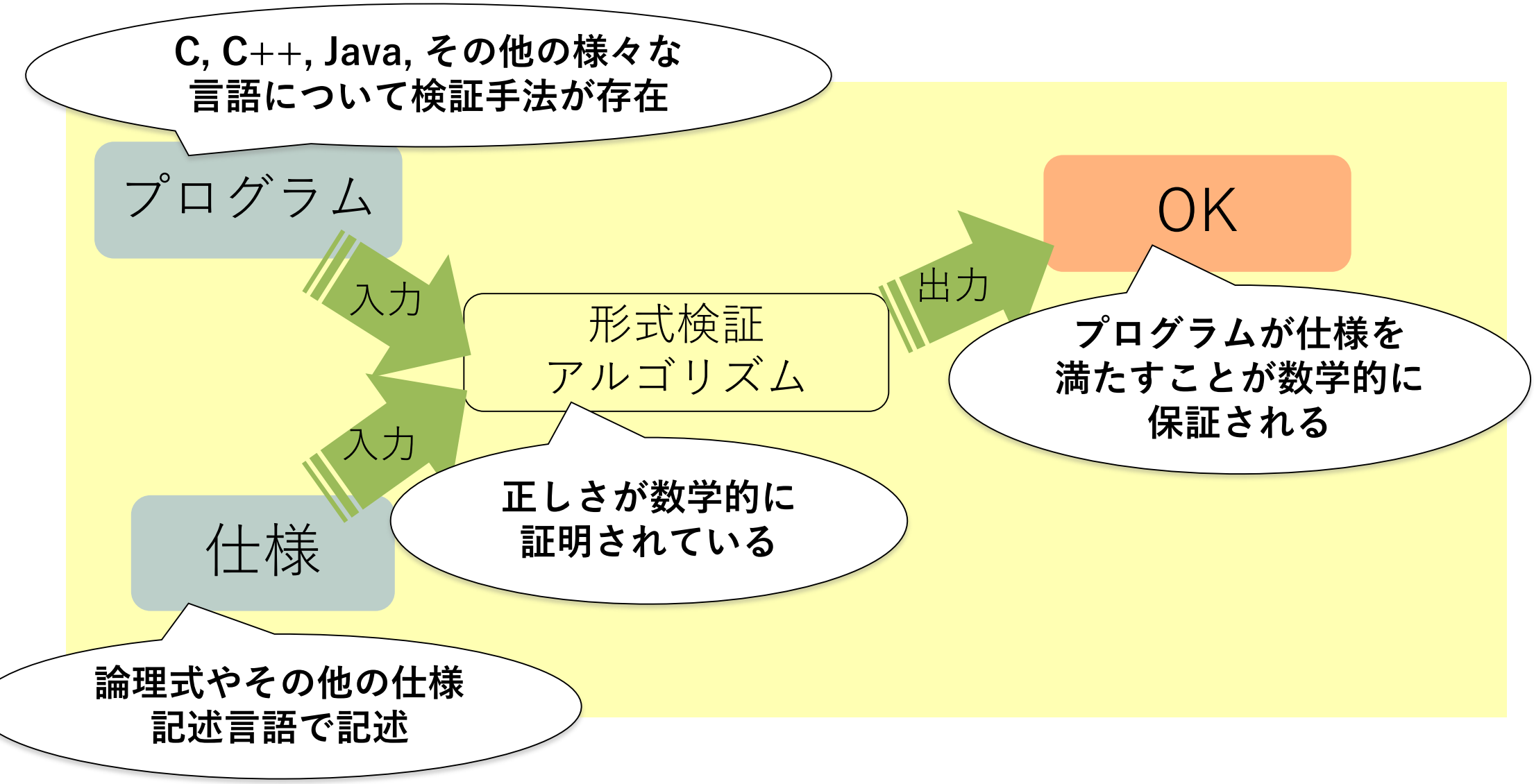
プログラムのバグが損害を引き起こした事例

- **アリアン 5 ロケット爆発事故 (1996)**
  - プログラムの誤りでロケットが爆発
  - 5億ドルの積み荷が失われる
- **放射線治療用機器 Therac-25 事故 (1985～1987)**
  - 意図しない量の放射線を浴びて5人が死亡
- **暗号通信用ソフトウェア OpenSSL のバグ (Heartbleed; 2014)**
  - インターネット上のサーバの 66% に影響
- **暗号通貨 Ethereum の流失事件 (The DAO Attack; 2016)**
  - プログラム（スマートコントラクト）のバグで $170M 相当の仮想通貨が流失

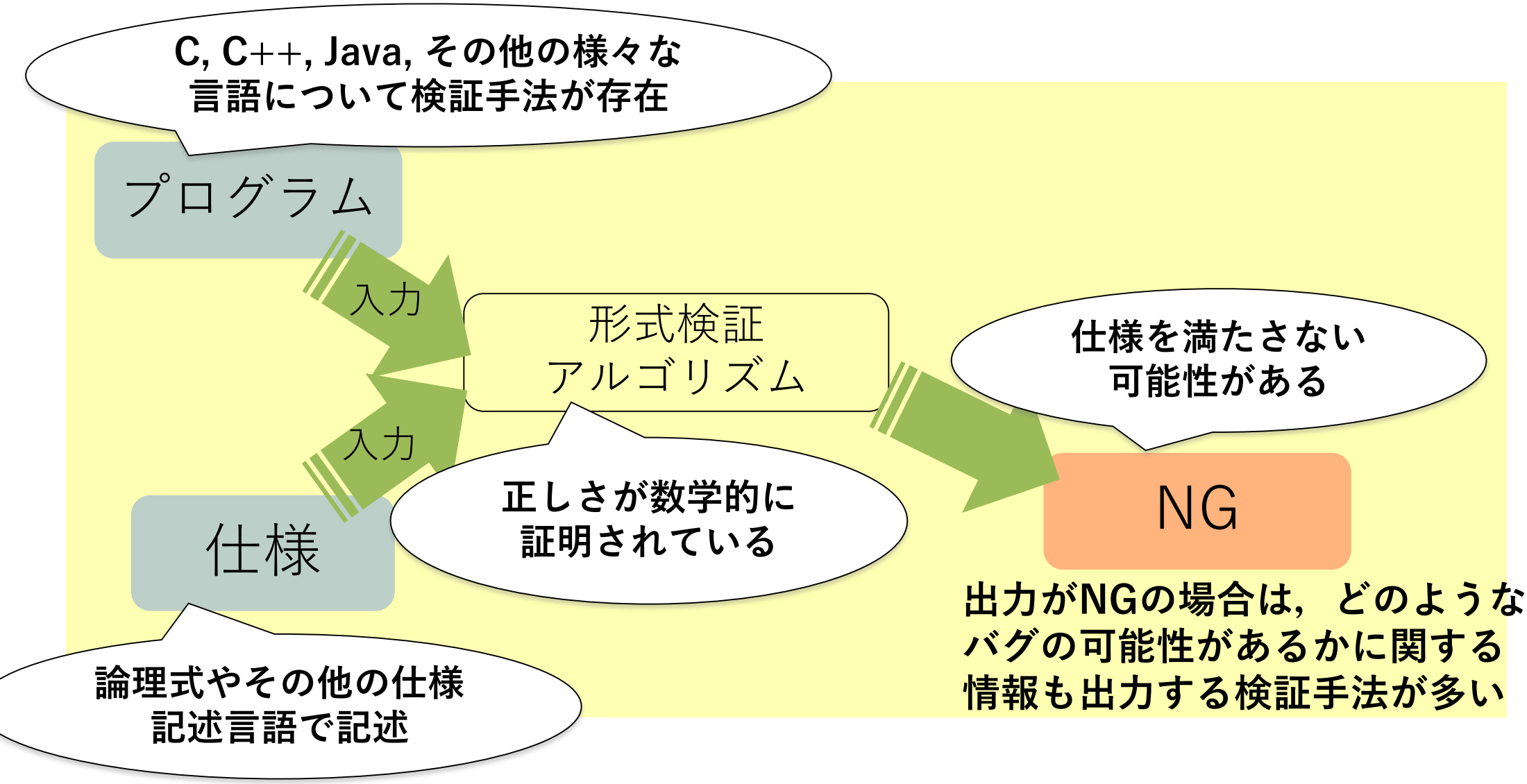
形式検証

- ソフトウェアの正しさを数学的に**証明する手法**
  - 「どんな入力でもプログラムが正しく動作する」という性質を数学の定理として表現し，(できるだけ自動的に) 証明
- 長く研究レベルに留まっていたが，最近実際に使われ始めた

形式検証の概念図


C, C++, Java, その他の様々な
言語について検証手法が存在
プログラム
入力
形式検証
アルゴリズム
出力
OK
プログラムが仕様を
満たすことが数学的に
保証される
入力
正しさが数学的に
証明されている
仕様
論理式やその他の仕様
記述言語で記述

形式検証の概念図


C, C++, Java, その他の様々な
言語について検証手法が存在
プログラム
入力
形式検証
アルゴリズム
仕様を満たさない
可能性がある
入力
正しさが数学的に
証明されている
NG
仕様
出力がNGの場合は，どのような
バグの可能性があるかに関する
情報も出力する検証手法が多い
論理式やその他の仕様
記述言語で記述

形式検証器の例

- **Infer: Facebook 社が開発を進める検証器 [1]**
  - Java, C, C++, Androidアプリ, iOSアプリの検証が可能
  - NullPointerExceptionの可能性やメモリリークの可能性等を検出
  - AWS, Instagram, Mozilla, UBER, Spotify 等が使用
- **Astree: C プログラムのプログラム解析器 [2]**
  - Airbus A340, A380 の制御ソフトウェアや Jules Vernes ATV (宇宙ステーション補給機) の制御ソフトウェアを検証
- **CPAchecker: C プログラムのためのモデル検査器 [3]**
  - モデル検査: プログラムを自動的に抽象化してエラー状態に到達する可能性がないか検査する手法
  - Linux カーネルのバグを多数発見 [4]

[1] Infer: http://fbinfer.com/
[2] The Astree Static Analyzer: http://www.astree.ens.fr/
[3] CPAchecker: The configurable software-verification platform: https://cpachecker.sosy-lab.org/
[4] https://cpachecker.sosy-lab.org/achieve.php

アウトライン

- **形式検証の概説**
  - バグによって引き起こされた重大事例
  - 形式検証とは?
  - 自分で試せる形式検証器や，それらを採用している企業
- **形式検証のもう少し詳しい話**
  - 形式検証のコア技術である不変条件の話
- **末永の研究**
  - 形式検証を軽量化するための手法２つ

形式検証における技術的なチャレンジ

ループや再帰関数呼び出しを含むプログラムの検証

例: 1からn_0-1の和を計算する
プログラム

```
i = 0; sum = 0;
while (i < n0) {
  sum = sum + i;
  i = i + 1;
}
```

ループがどのような計算を行うか
字面上から検証しづらい

仕様: ここでsumが
1からn_0-1までの和に
なっている

不変条件を用いた検証手法

ループの行う計算を「要約」する論理式を使う

例: 1からn_0-1の和を計算する
プログラム

```
i = 0; sum = 0;
while (i < n0) {
  sum = sum + i;
  i = i + 1;
}
```

論理式
$i \leqq n_0 \wedge 2 \times sum - i(i-1) = 0$
はループの実行について不変

仕様: ここでsumが
1からn_0-1までの和に
なっている

不変条件：もう少し詳しく

不変=ループの条件チェックの際に必ず成立

例: 1からn_0-1の和を計算する
プログラム

```
i = 0; sum = 0;
while (i < n0) {
  sum = sum + i;
  i = i + 1;
}
```

$i \leqq n_0 \wedge 2 \times sum\text{-}i(i\text{-}1) = 0$ は
ループが初めて実行されるとき成り立つ

$i \leqq n_0 \wedge 2 \times sum\text{-}i(i\text{-}1) = 0$
が成り立つ状態でループ本体を
一度実行すると再びこの論理式は
成り立つ

仕様: ここでsumが
1からn_0-1までの和
なっている

よって $i \leqq n_0 \wedge 2 \times sum\text{-}i(i\text{-}1) = 0$ は
ループの先頭で常に成り立つ

不変条件を用いると

仕様を証明することができる！

例: 1からn_0-1の和を計算する
プログラム

```
i = 0; sum = 0;
while (i < n0) {
  sum = sum + i;
  i = i + 1;
}
```

論理式
$i \leqq n_0 \wedge 2 \times sum - i(i-1) = 0$
はループの先頭で常に成り立つ

なので，ループの実行が
終了した時点でも成り立つ

仕様: ここでsumが
1からn_0-1までの和
なっている

ループを抜けたということは$i \geqq n_0$が
成り立っているはずなので，不変条件
と合わせてsum=n0(n0-1)/2が成立！

要するに…

良い不変条件を見つけると
形式検証に役立つ！

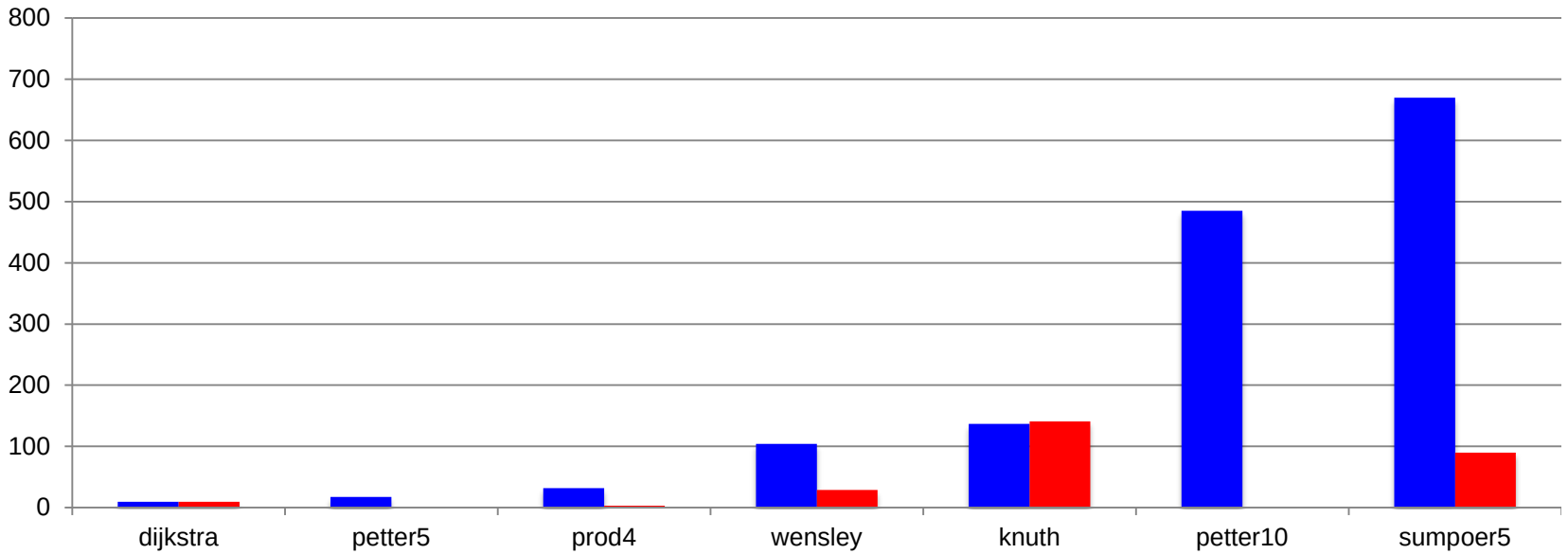
仕様の証明に役立つ不変条件を**自動的に**かつ**高速に**
見つけることができると，形式検証の有用性が高まる

アウトライン

- **形式検証の概説**
  - バグによって引き起こされた重大事例
  - 形式検証とは?
  - 自分で試せる形式検証器や，それらを採用している企業
- **形式検証のもう少し詳しい話**
  - 形式検証のコア技術である不変条件の話
- **末永の研究**
  - 形式検証を軽量化するための手法２つ

研究事例: 高速な不変条件発見手法

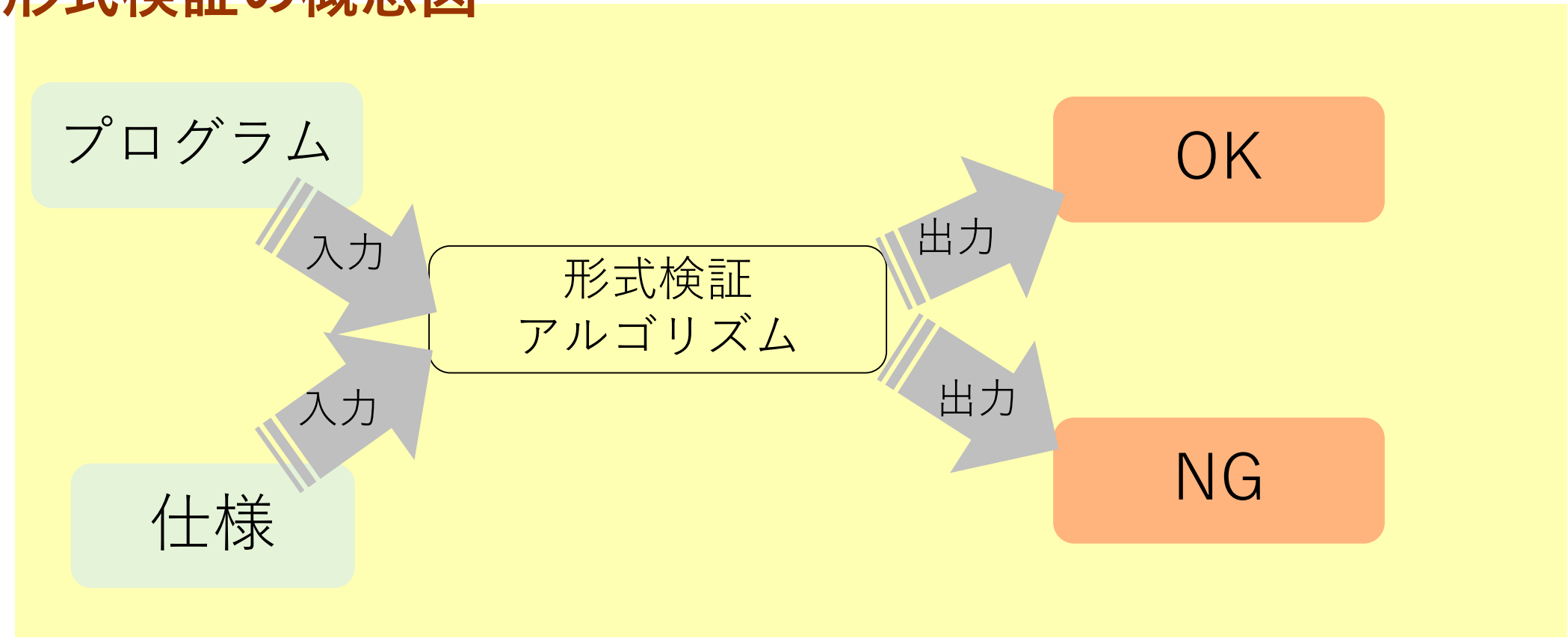
- 不変条件を高速に発見する手法 [SAS'16, TCS'18, 特願2016-017419, 特願2016-017441, PCT/JP2017/003295]
- 物理学における**次元解析**の考え方を応用して従来手法に対し**最大10倍の速度向上**
  - 青: 従来手法 (FastInd); 赤: 本手法



[TCS'18] Kojima, Kinoshita, S: Generalized homogeneous polynomials for efficient template-based nonlinear invariant synthesis. Theor. Comput. Sci. 747: 33-47 (2018)

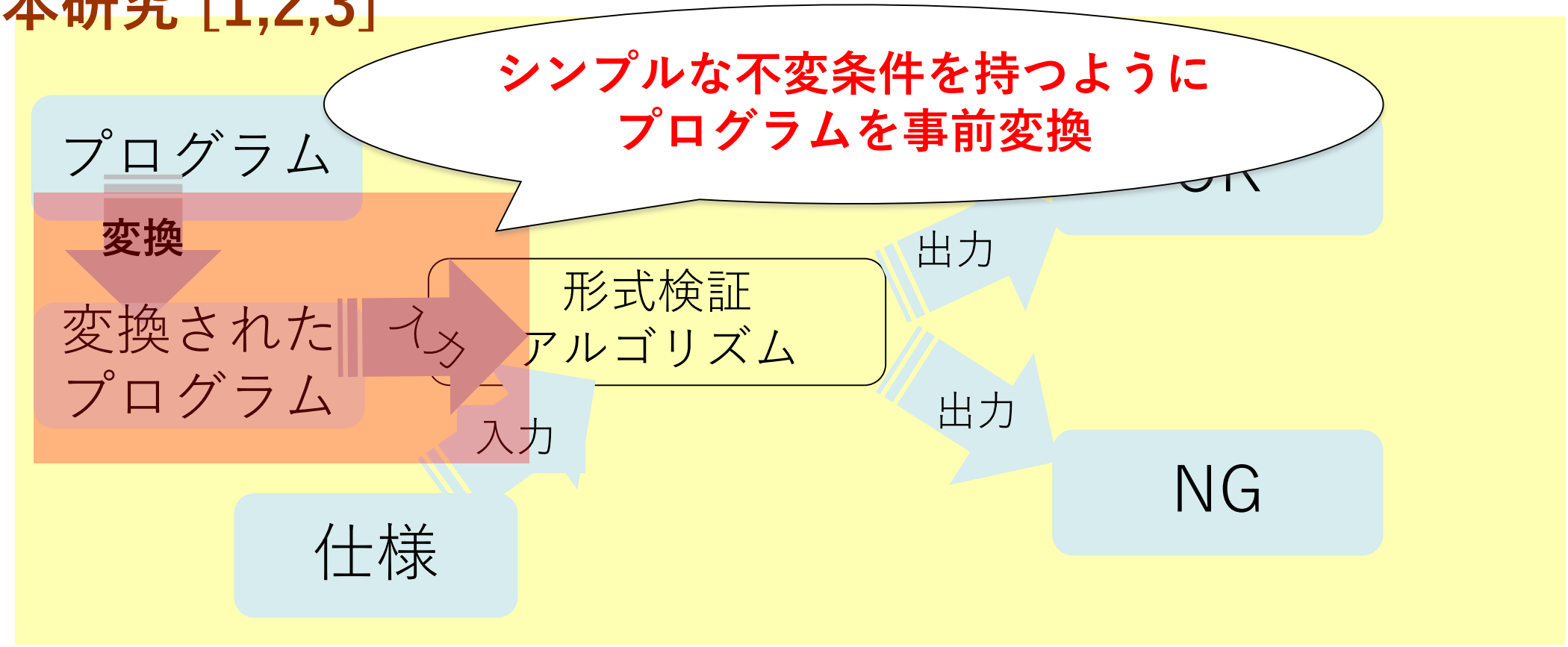
研究事例（末永）: 形式検証を高速化するためのプログラム変換

形式検証の概念図



研究事例 (末永): 形式検証を高速化するためのプログラム変換

本研究 [1,2,3]



[1] A. Imanishi, S, A. Igarashi: A guess-and-assume approach to loop fusion for program verification. PEPM 2018: 2-14
[2] 特願2017-246154 : プログラム検証装置、プログラム検証方法、プログラム検証のためのコンピュータプログラム、プログラム変換器、プログラム変換方法、プログラム変換のためのコンピュータプログラム、プログラム製造方法、及び検証用プログラム． 末永 幸平, 今西 諒文, 五十嵐 淳
[3] PCT/JP2018/044029 : プログラム検証装置、プログラム検証方法、プログラム検証のためのコンピュータプログラム、 プログラム変換器、プログラム変換方 法、プログラム変換のためのコンピュータプログラム、 プログラム製造方法、及び検証用プログラム． 末永 幸平， 今西 諒文， 五十嵐 淳

実験結果

- **いくつかのベンチマークでの CPAChecker の実行時間の比較**
  - CPAChecker が検証できないプログラムを検証可能に
  - 著しい実行時間の悪化は見られず

ベンチマーク名	本手法を適用せず検証 (sec)	本手法を適用して検証 (sec)
loop01	検証不能	7.91
loop02	検証不能	32.64
loop03	検証不能	11.86
loop03_false	5.33	5.79
loop04	検証不能	16.68

[1] A. Imanishi, S, A. Igarashi: A guess-and-assume approach to loop fusion for program verification. PEPM 2018: 2-14
[2] 特願2017-246154 : プログラム検証装置、プログラム検証方法、プログラム検証のためのコンピュータプログラム、プログラム変換器、プログラム変換方法、プログラム変換のためのコンピュータプログラム、プログラム製造方法、及び検証用プログラム． 末永 幸平, 今西 諒文, 五十嵐 淳
[3] PCT/JP2018/044029 : プログラム検証装置、プログラム検証方法、プログラム検証のためのコンピュータプログラム、 プログラム変換器、プログラム変換方 法、プログラム変換のためのコンピュータプログラム、 プログラム製造方法、及び検証用プログラム．末永 幸平，今西 諒文，五十嵐 淳

最近進行中の研究

- **自動運転車のための開発手法と検証手法**
  - 本 ERATO プロジェクトにおける研究
  - 従来の形式検証手法を自動運転車に適用できる形に
- **深層学習のための検証・テスト手法**
  - 学習させたモデルのバグやデータセットの問題を検出・修正するための手法を深層学習を用いている企業と共同研究中
- **ブロックチェーンやスマートコントラクトへの形式検証の適用**
  - 従来の形式検証手法をブロックチェーンやスマートコントラクトに適用する手法を企業と共同研究中

まとめに代えて: 形式検証は実用になるか?

- **ここ数年導入は進みつつある**
  - 有能な検証器の登場 (e.g., infer, CPAChecker)
  - スタートアップを中心とした実践ノウハウの蓄積
- **一方で，プロダクト全体を検証することは困難**
  - 「仕様」を決めることが難しい製品が多い
  - 開発を継続的に進めなければならない状況で，コストの高い形式検証は採用できない
- **テスト等の軽量な品質向上手法との組み合わせが重要になる**
  - 品質保証と保証のコストのバランスを取ることのできる形式検証手法を研究したい