

AI時代のテスト技術

国立情報学研究所 石川 冬樹

<http://research.nii.ac.jp/~f-ishikawa/>



自己紹介

■ソフトウェア工学と先端自律・スマートシステム 特に品質・ディペンダビリティに関する技術

- 形式手法, テスティング, 自己適応,
最適化, 要求分析, 安全性論証など

- サービス合成 (クラウド・IoT), CPS/AI (特に車)

■産業界向け教育・応用研究

- トップエスイー, 日科技連SQiP, 電通大社会人博士

■最近 (1) : 自動 (運転) 車ソフトのテスト



■最近 (2) : 機械学習ベースAIのテスト・品質



QRML



機械学習工学研究会
MACHINE LEARNING SYSTEMS ENGINEERING



おまけ：形式手法の産業応用

■使っている人は使っている

- クラウド内部の複製管理：網羅的な検査，1チームの成功から組織に展開（AWS）
- モバイルアプリ：メモリリークに関する静的解析をリリースサイクル上で自動起動（Facebook）
- 自動運転地下鉄やホームドアのコア部分：正しさの証明済みコードを生成，単体テスト不要に（ClearSy）
- 組み込みチップ：複数組織に展開する外部仕様を厳密化，テスト，仕様に起因する不具合をゼロに（FeliCa）

[C. Newcombe et al., How Amazon Web Services Uses Formal Methods, Comm. of the ACM'15]

[C. Calcagno et al., Moving Fast with Software Verification, NFM'15]

[J. Abrial, Formal Methods in Industry: Achievements, Problems, Future, ICSE'06]

[T. Kurita et al., Practices for Formal Models as Documents: Evolution of VDM application to “Mobile FeliCa” IC Chip Firmware, FM'15]

とはいえ

- 開発対象全体の全問題に対処できるわけではない
- スケーラビリティに懸念
 - 特に，連続系・物理的挙動を扱う場合
- というかUML/Simulinkモデルやコードしかない
 - （厳密に意味論が与えられた）モデルは（最新で）ない
- 今のやり方を変えられない
 - （上流にかかる工数，エンジニアスキル， などなど）
 - 費用対効果には不確かさが多く踏み込めない
 - 移行コストを乗り越えられない

ということで、
私からは「力業」「テスト」の話を



実行しまくって確認する！

「試行&評価」
サイクル！

キレイに言うと

「発見的・経験的アプローチ」

関連動向：

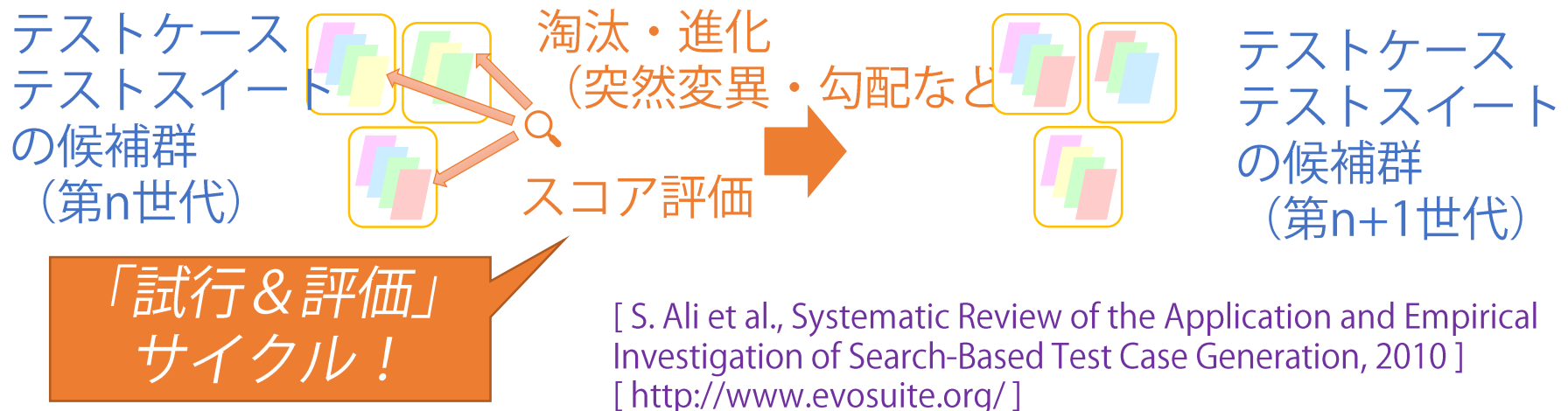
「AIによるテスト」 (の一種)

ソフトウェアテスト分野の一潮流

■サーチベースドテスト（Search-based）

- 最適化技術（特に進化計算・メタヒューリスティック）により「欲しいもの」を表すスコアを最大化するようなテストスイートやテストケースを生成

- 例：「車が人に最も近づくような危ういテスト」ケースを生成
- 例：「前バージョンと比べ出力が大きく変わる」入力を発見
- 例：「高カバレッジで数が小さい」回帰テストスイートを生成



おまけ：サーチベースドテストティングのレベル

■2018年のJava Unit Testing Competition

- 比較用に複数ツールを組み合わせた「最強ツール」は、
人が作ったものよりよいテストスイートを10秒で生成
- ここでの評価基準は、コードカバレッジとミュレーションスコア（人工バグの検出率），テストケース数

■Facebookでの事例

- モバイルアプリのContinuous Integrationに組み込み
- テスト数が少なく，コマンド数が短く，多くのクラッシュを報告するようなテスト一式をまとめあげる
- サーチベースドのバグ自動修正ツールも連動

[<https://github.com/PROSRESEARCHCENTER/junitcontest/blob/master/README.md#6th-junit-contest>]

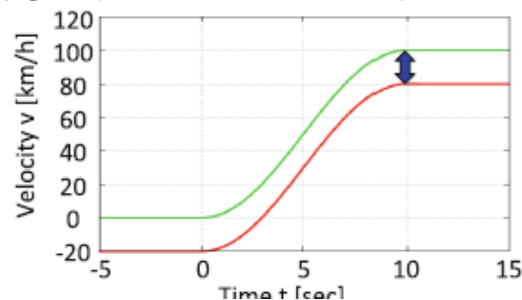
[Molina et al, Java Unit Testing Tool Competition - Sixth Round]

[<https://code.fb.com/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/>]

形式手法分野での一潮流

■ 検証式の定量化

シミュレーション結果
例 (2 サンプル)



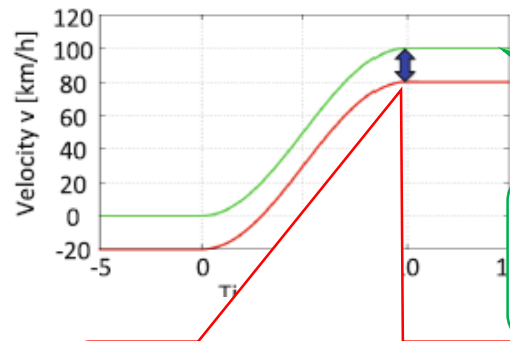
イベントBが起きてから10秒以内に
速度は 80km/h 以上になる

[Fainekos et al., Robustness of temporal logic specifications for continuous-time signals, TheoCompSci'09]
Figure from [Akazaki et al, Time Robustness in MTL and Expressivity in Hybrid System Falsification, CAV'15]

形式手法分野での一潮流

■ 定量検証式の定量的な充足評価

シミュレーション結果
例 (2 サンプル)



イベントBが起きてから10秒以内に
速度は 80km/h 以上になる

OK! 10秒後, 求められたより
も20km/h余裕を持って

ロバストな
充足

OK! 10秒後, ちょうど
ギリギリ求められた値クリア

危うい充足 (違反に近い)

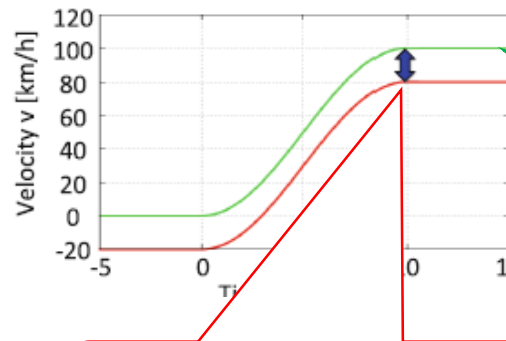
(時間についても同様な議論ができる)

[Fainekos et al., Robustness of temporal logic specifications for continuous-time signals, TheoCompSci'09]
Figure from [Akazaki et al, Time Robustness in MTL and Expressivity in Hybrid System Falsification, CAV'15]

形式手法分野での一潮流

■ 定量検証式に対する最適化を用いた反例探索

シミュレーション結果
例 (2 サンプル)



イベントBが起きてから10秒以内に
速度は 80km/h 以上になる

OK! 10秒後, 求められたより
も20km/h余裕を持って

ロバストな
充足

OK! 10秒後, ちょうど
ギリギリ求められた値クリア

危うい充足 (違反に近い)

(時間についても同様な議論ができる)

「試行&評価」
サイクル!

➡ 「ロバスト度合い」の最適化問題に帰着

[Fainekos et al., Robustness of temporal logic specifications for continuous-time signals, TheoCompSci'09]
Figure from [Akazaki et al, Time Robustness in MTL and Expressivity in Hybrid System Falsification, CAV'15]

AIによるテスト： 自動車関連の研究事例

(ものすごく簡単な) 問題イメージ

Simulinkモデル：
自動ブレーキ付の車の挙動

入力

- ユーザの挙動
- アクセルペダル
 - ブレーキペダル

環境

- 初期速度
- 歩行者の位置・動き
- 路面状況
- ...



出力

- 衝突有無
- 衝突速度

- 実行可能なものがある
(シミュレーションモデルや
コード, 実機)
- 一つの入力を決めて,
出力を得ることはできる

➡ 「危ない」 ケースを見つける

・・・だけではなく何をする？

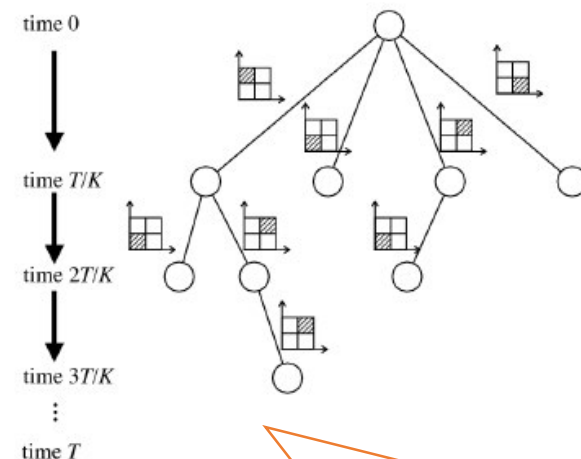
研究事例：より探索的な反例探索

■ 系統的な探索手法を適用

- たまたま見つけた悪い例に深入りしない
- 探索空間全体に対する情報を随時管理・更新

➡ Monte-Carlo Tree Search

- 囲碁プレーヤの構築などで有名なAI技術
- 連続変化する入力信号を分割,
「匂い」「怪しさ」を継続記録し
「試行&評価」サイクル



最初にアクセルを踏み込んだら
どうだろうか？その次は・・・

[Zhang et al., Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search, EMSOFT/TCAD'18]

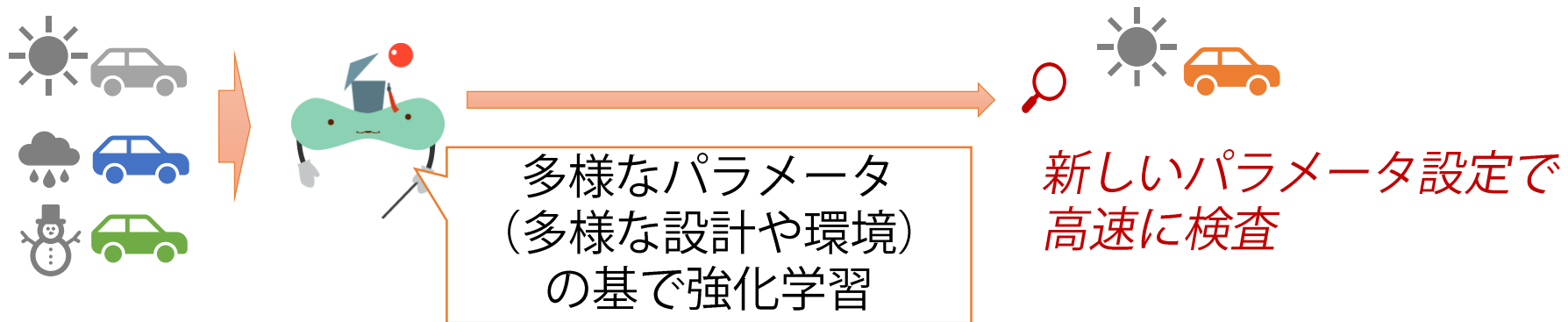
研究事例：事前訓練による反例探索

■探索方法を学習

- 毎回ランダムから探索するのを避ける
- 囲碁プレーヤが様々な相手に対し事前訓練するように

■モデルファミリー式に対する反例生成器の構築

- 試行錯誤の度に行う検査を高速に
- 設計や環境の多少の変化があっても高速に検査



[Kato et al., Falsification of Cyber-Physical Systems with Reinforcement Learning, MT-CPS'18]

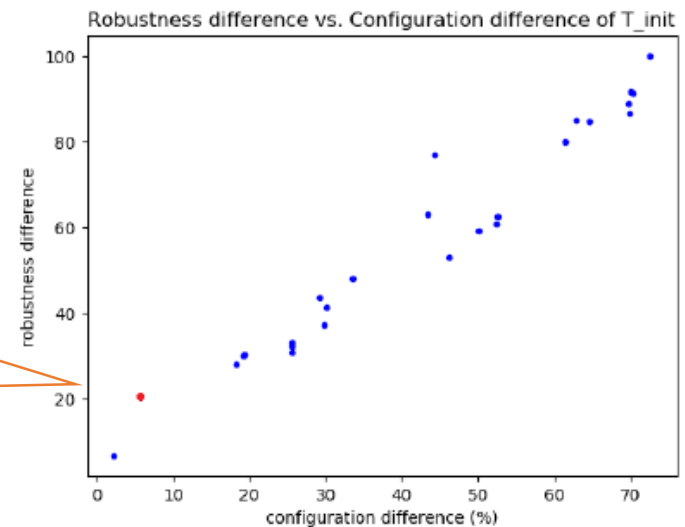
研究事例：安定性（敏感性）分析

■不安定なパラメータ領域を探索

パラメータ値の少しの変更が安全や品質に大きく影響するような状況を報告

- 設計パラメータ（避けるか重点的にテスト）も、
- 環境パラメータ（重点的にテスト）も

動き出し時間設定の
0.12 秒の差により、
衝突無しの状況が
20km/h衝突に変わる

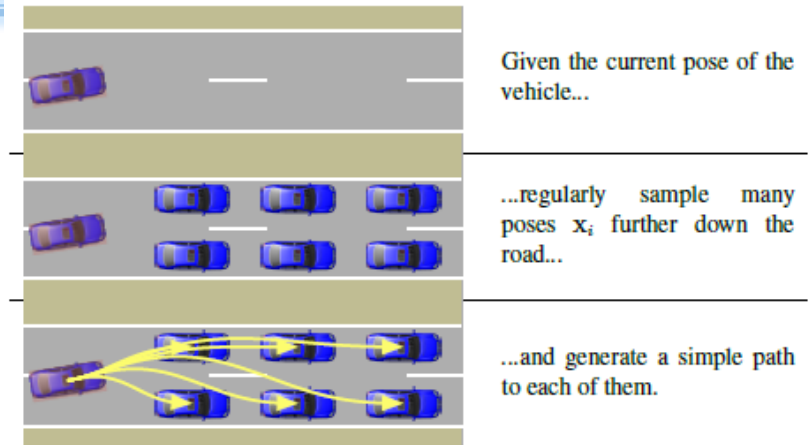


[Lee et al., Stability Analysis for Safety of Automotive Multi-Product Lines: A Search-Based Approach, GECCO'19]

システムレベルへ発展中

■ 自車の経路を定める部品

[Figure from McNaughton, Parallel Algorithms for Real-time Motion Planning]



- 障害物との距離，レーン遵守，車両の物理的限界，
なめらかさ・快適さ，法令遵守，運転慣習遵守，・・・
- 道路形状，歩行者・他車の位置や動きなどを
設定してシミュレーション実行

➡ **膨大な可能性（シミュレーション設定）の中で
「意味のあるテスト」への探索も同時に**

おわりに

機械学習ベースAIシステムのための深化

■ 自動運転・物理系に限らず最近の重要課題

■ 例： 機械学習工学研究会キックオフシンポ（18/5/17）

■ 例： Open QA4AI Conference



機械学習工学研究会
MACHINE LEARNING SYSTEMS ENGINEERING

QA4AIガイドライン初版リリース（19/5/17）



■ 発見的・経験的テストの応用は世界的にも注目

■ 例： 誤認識が起きる方向にノイズの足し方を寄せていく

■ 様々な方向で取り組み中

■ メタモルフィックテストティング,
全体精度よりも密な評価,
再訓練（デバッグ）サイクルなど



1.1 original



1.2 with added rain

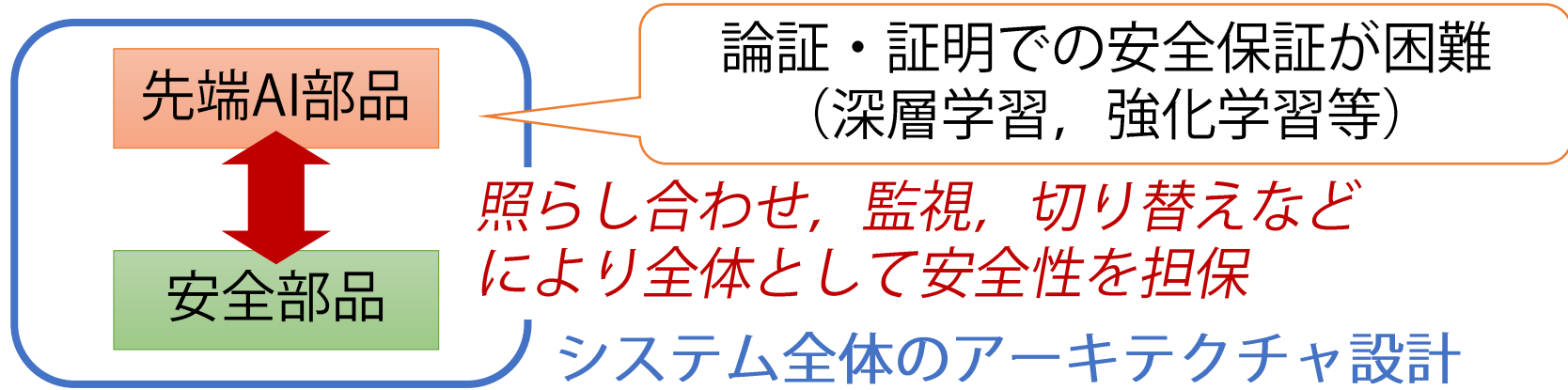
[Tian et al., DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars, 2018]

演繹と帰納の融合

■システム全体をブラックボックステスト，以上？

- 問題が大きすぎる（例：不具合の原因追及が困難）

■自動運転や，機械学習工学一般での典型課題



➡ 形式手法（演繹的保証）による問題分割の力・強い保証の力をテスト（帰納的保証）と組み合わせ

- 手持ち技術：「証明リファクタリング・再利用」

[T. Kobayashi et al., Consistency-Preserving Refactoring of Refinement Structures in Event-B Models, 2019]

まとめ：AIによるテスト・AIのためのテスト

■発見的・経験的なアプローチ

「実行可能なもの」があれば適用できる、
(一つの) 重要で現実的なアプローチ

■実世界の難しさとの戦いも主軸に (不確実性・オープン性)

■理論・演繹と経験・帰納の協働へ

今日お話ししなかった
ビッグデータを活用した
AIによるテストの可能性も

■皆様からの問題投げ込みを受けて活発化・技術深化

➡トップダウンの取り組みとツール化へ