

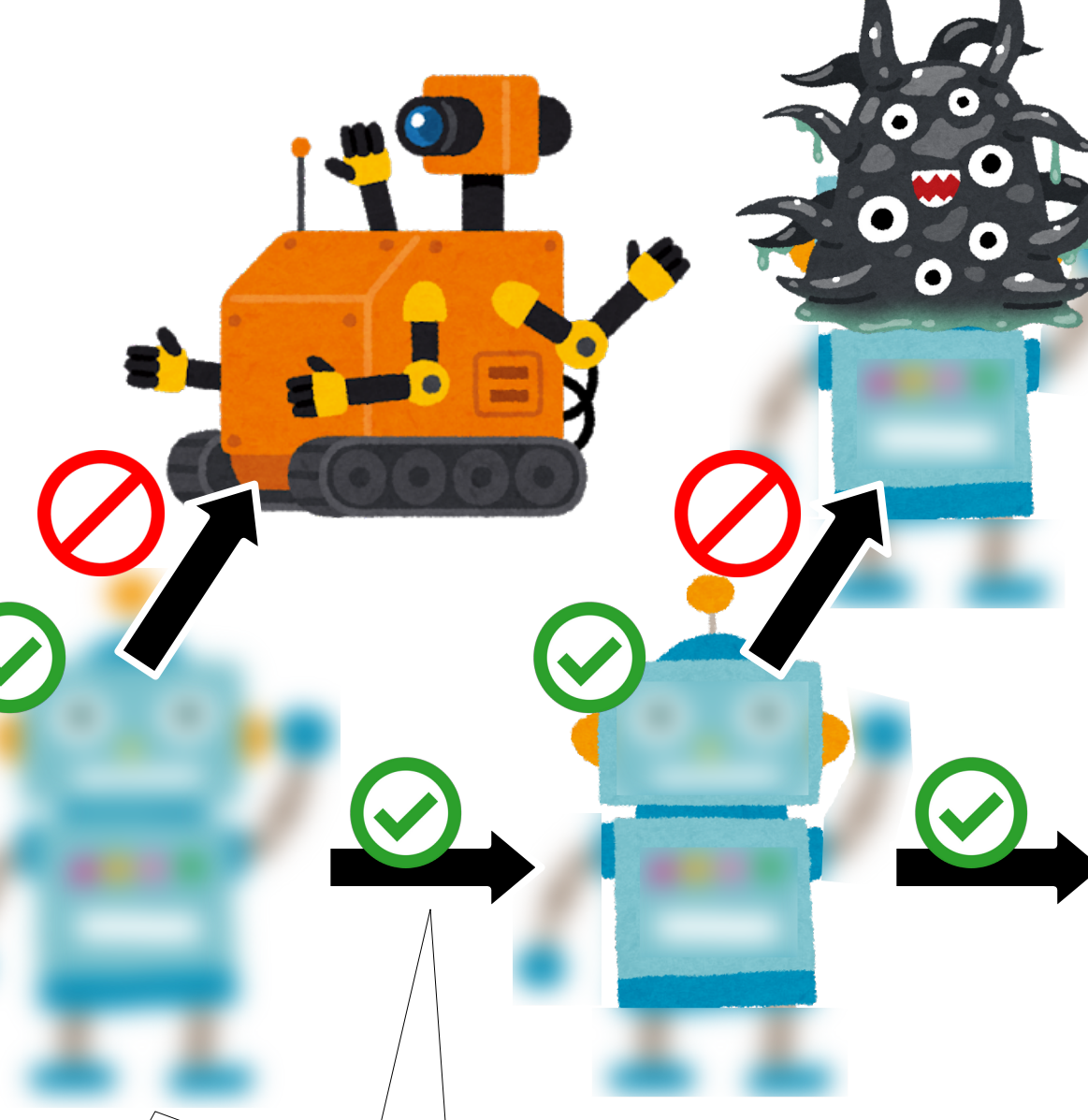
## 既存研究：形式仕様・段階的詳細化・定理証明による高信頼ソフトウェアの開発

### 形式仕様と定理証明による開発アプローチ

- ソフトにテストで得られる以上の高信頼性が欲しい
  - そもそもソフトが何を (what) 達成するか厳密に確かめたい
  - ……どう (how) 達成するか以前に what の検証も困難
- **仕様**を数学の言語で書いて、**数学的証明**で**正しさ**を検証
- 何が起る？どんな安全性が要る？      なぜ安全性が満たされる？
- 検証できた部分は（原則）**バグ無し！！**

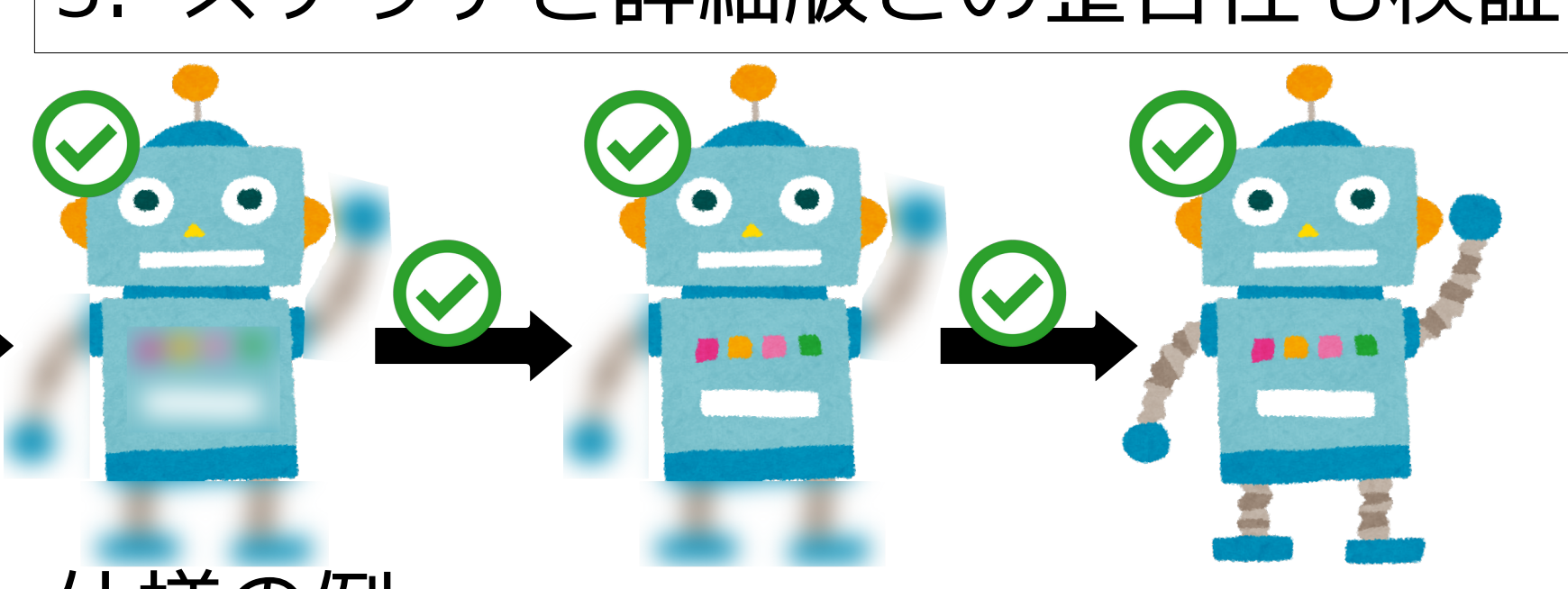
### 段階的詳細化による形式仕様記述

- 問題：現代のソフトは複雑 → 仕様構築・検証が難しい
- **多段階の抽象度**を厳密に扱う手法の出現 (Event-B<sub>[1]</sub> など)
- まずスケッチを作り、後で詳細版を作り、両者の整合性を確認可（**厳密な「ぼかし」**）
- 複雑さを軽減しながら厳密に仕様構築！**  
**ヨーロッパを中心に産業応用も！** [2]



**モデリング・検証の手順**

1. 詳細をぼかしたスケッチを記述・検証
2. 後から詳細版を記述・検証
3. スケッチと詳細版との整合性も検証



**仕様単体の整合性：**

- 矛盾が無い？
- 振舞いは安全性を満たすか？

**2仕様間の整合性：**

- 2つの振舞いは対応付いているか？

**仕様の例**

**不変条件：** 受信ファイル = 送信ファイルの先頭 (1...i) 番目

**イベント：**

受信イベント定義：

$i \leq$  送信ファイルのサイズ      のとき

受信ファイルの i 番目

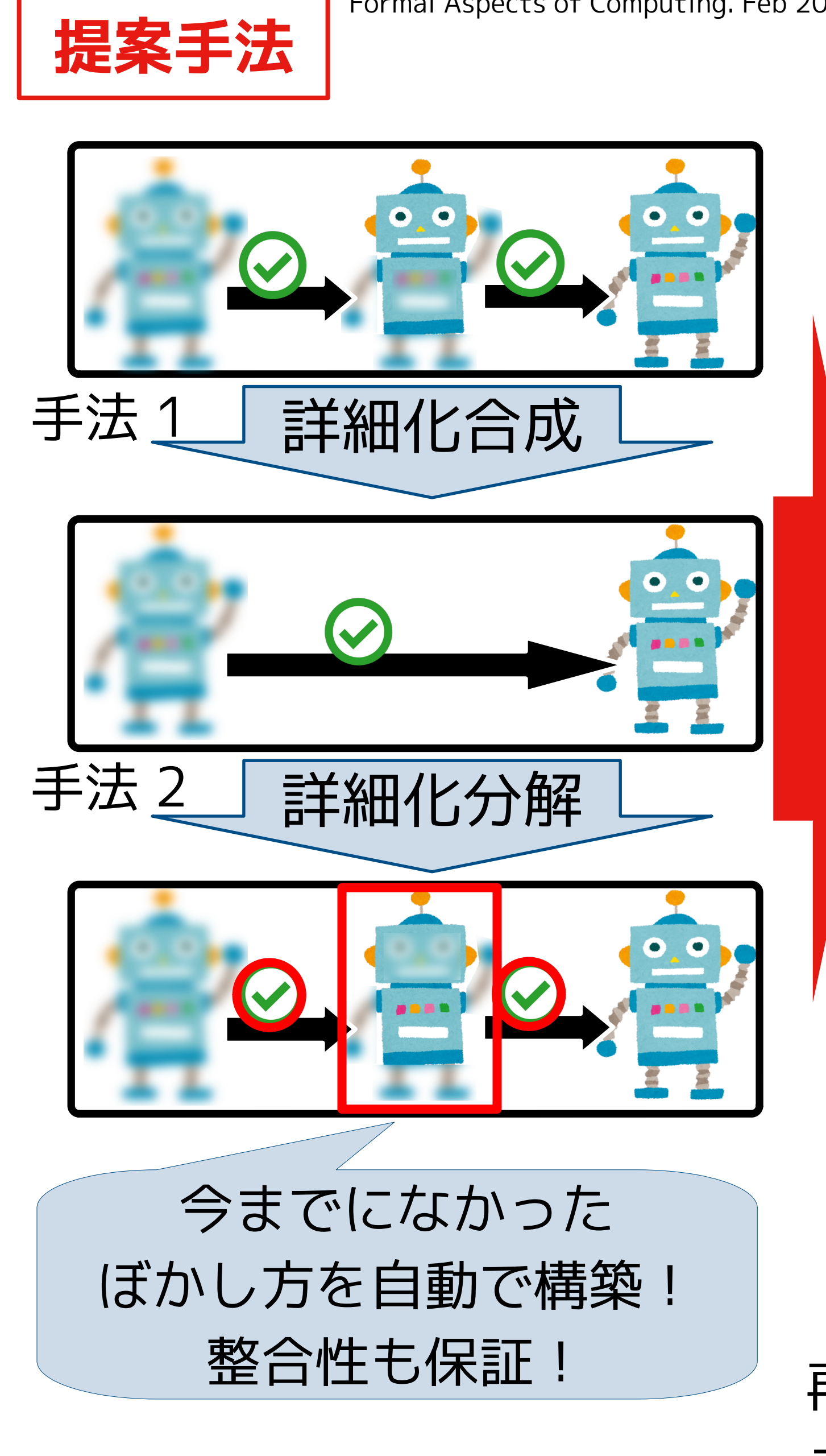
$:=$  送信ファイルの i 番目      に変化

$i := i+1$       に変化

## 研究1：形式仕様の抽象度の厳密・柔軟な変更

- 動機**
- 段階的詳細化の仕組みはとても有用だが……  
実際の開発に適用する際には困難がある
  - どうぼかし方を設計したら良いかわからない
  - 作った後の変更・再利用が難しい
  - 変更する部分（再利用しない部分）を変更  
→ 他の部分と整合性が無くなることもある

- 貢献**
- 既存仕様のぼかし方を、整合性を保ちつつ変更  
→ いろいろなぼかし方を生成・比較  
→ 変更の不要な（再利用可能な）部分を  
集めたスケッチを構築、それを元に新しい  
仕様を容易に構築できる
- **形式仕様記述のコスト削減、適用範囲拡大**
- 段階的詳細化による仕様記述・検証を  
より使いやすくし、適用の幅を広げる！**



**応用1：ぼかし方の戦略の分析** [3]

問題：どうぼかし方を設計したらいいかわからない

仕様複雑さ 5      証明複雑さ 4

仕様複雑さ 6      証明複雑さ 8

仕様を入力として、いろいろなぼかし方を生成・分析  
→ **どんなぼかし方が複雑さを減らすかの知見を獲得**

**応用2：仕様の効果的な再利用** [4]

目的：既存の仕様を過不足なく再利用したい

① → ② → ③ → ④

を作るには②に戻って作り直す必要あり

再利用可能な部分以外をぼかす      再利用可能な部分  
→ **再利用可能な部分を過不足なく獲得、効果的に再利用**

## 研究2：自動運転ソフトウェアの安全アーキテクチャの構築・検証 @ ERATO 蓮尾プロジェクト（ポスター [B-08]）

- 動機**
- 自動運転ソフトをできるだけ安全にしたい
  - 仕組みとして安全にしたいし、  
個別のコンポーネントや動作もできるだけ検証したい
  - さまざまな車に適用できる一般的な枠組みを作りつつ  
個別の車の問題にも適用したい

- 目標**      **安全な自動運転アーキテクチャを作る！**
- 一般的な安全アーキテクチャの要求項目を提案
  - 自動運転の難しさ（→）に対処する
  - **アーキテクチャ仕様を Event-B など**で構築・検証
  - **一般アーキテクチャ**をもとに  
安全性の保証された**個別アーキテクチャ**を構築・検証

