ソフトウェアの働き・性質を正しく要約

整合性を保持する形式仕様の自動抽象化システム「ソフトウェア展望台」の開発

科学技術振興機構　ACT-I 専任研究者　小林 努　t-kobayashi@nii.ac.jp　Web:

研究の目的

背景
- 高信頼性が必要で複雑なソフトウェア増
- 実装詳細を除いた仕様を形式言語で記述・検証する手法が有効

問題・提案　せっかく explainable な形なのに…

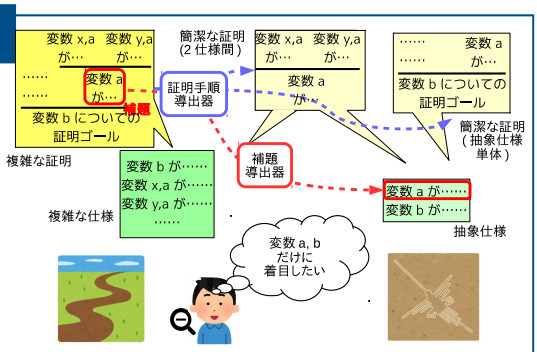
- 仕様で多くの要素が複雑に絡み合い改良・再利用時に理解が難しい

要は何が起こる？　→　要はなぜ正しい？

- 「抽象版」仕様を整合性証明付きで自動構築
  - 元の仕様と整合性を持つ
  - 一部の変数のみで書かれている

主な達成目標

Challenge:
元仕様と整合性のある抽象仕様を作る



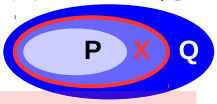
1. 補題導出器の構築
   元仕様に直接は記述されていないが整合性証明で本質的な式を獲得、抽象仕様に追加
2. 証明手順導出器の構築
   抽象仕様単体の整合性と抽象仕様 - 元仕様間の整合性証明を構築

ACT-I 期間中の成果

基本アイデア

整合性の根拠を特定の語彙で説明

- 元仕様の整合性から抽象仕様の整合性に必要な式を獲得
- Craig の補間定理を利用
  元仕様の整合性条件を変換することで
  ⇒の前後の式に現れる変数を調整



P ⇒ Q なる P と Q について、
P と Q に共通の記号が存在するならば、
以下を満たす式 X(補間) が存在する：
1. P ⇒ X　かつ　X ⇒ Q
2. X に現れる記号は P にも Q にも現れる

- 特定の変数で書かれた、整合性に必要な式獲得

補題導出器の開発 [1]

語彙を指定
→整合性の根拠を獲得

例：抽象版イベントの発火条件は元イベントの発火条件より弱い

$$I_A \wedge I_C \wedge G_C \Rightarrow G_A$$

元の整合性の式

$$\widetilde{I}_A \wedge I_{AB} \wedge I_{BC} \wedge \widetilde{I}_C \wedge G_{BC} \wedge \widetilde{G}_C \Rightarrow G_A$$

1. 抽象仕様の変数で記述できるものとできないものに分離

$$I_{AB} \wedge \widetilde{I}_C \wedge \widetilde{G}_C \Rightarrow G_A \vee \neg \widetilde{I}_A \vee \neg I_{BC} \vee \neg \widetilde{G}_{BC}$$

2. ルールに従い移項

$$\Rightarrow X \Leftarrow$$

3. 補間獲得

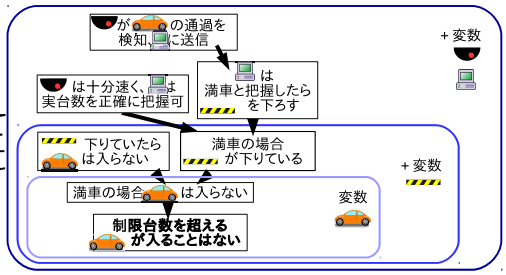
I_A … 超抽象仕様の不変条件　G_A … 超抽象仕様のイベント発火条件　I_{BC} … I_Cの中で抽象仕様の変数で表せるもの
I_C … 具体仕様の不変条件　G_C … 具体仕様のイベント発火条件　$\widetilde{I}_C$ … I_Cの中で抽象仕様の変数で表せないもの

- 各種整合性条件 (不変条件が保たれる、抽象版イベントの発火条件は元より弱い、…) の式の変換
- ソルバを利用して補間を獲得、抽象仕様に追加
- 並列分散システムの理論 Action Systems に一般化も

証明手順導出器の開発

使う語彙を少しずつ増やして整合性の根拠を述べる
→ 正しさ証明のスケッチ獲得

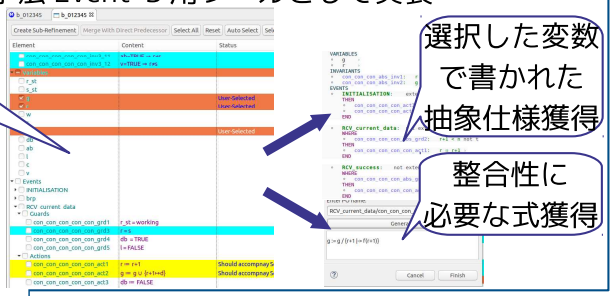
- 各記述に現れる変数の集合を狭めるように証明木を作る
- 補間を利用し、証明木の各ノード (各補題) を獲得
- 補題をつなげて証明木を構築
- 元仕様の証明木の一部を取得してより詳しい証明に



展望台ツールとして統合

http://research.nii.ac.jp/slicenmerge/

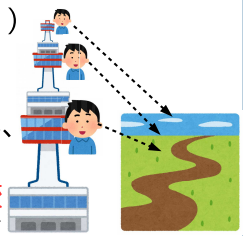
- 仕様記述手法 Event-B 用ツールとして実装



多段階の抽象化・詳細化の戦略分析 [2]

どう語彙を増やすと見通しが良くなる？

- 抽象化を繰り返し適用すると、多段階の仕様が得られる
  (抽象から具体に向け詳細化していく過程が得られる)
- 抽象仕様の変数の選び方によって詳細化は多様
  … 変数 {a, b, c, d} のシステムを作るために、
  [{a}, {a, b}, {a, b, c}, {a, b, c, d}] と段階を踏むか、
  [{c, d}, {a, b, c, d}] と段階を踏むか、……
- 展望台ツールにより仕様の様々な詳細化を生成・比較
  → 詳細化の粒度や、先に入れると簡潔になる変数などについての知見を獲得
- 証明手順導出器でどの順番で変数を狭めるかに応用



今後の予定・展望

- 本格的なユーザ実験
  - 大きめの規模の題材に適用、理解を促進できるか？
  - 誤りのある仕様に適用、見通し良く発見・修正できるか？
- 工学的応用
  - 環境の変化に合わせた仕様の適応
  - 既存仕様を抽象化→ライブラリ化
  - プログラムコードを抽象化・解析

厳密かつ見通しの良い開発で安全なソフトウェア社会へ！

[1] Tsutomu Kobayashi, Fuyuki Ishikawa, and Shinichi Honiden. Consistency-preserving refactoring of refinement structures in Event-B models. Formal Aspects of Computing. Feb 2019.
[2] Tsutomu Kobayashi and Fuyuki Ishikawa. Analysis on strategies of superposition refinement of Event-B specifications. In Proceedings of the 20th International Conference on Formal Engineering Methods (ICFEM'18), pp. 357-372. Nov 2018. (Best paper award)