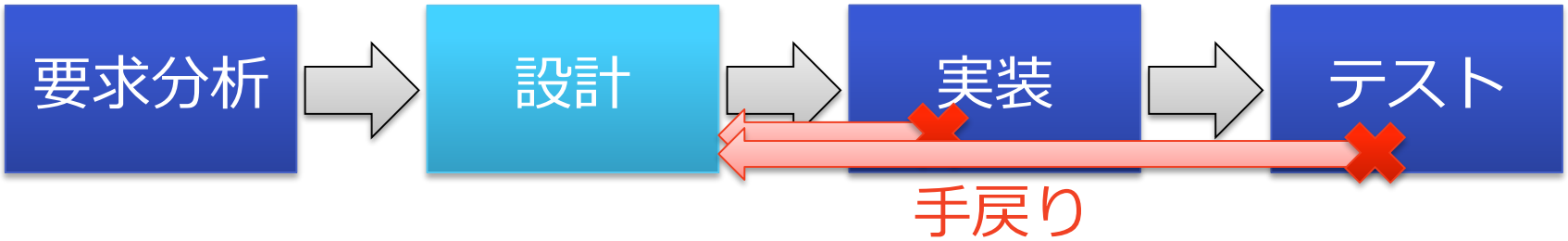
2013年度ソフトウェア工学分野の先導的研究支援事業

抽象化に基づいた UML設計の検証支援ツールの開発

有本和民　　横川智教
佐藤洋一郎　　天嵜聡介
岡山県立大学

研究背景



手戻りによる開発コスト増大 ⇒ 設計検証の必要性の高まり

1. クリティカルな分野のソフトウェアでは，１件の不具合が社会に及ぼす影響が甚大である
2. 大規模・複雑化が顕著な組込みソフトウェアでは，システムの取り得る状態数が人手でテストを行う限界を越える

⇒ 網羅的・自動検証の可能なモデル検査技術の利用

研究課題

モデル検査を設計検証に導入する上での問題点

問題１．モデル作成の困難さ

対象システムを検証ツール固有のモデル化言語で記述

⇒ モデル作成には専門的な知識やノウハウが必要

問題２．状態爆発の危険性

モデル化された対象システムの状態空間を網羅的探索

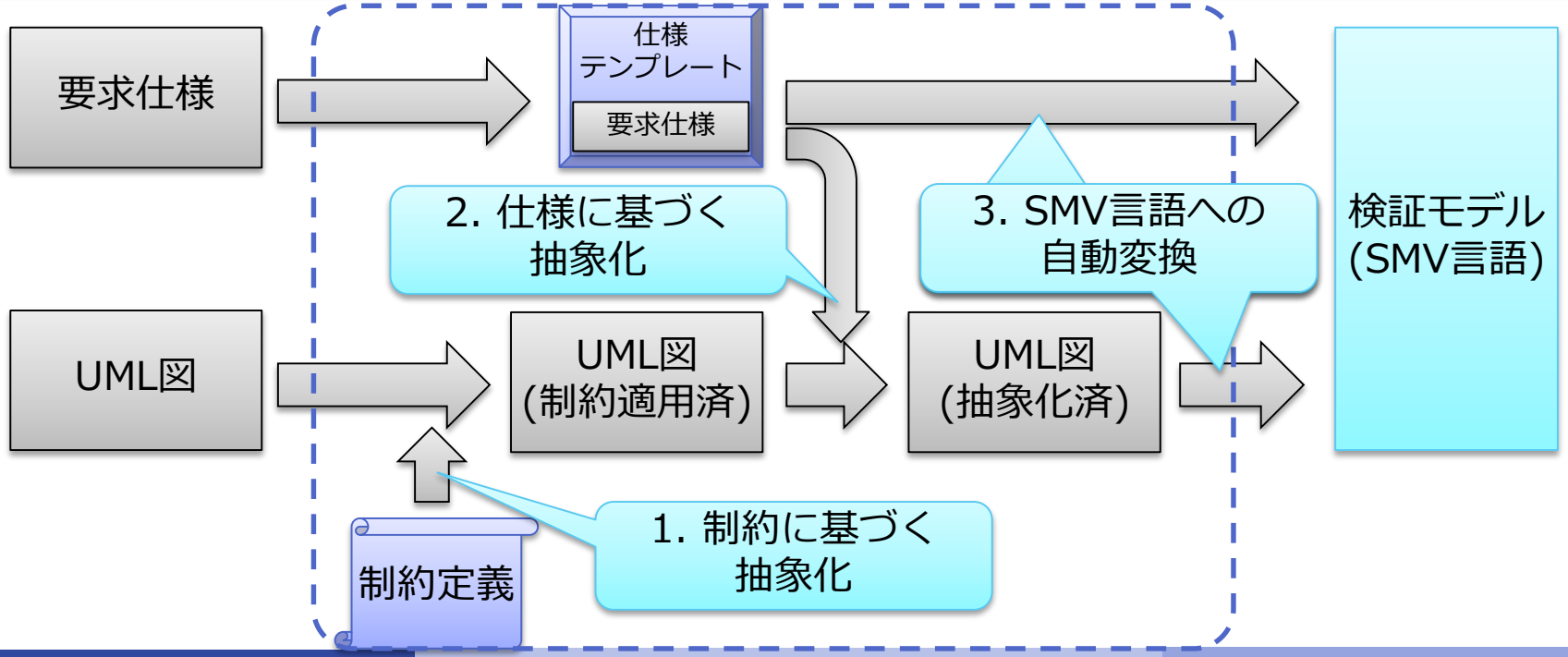
⇒ モデル規模によっては現実的な時間での検証が困難

モデル作成を支援するためのツールを開発し問題を解決

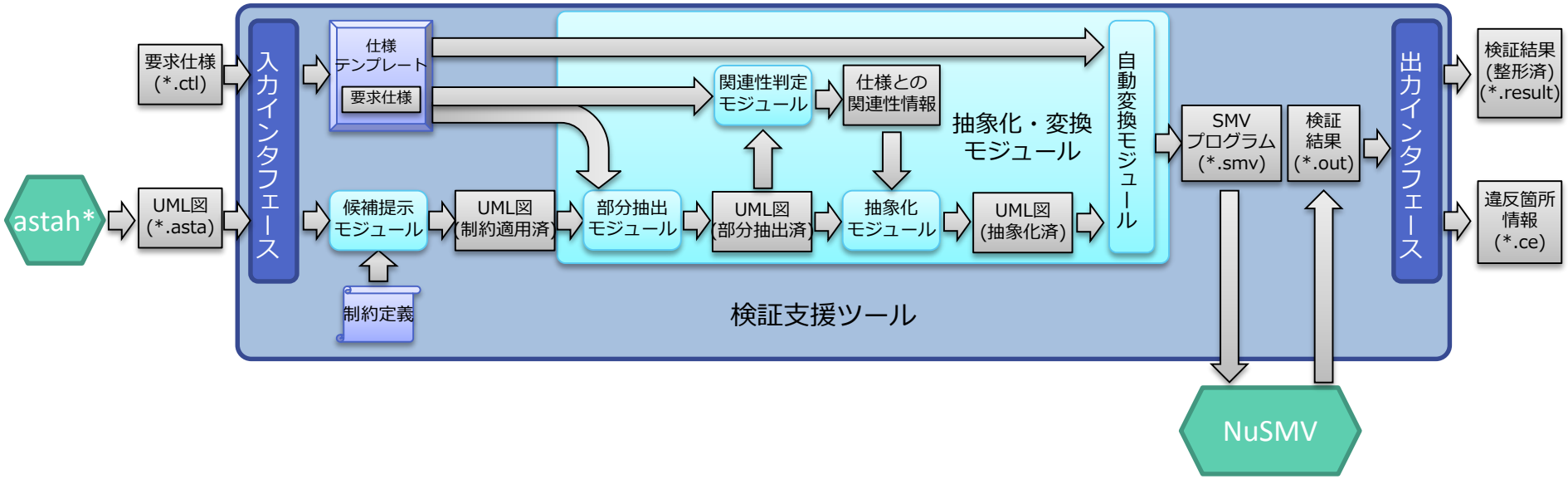
研究目標

以下の３つの機能をもつ検証支援ツールの開発

1．SMV言語への自動変換を目的としたUML図の抽象化
2．モデルサイズ削減を目的としたUML図の抽出・分割および抽象化
3．UML図からSMV言語への自動変換



検証支援ツールの概要



- UML図の記述にはastah*を用い，モデル検査ツールはNuSMVを用いる
- ５つの機能モジュールによって構成される
- 検査対象となるUML図が満たすべき制約定義と，要求仕様を記述するための仕様テンプレートが定められている

中間目標

研究目標である検証支援ツールの実現のため，
以下の７つの中間目標を定める

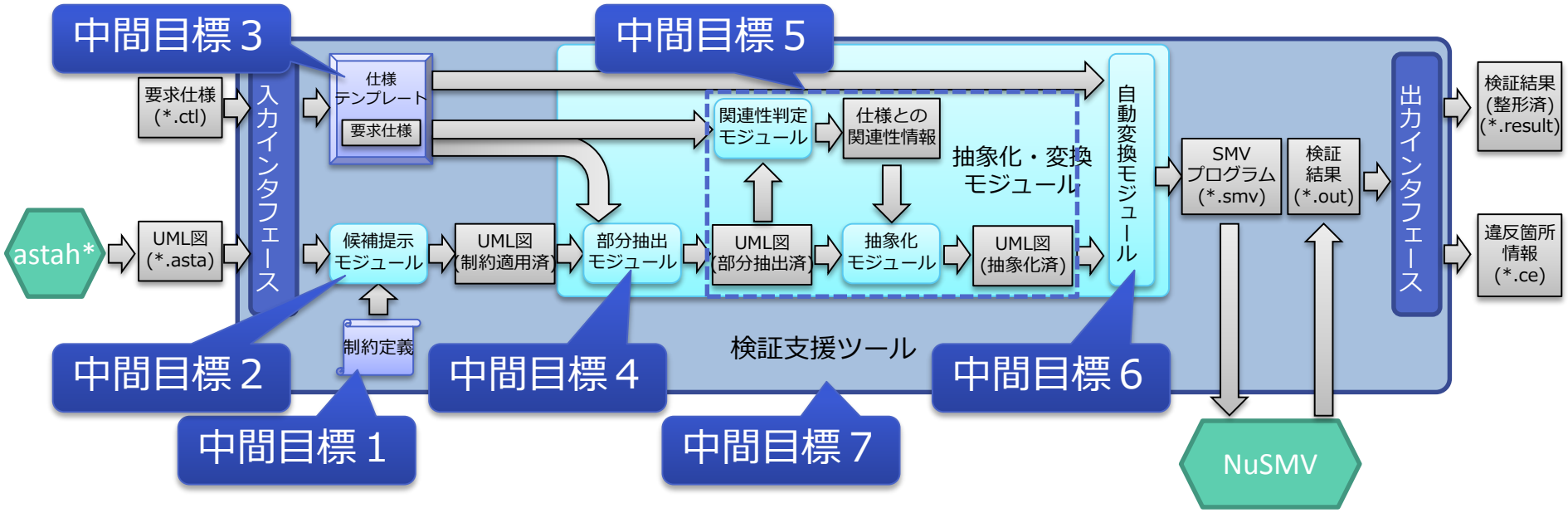
中間目標１．自動変換を行うUML図に対する制約定義書の作成
中間目標２．UML図の制約違反検出および抽象化手法の開発
→ 機能１

中間目標３．仕様の入力テンプレートの作成
中間目標４．仕様に基づくUML図の部分抽出および分割手法の開発
中間目標５．仕様に基づくUML図の抽象化手法の開発
→ 機能２

中間目標６．UMLおよび仕様からSMV言語への自動変換手法の開発
→ 機能３

中間目標７．検証支援ツールの作成

到達課題と中間目標の関係



中間目標１．自動変換を行うUML図に対する制約定義書の作成
中間目標２．UML図の制約違反検出および抽象化手法の開発
中間目標３．仕様の入力テンプレートの作成
中間目標４．仕様に基づくUML図の部分抽出および分割手法の開発
中間目標５．仕様に基づくUML図の抽象化手法の開発
中間目標６．UMLおよび仕様からSMV言語への自動変換手法の開発
中間目標７．検証支援ツールの作成

研究成果

1. 検証支援ツールの開発
   - 制約定義，仕様テンプレート，機能モジュール，そしてツールの入出力インタフェースを設計
   - ツール実装は業者に外注
2. 事例への適用
   - 某ソフトウェア開発企業から提供された事例に対して本ツールを適用
   - 設計の誤りを正しく検出できた

検証支援ツールの開発

- 入出力
  1. 入力
     - UML図
     - 要求仕様
  2. 出力
     - 生成されるSMVプログラム
- 機能
  1. 制約違反の検出・修正候補の提示
  2. 検査性質に基づくUML図の部分抽出・抽象化
  3. SMVプログラムの自動生成
     - 状態マシン図の変換
     - シーケンス図の変換
     - 仕様テンプレートの変換

検証支援ツールの入出力

ツールの入力

1. 検査対象となるUML図
   記述制約を満たす状態マシン図およびシーケンス図
2. 要求仕様
   仕様テンプレートを用いて記述，またはCTLを用いて直接記述

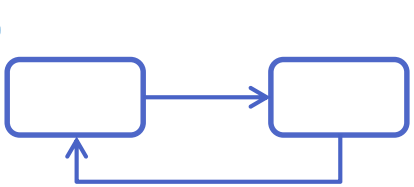
ツールの出力

1. SMVプログラム
   状態マシン図を振る舞いモデル，シーケンス図および要求仕様をCTL式で記述

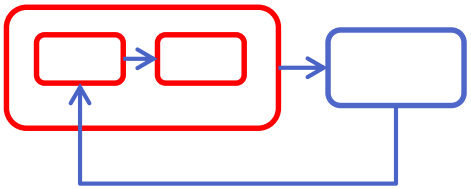
状態マシン図の記述制約

- ✓ 単純状態および状態間の遷移を扱う
- ✘ 合成状態など状態の階層構造は扱わない

○



×



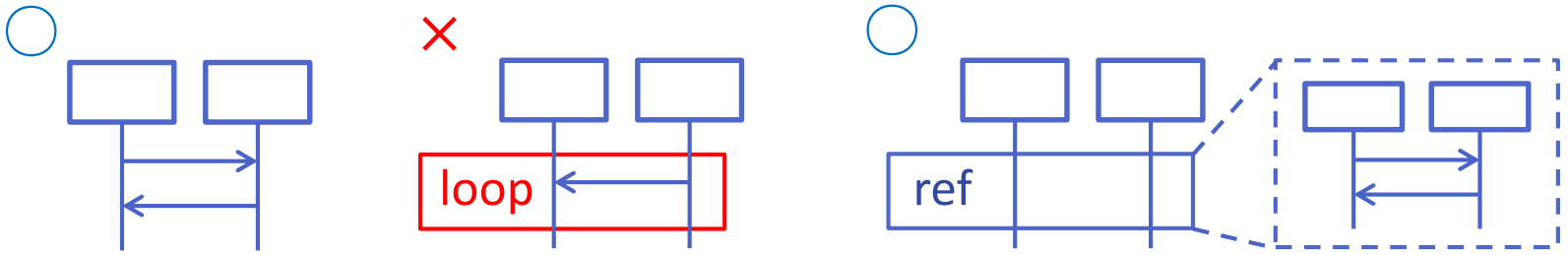
- ✘ 整数型とブール型以外の変数は使用不可
- ✘ 変数に関するガード条件は整数値との線形制約以外は不可
- ✘ アクションでの変数更新は整数との四則演算以外は不可

○ [x>2]/y=x+2

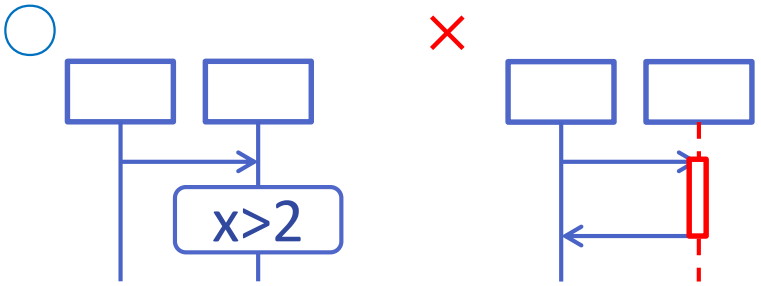
× [x==y]/y=log z

シーケンス図の記述制約

- ✓ 単純なシーケンス図を扱う
- ✘ 結合フラグメントによる相互作用の階層構造は扱わない
- ✓ 単純な参照(ref)は使用可能



- ✓ 状態不変表明として整数値との線形制約を利用可能
- ✘ オブジェクトの生成・終了は扱わない



仕様テンプレート

1．３つの要素に関する特性を記述可能

状態　メッセージ　変数

2．３つの特性を記述可能

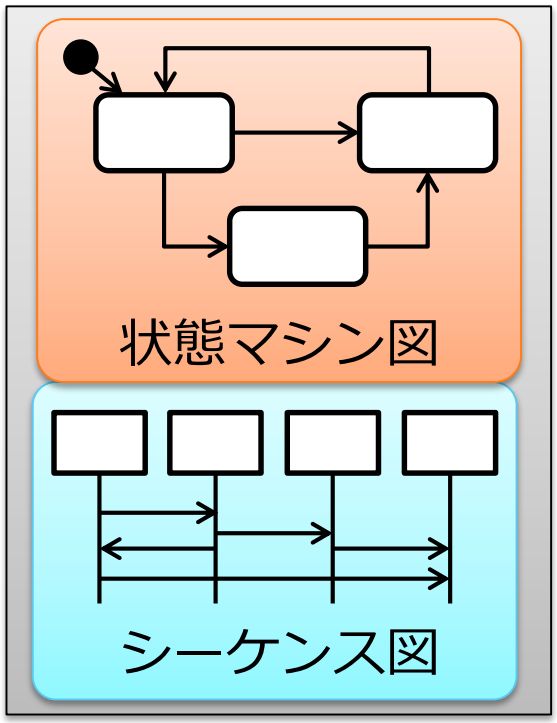
安全性：決して～～とならない

活性：いつか必ず～～となる

到達可能性：将来的に～～となる可能性がある

例）safe(x, a)
決して変数xの値がxにならない

生成されるSMVプログラム



astah*ファイル(*.asta)

safe(xxx = 1)
live(message_1)
reachable(S)

特性ファイル(*.ctl)

モデル生成

CTL式
生成

```
MODULE main

VAR
  XXX : {aaa, bbb, ccc};
  YYY : {ddd, eee, fff};
  fg1 : boolean;

ASSIGN
  init(XXX) := aaa;
  next(XXX) := case
    fg = TRUE : bbb;
    TRUE : XXX;
  esac;

  init(YYY) := ddd;
  next(YYY) := case
    fg = FALSE : fff
    TRUE : YYY;
  esac;

  init(fg) := FALSE;
  next(fg) := case
    fg = FALSE : {TRUE, FALSE};
    TRUE : fg;
  esac;

SPEC AG !(XXX = ccc)
SPEC EF(ZZZ = iii)
SPEC AF(YYY = fff)
```

モデル

CTL式

SMVファイル(*.smv)

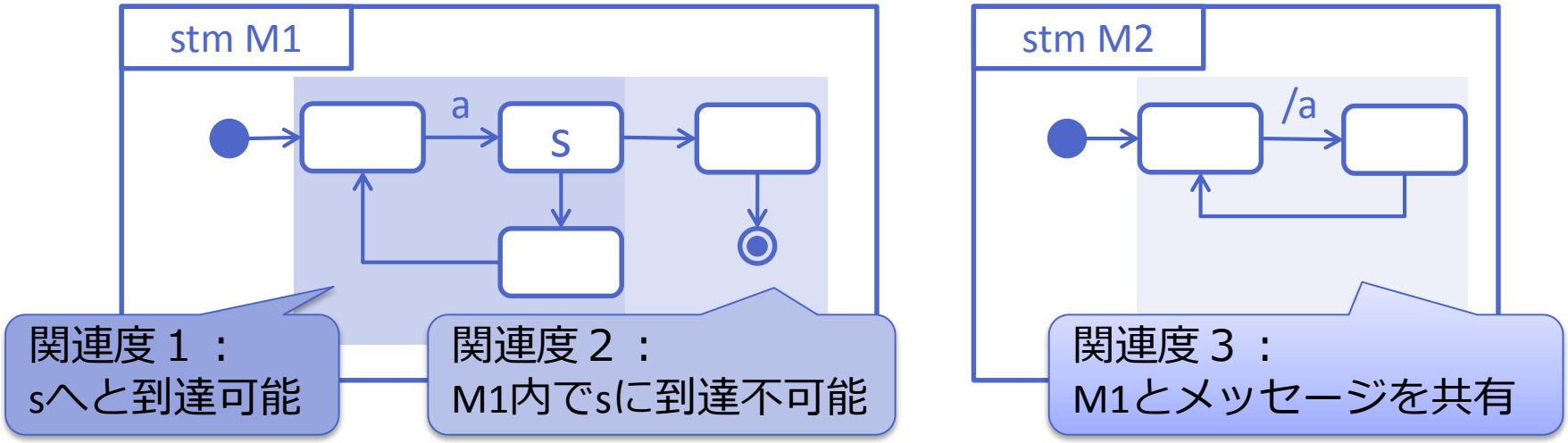
制約違反の検出・修正候補の提示

1. 状態マシン図・シーケンス図を読み込み，制約に違反する箇所全てを検出する
2. 以下の制約違反については修正候補を提示する
   - 整数型・ブール型以外の変数の利用
     - 小数点以下を切り捨てた上で整数型と解釈
   - 生存線の生存期間の記述
     - 常時生存しているものとして扱う
   - オブジェクトの生成・終了メッセージの記述
     - 非同期送信メッセージとして扱う

検証性質に基づくUML図の部分抽出・抽象化(1)

1. 仕様に対する関連度の計算
   - 状態に関する仕様（例：safe(s)）に対して，状態マシン図の状態・遷移との関連度を求める
2. 関連度に基づく部分抽出
   - ユーザが与えた閾値を越える状態・遷移のみを抽出する

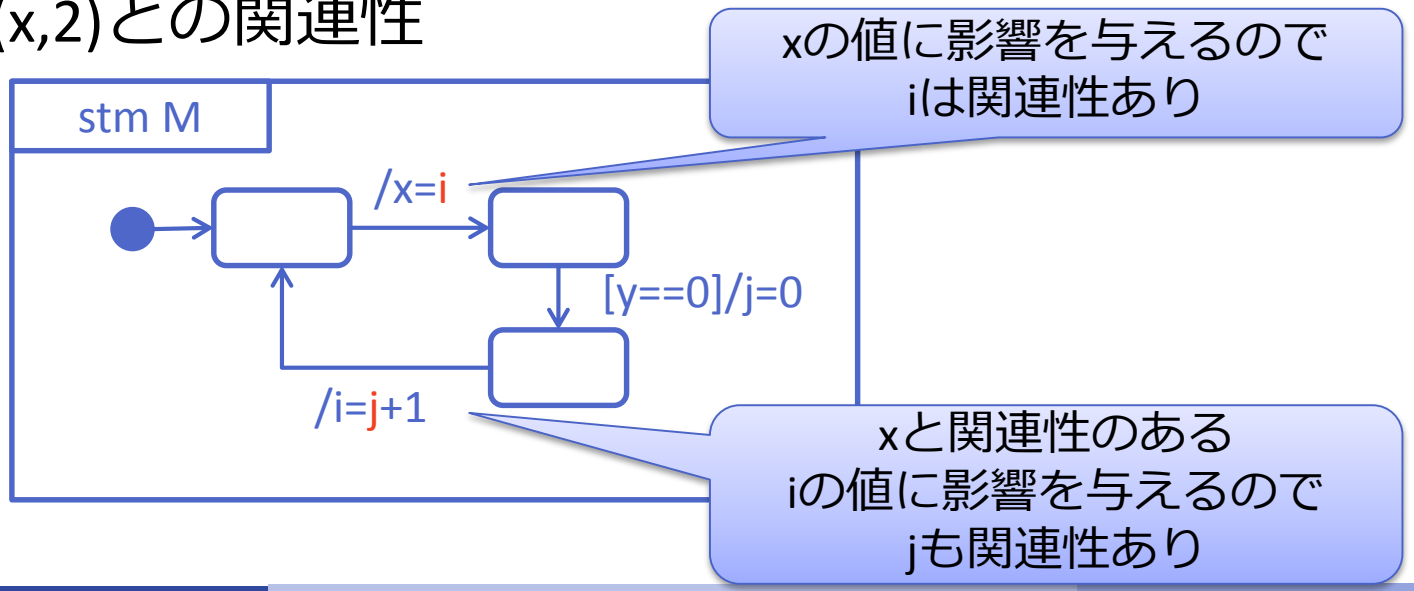
例：safe(s)との関連度



検証性質に基づくUML図の部分抽出・抽象化(2)

3. 仕様に対する関連性の判定
   - 変数に関する仕様（例：safe(x,2)）に対して，状態マシン図の変数との関連性を判定する
4. 関連性のない部分の抽象化
   - 関連性のない変数yについて，yの値を変化させるアクションやyを含むガード条件を削除する

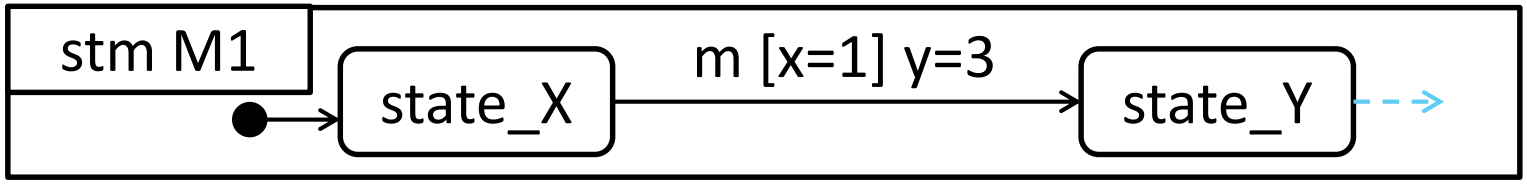
例：safe(x,2)との関連性



SMVプログラムの自動生成(1)

状態マシン図の変換

状態遷移とアクションによる振る舞いをそれぞれ
ASSIGN構文によりモデル化する




```
MODULE main
VAR
 x : 【型を入力してください】;
 y : 【型を入力してください】;
ASSIGN
 init(M1) := state_X;
 next(M1) := case
  M1 = state_X & (m) & (x=1) : state_Y;
  state_Yからの遷移条件 : state_Yの次状態;
  TRUE : M1;
 esac;
```

状態遷移

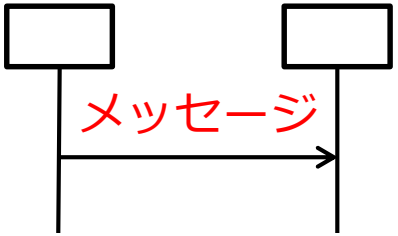
```
init(y) := 【初期値を入力してください】;
 next(y) := case
  M1 = state_X & next(M1) = state_Y : 1 ;
  TRUE : y;
 esac;
```

アクション

SMVプログラムの自動生成(2)

シーケンス図の変換

シーケンス図に記載されたメッセージの安全性，活性，到達可能性を確認するためのCTL式を生成する



項目	CTL式
安全性	SPEC AG !(メッセージ = TRUE)
活性	SPEC AF(メッセージ = TRUE)
到達可能性	SPEC EF(メッセージ = TRUE)

SMVプログラムの自動生成(3)

仕様テンプレートの変換

NO	式表現	CTL式
1	safe(変数 = 状態)	SPEC AG !(変数 = 状態)
2	safe(メッセージ)	SPEC AG !(メッセージ = TRUE)
3	safe(変数, 値)	SPEC AG !(変数 = 値)
4	live(変数 = 状態)	SPEC AF(変数 = 状態)
5	live(メッセージ)	SPEC AF(メッセージ = TRUE)
6	live(変数, 値)	SPEC AF(変数 = 値)
7	reachable(変数 = 状態)	SPEC EF(変数 = 状態)
8	reachable(メッセージ)	SPEC EF(メッセージ = TRUE)
9	reachable(変数, 値)	SPEC EF(変数 = 値)

事例への適用

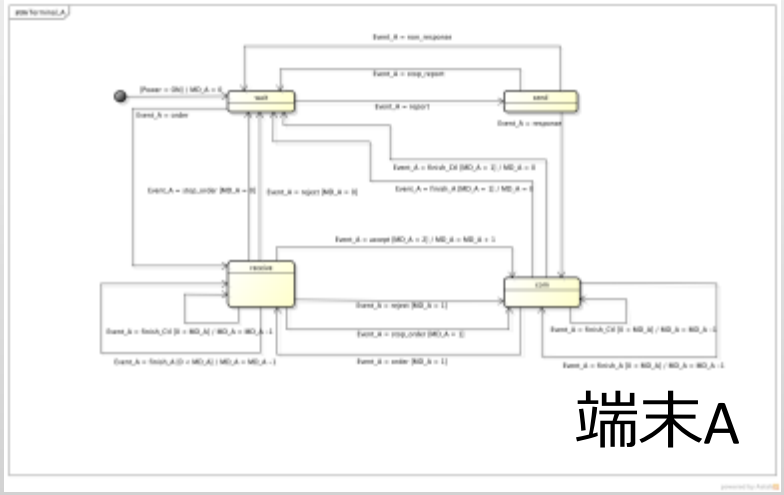
- 経緯
  - 某ソフトウェア開発企業に事例適用を打診
  - 既存の状態遷移表からUML図(状態マシン図)を作成し，提供していただいた
  - 対象システムは店舗従業員向けの「商品供給指示システム」(Realtime Commodity supply Directions System(R-CDS)である
- 概要
  - 検査１：仕様テンプレートを用いた基本特性の検査
    - 提供していただいた状態マシン図に本ツールを適用
    - 生成したSMVファイルをNuSMVを用いて検査
    - 検査結果が全てTRUE(誤りなし)であることを確認
  - 検査２：事例提供元より要望のあった特性の検査
    - CTL式で記述された特性および状態マシン図に本ツールを適用
    - 生成したSMVファイルをNuSMVを用いて検査
    - 特性の１つに対してFALSE(誤りあり)の結果が出力された
    - 反例解析の結果，状態遷移表から状態マシン図を作成するにあたってのミスがあったことがわかった

R-CDSの概要

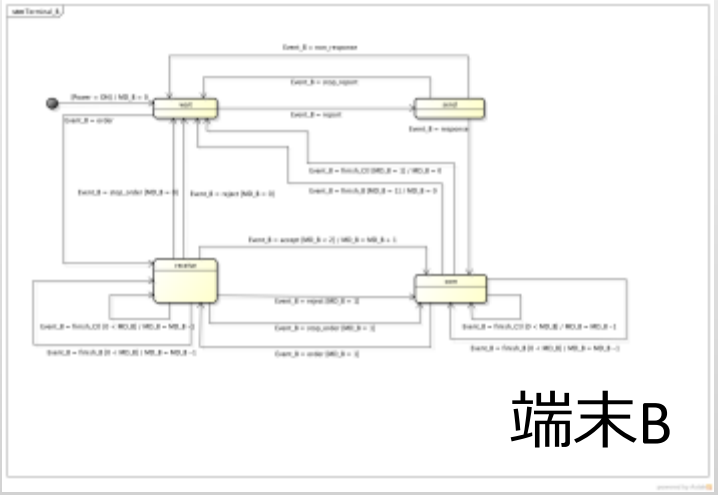
- 売り場従業員と商品管理室の責任者との通話システム
- 従業員は端末を用いて，責任者は管理用PCを用いて相互通信を行う
- 責任者は従業員に，什器への商品の補充命令を発令する
- 従業員は補充の結果や，現場の状況報告等を行う
- 従業員が持つ端末は同時に２通話(通常＋緊急)が可能である
- 商品管理室のモニターには，通信回線の利用数(＝通話数)に応じて，売り場従業員の配置数を評価する指標(過剰～不足)が表示される

検査対象となるUML図

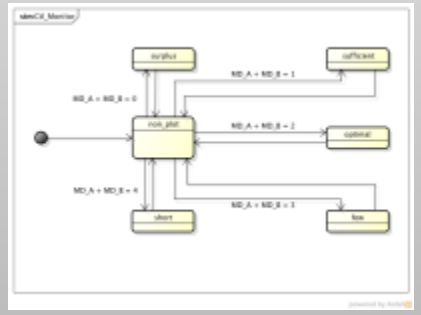
端末が２台の場合の状態遷移表とモニターの表示方法を
状態マシン図で書いて貰い提供していただいた



端末A



端末B



モニター

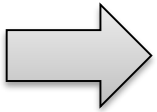
R_CDS.asta

検査 1

仕様テンプレートを用いて以下の検査を行った

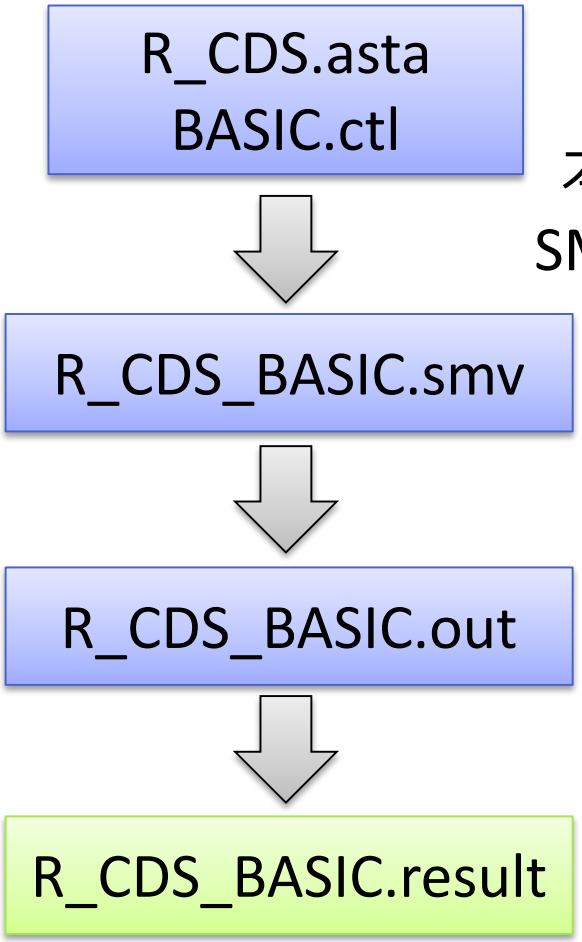
- 端末状態の到達可能性
- 通信モード(使用回線数)の到達可能性
- 通信モードの安全性
- 商品管理室のモニターの到達可能性

検査項目	テンプレート	式
状態の到達可能性	reachable(s)	reachable(Terminal_A = send) reachable(Terminal_A = receive) reachable(Terminal_A = com) reachable(Terminal_B = send) reachable(Terminal_B = receive) reachable(Terminal_B = com)
通信モードの到達可能性	reachable(x,a)	reachable(MD_A, 1) reachable(MD_A, 2) reachable(MD_B, 1) reachable(MD_B, 2)
通信モードの安全性	safe(x,a)	safe(MD_A, 3) safe(MD_A, -1) safe(MD_B, 3) safe(MD_B, -1)
モニターの到達可能性	reachable(s)	reachable(Ctl_Monitor = surplus) reachable(Ctl_Monitor = sufficient) reachable(Ctl_Monitor = optimal) reachable(Ctl_Monitor = few) reachable(Ctl_Monitor = short)



BASIC.ctl

本ツールの適用結果(検査１)



本ツールを用いて
SMVファイルを生成

モデル検査器
NuSMVで検査

検査結果の
整形

```
(001) EF Terminal_A = send  is true
(002) EF Terminal_A = receive  is true
(003) EF Terminal_A = com  is true
(004) EF Terminal_B = send  is true
(005) EF Terminal_B = receive  is true
(006) EF Terminal_B = com  is true
(007) EF MD_A = 1  is true
(008) EF MD_A = 2  is true
(009) EF MD_B = 1  is true
(010) EF MD_B = 2  is true
(011) AG !(MD_A = 3)  is true
(012) AG !(MD_A = -1)  is true
(013) AG !(MD_B = 3)  is true
(014) AG !(MD_B = -1)  is true
(015) EF Ctl_Monitor = surplus  is true
(016) EF Ctl_Monitor = sufficient  is true
(017) EF Ctl_Monitor = optimal  is true
(018) EF Ctl_Monitor = few  is true
(019) EF Ctl_Monitor = short  is true
```

R_CDS_BASIC.result

全ての結果がTRUE

⇒ 誤りは存在しない

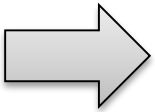
検査２

事例提供元より要望があり，以下の検査を行った

1. 端末が待機状態であり，かつ通信モードが通話無しでない状態への到達可能性
2. 端末が着信中状態であり，かつ通信モードが二話中である状態への到達可能性
3. 端末が通話中であり，かつ通信モードが通話無しである状態への到達可能性

```
1A: SPEC !EF(Terminal_A = wait & !(MD_A = 0))
2A: SPEC !EF(Terminal_A = receive & MD_A = 2)
2A: SPEC !EF(Terminal_A = com & MD_A = 0)

1B: SPEC !EF(Terminal_B = wait & !(MD_B = 0))
2B: SPEC !EF(Terminal_B = receive & MD_B = 2)
3B: SPEC !EF(Terminal_B = com & MD_B = 0)
```



R_CDS_REQ.ctl

本ツールの適用結果(検査２)

SMVファイルを生成し，NuSMVで検査した結果

(001) !(EF (Terminal_A = wait & !(MD_A = 0))) is true
(002) !(EF (Terminal_A = receive & MD_A = 2)) is true
(003) !(EF (Terminal_A = com & MD_A = 0)) is false
(004) !(EF (Terminal_B = wait & !(MD_B = 0))) is true
(005) !(EF (Terminal_B = receive & MD_B = 2)) is true
(006) !(EF (Terminal_B = com & MD_B = 0)) is false

R_CDS_REQ.result

3Aおよび3Bの特性について
結果がFALSEとなる

反例ファイル(R_CDS_REQ.ce)
を用いて反例解析を行った

3Aに対する反例(3Bも同様)

	1	2	3	4	5
Event_A	emp	emp	report	response	emp
MD_A	0	0	0	0	0
Terminal_A	initial	initial	wait	send	com

端末が通信中(Terminal_A=com)であるにもかかわらず，
通信モードが通話無し(MD_A=0)となる

状態遷移表のアクションの１つがUML図に記述されていなかった

事例提供元とのQ&Aとコメント

Q) UML図を動かしてテストしたのか？それともレビューか？

A) UML図を動かしていない．プログラム生成もレビューも無し．モデル検査という技術を行って漏れを発見した．網羅的な検査が可能なので，一カ所でも漏れがあれば発見可能．

Q) 記述間違いも発見できるのか？

A) もちろん発見できる．割込みのタイミング等の不具合も発見可能

コメント：

- 今回はUML図への転記ミスだが，当社では最初からUML図を用いて設計することもあるので，このような漏れや間違いを発見できるのであれば，モデル検査は非常に有用な技術である．
- ツールとしては，状態マシン図の全ての記法に対応して欲しい．また，GUIがあれば操作性が良くなるのではないか．

今後の予定

研究の継続および発展

- ツールの機能強化
  - 状態マシン図の全ての記法への対応
- 操作性・利便性の向上
  - GUIの開発
- さらなる事例への適用と検証コストの評価

成果の公表

- ツールの公開
- SEC Journalなど論文誌への投稿
- ETなど展示会への出展

成果の産業界への展開

- ソフトウェア開発企業との共同研究(本事例の提供元企業など)