基礎理論講座(第1回)
―導入編―

平成22年4月14日

トップエスイープロジェクト

改訂3版

第1回の授業内容

- 概要
  - 本講座の目的・内容・講師
  - 形式手法
- 前半: 形式仕様記述
- 後半: モデル検査

本講座の目的

- 本教育プログラムにおける，形式手法に関する基礎知識を与える.
- 対象講座シリーズ
  - 形式仕様記述講座シリーズ
  - モデル検査講座シリーズ
- 教科書
  - ソフトウェア科学基礎―トップエスイー基礎講座1（近代科学社）

必ず購入してください

本講座の内容

- 論理学
- 集合論
- 時相論理
- システムの性質の記述
- オートマトン
- モデル検査アルゴリズム
- モデル検査の実装方法
- 並行システム
- 抽象解釈
- モデル検査ツール

本講座を学ぶ理由

- 形式手法を適用するために必要な基礎的な知識が網羅されている.
- 基礎的な知識無しに形式手法のツールを利用することは難しい.
- 今後，検証可能なモデルの作成や，離散数学の概念に基づくシステムのモデル化について学ぶが，その基礎知識を得ることができる.

講師

- 磯部祥尚
  - (独)産業技術総合研究所
  - 国立情報学研究所
  - 形式手法による並行システムの検証に関する研究に従事
- 粂野文洋
  - (株)三菱総合研究所
  - 国立情報学研究所
  - 自動推論、エージェント技術、自己適用型ソフトウェアアーキテクチャ、形式手法ツールの応用に興味を持つ
- 櫻庭健年
  - (株)日立製作所システム開発研究所
  - OS，情報セキュリティの研究に従事
- 田原康之
  - 電気通信大学大学院
  - エージェント技術、ソフトウェア工学の研究に従事
- 田辺良則
  - 国立情報学研究所．(トップエスイー担当)
  - ソフトウェアモデル検証，時相・様相論理の研究に従事

形式手法（Formal Methods)

- 離散数学を用いたシステムのモデル化，検証方法.
- 次のような幅広い分野を含む.
  - プログラム論理（Hoare 論理）
  - プロセス代数
  - 形式仕様記述言語（代数型、モデルベース）
  - 定理証明
  - モデル検査
- ハードウェア設計の形式検証技術は，
  チップメーカなどにおいて幅広く利用されている.
- ソフトウェア設計や，プロトコルの検証にも実績がある.
- ソフトウェア実装についての検証も，困難はあるが，
  試みられている.

形式手法の分類

- モデル化する側面
  - モデルベース（状態型）
    - システムの静的な側面の仕様記述
    - 述語論理や集合論を基礎にした言語
  - プロセスベース（振舞い型）
    - システムの動的な側面のモデル化
    - オートマトンやプロセス代数を基とした言語
- 使いやすさや利用する際のコスト
  - Heavy-weight FM
  - Light-weight FM
- 検証する方法
  - モデル検査
  - 定理証明

今日における形式手法

- 標準での採用
  - Common Criteria (ISO/IEC 15408)
    - IT製品に関する国際セキュリティ標準
    - EAL5以上で，形式手法の利用を義務づけ
  - 機能安全 (IEC 61508)
    - 電機，電子，プログラマブル電子．
    - SIL2以上で，形式手法を推奨
- 産業界での適用事例
  - FeLiCa ICチップ開発: 形式仕様記述(VDM++)
  - C言語検証ツール VARVEL: 有界モデル検査
  - モデル検査の適用事例多数

講座概要

- 第2回～第5回: 形式仕様記述の基礎理論
  - 数理論理学，集合論，再帰的データ構造
- 第6回～第14回: モデル検査の基礎理論
  - 並行プログラム，時相論理，オートマトン，モデル検査アルゴリズム
- 第15回: モデル検査ツール概観

講座構成

関連講座

本講座

コンポーネントベース開発
ソフトウェアパターン
アスペクト指向開発
形式仕様記述（基礎編）
形式仕様記述（応用編）
形式仕様記述（セキュリティ編）
設計モデル検証（基礎編）
設計モデル検証（応用編）
実装モデル検証
性能モデル検証
並行システムの検証と実装
要求分析・定義
ゴール指向要求分析
超上流要求工学
セキュリティ要求分析
テスティング
プログラム解析
ソフトウェアメトリックス－測定と分析
基礎理論
修了制作

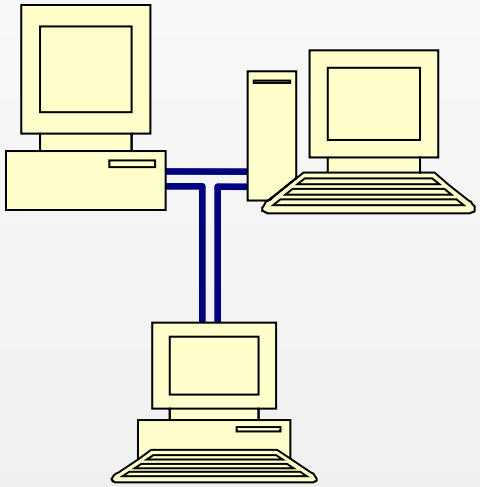
前半: 形式仕様記述の準備

形式仕様記述

- 現実世界（システム）を，数学 = 集合論 を使ってモデル化する．
- 数学の世界では，厳密な論証が可能となる．
- 数学の記述を形式化することで，誤りを排除でき，自動化も（ある程度）可能となる．

数学によるモデル化

システム



振る舞い
…
…
性質
…
…

仕様書

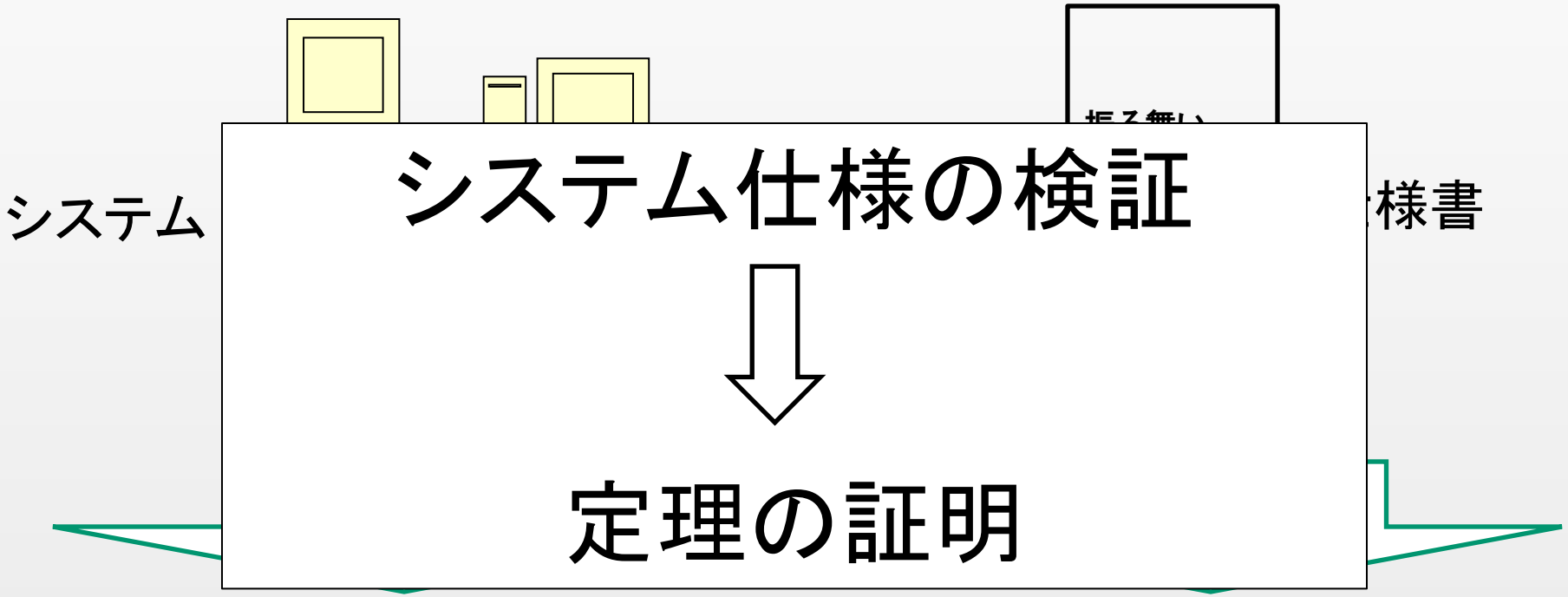
モデル化

記述

User = { Alice, Bob, … }
Alice ∈ Manager, Bob ∈ Employee
関数 title : User → Role は，全射.

すべての x ∈ User に対して，x ∈ Manager ならば title(x) ∈ Supervisor. 定理

数学によるモデル化



User = { Alice, Bob, … }
Alice ∈ Manager, Bob ∈ Employee
関数 title : User → Role は，全射.

すべての x ∈ User に対して，x ∈ Manager ならば title(x) ∈ Supervisor.　定理

数学によるモデル化の例

- Alice はシステムのユーザである.
  - システムのユーザ全体からなる集合(set) User を用意する.
  - Alice は, User の要素である.
    - Alice ∈ User
- Bob は, (ヒラの)従業員である.
  - 従業員の集合 Employee
    - Bob ∈ Employee
- 従業員は, 全員, ユーザである.
  - ?

数学によるモデル化の例

- 従業員は，全員，ユーザである．
  - すべてのEmployee の要素は，Userの要素である．
    - $\forall x.\ x \in \mathrm{Employee} \rightarrow x \in \mathrm{User}$
  - Employeeは，Userの部分集合(subset)である．
    - $\mathrm{Employee} \subseteq \mathrm{User}$
  - Employeeは，Userの冪集合(power set)の要素である．(冪集合: 部分集合全体からなる集合)
    - $\mathrm{Employee} \in \wp(\mathrm{User})$
- Bobの年齢は25歳である．
  - ?

数学によるモデル化の例

- Bobの年齢は25歳である.
  - 自然数の集合をNとする.
  - UserからNへの関数(function) age を用意する.
    age: User → N
    - age(Bob) = 25
- Alice は, Bob の上司である.
  - 関数 reports : User → User を用意
    - reports(Bob) = Alice
  - 上司が複数いるときは?

数学によるモデル化の例

- Alice は，Bob の上司(のうちの一人)である.
  - User x に，xの上司の全体からなる集合を対応させる関数 reports2 を用意する.
    reports2 : User → $\wp$(User)
    - Alice ∈ reports2(Bob)
  - 2人が上司-部下であるという関係(relationship) Rep を用意する.
    Rep ⊆ User × User
    - (Alice, Bob) ∈ Rep
    - Rep(Alice, Bob)

「数学」だけでは，不十分

定理

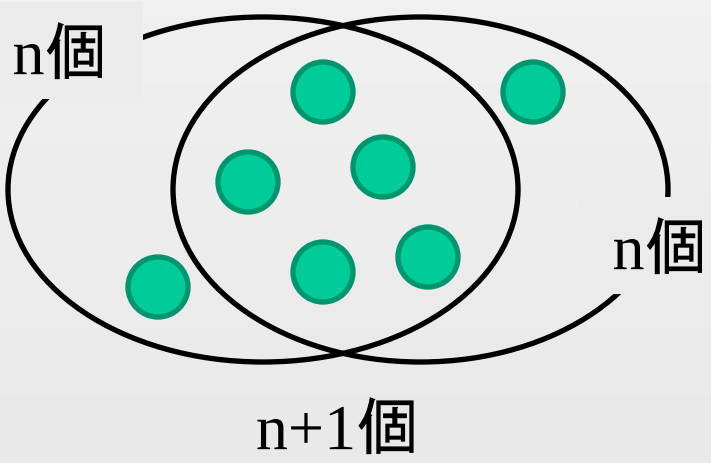
袋の中に碁石が入っている．このとき，常に，すべての碁石の色は同じである．

証明

袋の中に入っている石の数 n に関する数学的帰納法で示す．

[1] n = 1 の場合は，明らか．

[2] n まで OK と仮定して，
n + 1 がOKになることを言えば良いが，右図より明らか．



Q.E.D.　　???

「数学」だけでは，不十分

定理	1 + 1 = 2

証明

0 に何を足しても変わらないので，　1 + 0 = 1　　(A)

次に，x に 1 を足したものは，x に 0 を足したものより 1 つだけ大きいはずだから，x + 1 は，　x + 0 の次の数である．よって，1 + 1 は，1 + 0 の次の数である．

これと (A) より，1 + 1 は，1 の次の数であり，したがって，

1 + 1 = 2

Q.E.D.　　???

形式化が役に立つ

定義	$S: N \rightarrow N, \quad S(0) = 1, \ S(1) = 2$
公理	$\forall x.\ x + 0 = x \qquad \forall y \forall x.\ x + S(y) = S(x+y)$
定理	$1 + 1 = 2$

証明

$$\dfrac{\dfrac{\forall x.\ x + 0 = x}{S(0) + 0 = S(0)} \qquad \dfrac{\dfrac{\forall y \forall x.\ x + S(y) = S(x+y)}{\forall x.\ x + S(0) = S(x+0)}}{S(0) + S(0) = S(S(0)+0)}}{\dfrac{S(0) + S(0) = S(S(0))}{1 + 1 = 2}}$$

体系NKによる証明

公理 $\forall x.\ x + 0 = x$

公理 $\forall y \forall x.\ x + S(y) = S(x+y)$

$$\dfrac{\dfrac{\dfrac{}{S(0) + 0 = S(0)} \qquad \dfrac{\forall x.\ x + S(0) = S(x+0)}{S(0) + S(0) = S(S(0)+0)}}{S(0) + S(0) = S(S(0))}}{1 + 1 = 2}$$

推論規則

$$\dfrac{\forall x.\ P(x)}{P(t)}$$

推論規則

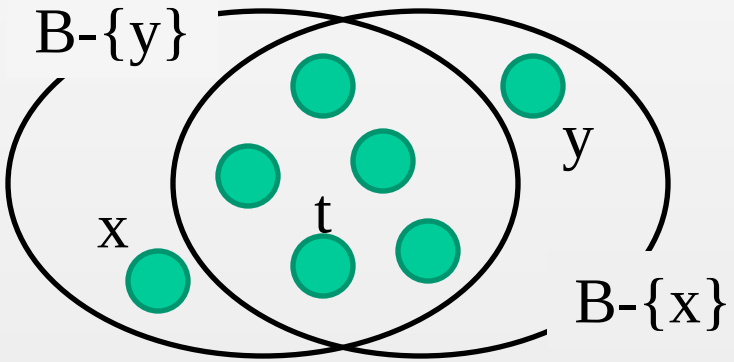
$$\dfrac{s = t \quad P(s)}{P(t)}$$

s, t : 任意の項

ある程度の自動化が可能

碁石問題は…

$$(\forall A.\ |A| = n \rightarrow \forall x,y \in A.\ color(x) = color(y)) \rightarrow (\forall B.\ |B| = n+1 \rightarrow \forall x,y \in B.\ color(x) = color(y))$$



推論規則

$$\frac{\exists x.\ P(x) \qquad \begin{array}{c} P(t) \\ \vdots \\ C \end{array}}{C}$$

公理ではない

この部分の証明が構成できない

$$\frac{\exists z.\ z \in (B - \{x\}) \cap (B - \{y\}) \qquad \begin{array}{c} t \in (B - \{x\}) \cap (B - \{y\}) \\ \vdots \\ color(x) = color(y) \end{array}}{color(x) = color(y)}$$

前半のまとめ

- システムを数学でモデル化する.
  - あいまいさのない記述
  - 仕様をみたす → 数学の定理　　⇒　　証明する
- 数学 ≒ 集合論
  - モデル化には，集合，関数などが必須.
- 数学的記述を形式化して，検証する.
  - 人間の考慮不足による誤りの排除
  - 自動化

後半: モデル検査の準備

例題: エレベータ制御

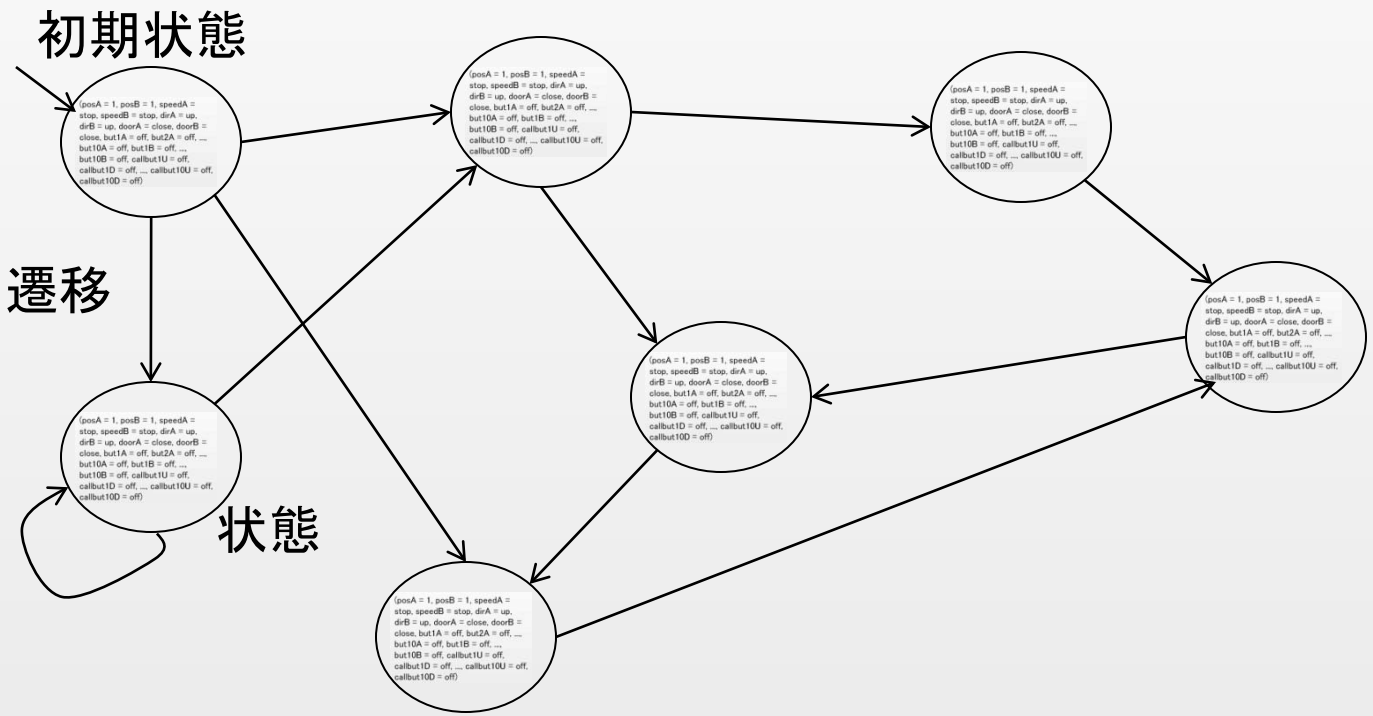
- 10階建てビル
- 2台のエレベータ A, B
- 各階に，上下方向の呼出ボタン
- エレベータ内に，各階用の停止ボタン

- 仕様
  - 動いているときには扉は閉まっている.
  - 扉が空いているうちにエレベータ内のボタンが押されたら，その階に停車する.
  - 各階の呼出ボタンが押されたら，いつかはそちら向きのエレベータがやってくる
  - ・・・

システムの状態

- システムの状態は，以下の組で表現できる:
  (posA, posB, speedA, speedB, dirA, dirB, doorA, doorB, but1A, but2A, ..., but10A, but1B, ..., but10B, call1U, call1D, ..., call10U, call10D)
  - posX ∈ {1, 1-2, 2, 2-3, ..., 10}
  - doorX ∈ {close, open}
  - speedX ∈ { stop, low, high }
  - dirX ∈ { up, down }
  - butNX, callND ∈ { on, off }

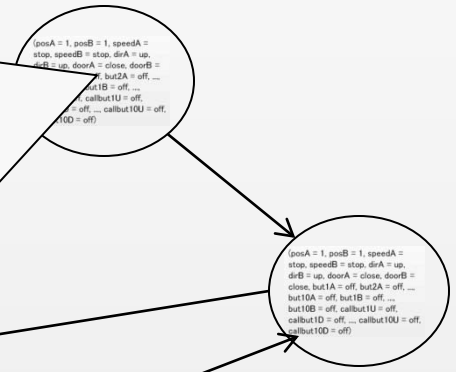
状態遷移系



- 全ての状態と遷移を網羅する

状態遷移系

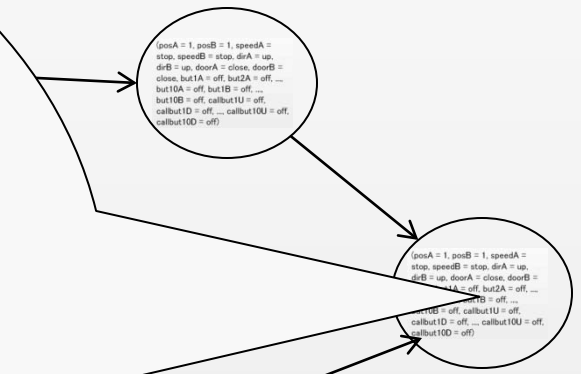
(posA = 5–6, posB = 1, speedA = high, speedB = stop, dirA = up, dirB = up, doorA = close, doorB = close, but1A = off, but2A = off, ..., but10A = off, but1B = off, ..., but10B = off, callbut1U = off, callbut1D = off, ..., callbut10U = off, callbut10D = off)



- 全ての状態と遷移を網羅する

状態遷移系

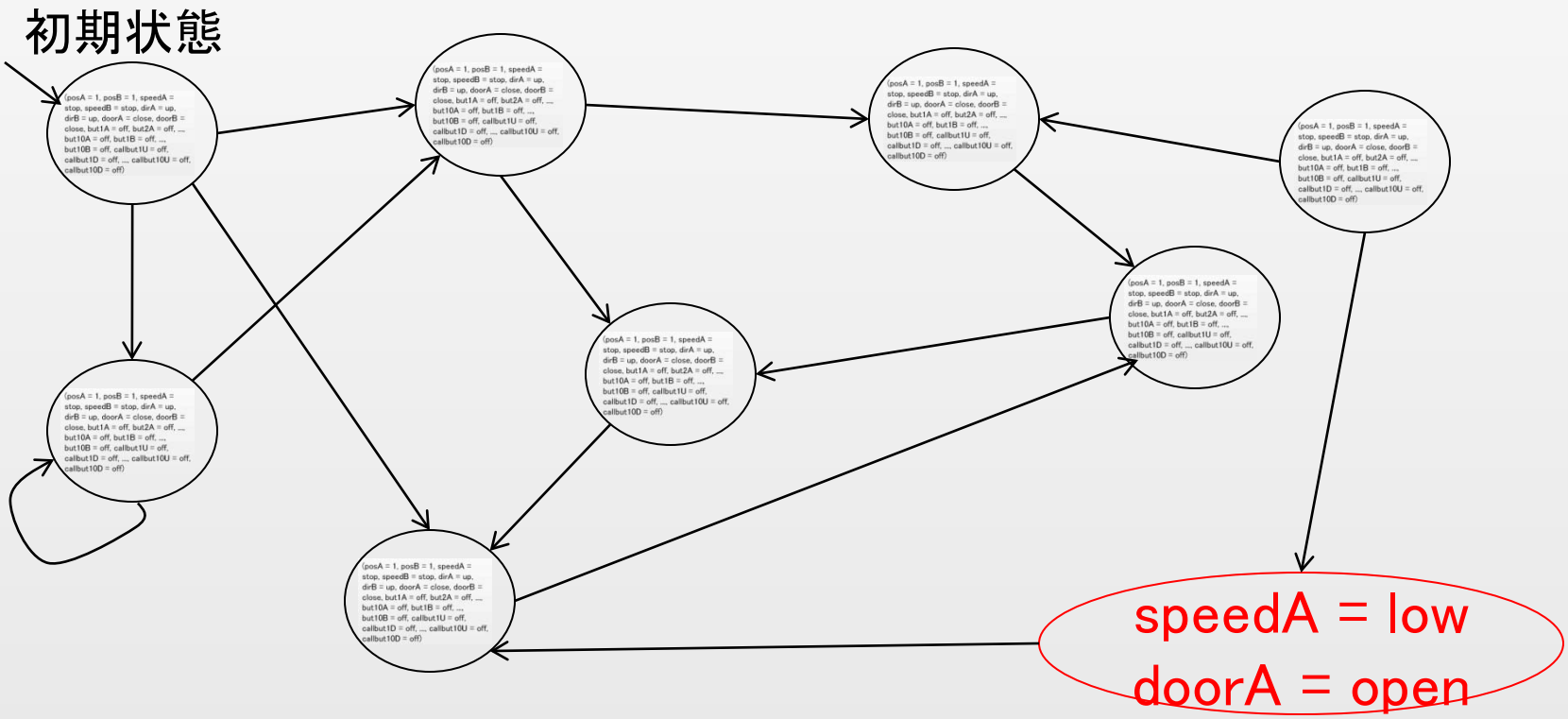
(posA = 6, posB = 1, speedA = high, speedB = stop, dirA = up, dirB = up, doorA = close, doorB = close, but1A = off, but2A = off, ..., but10A = off, but1B = off, ..., but10B = off, callbut1U = off, callbut1D = off, ..., callbut10U = off, callbut10D = off)



- 全ての状態と遷移を網羅する

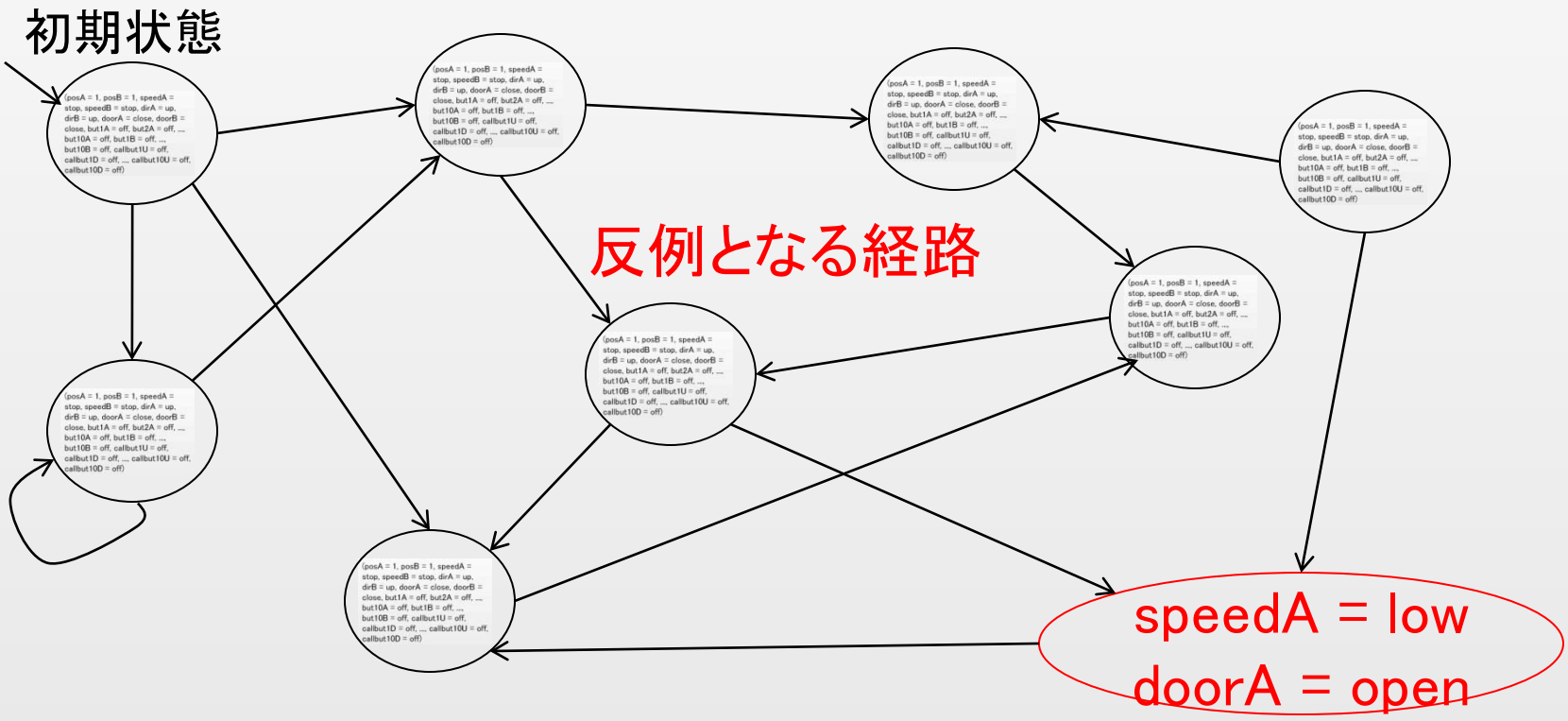
モデル検査

- 動いているときには扉は閉まっている



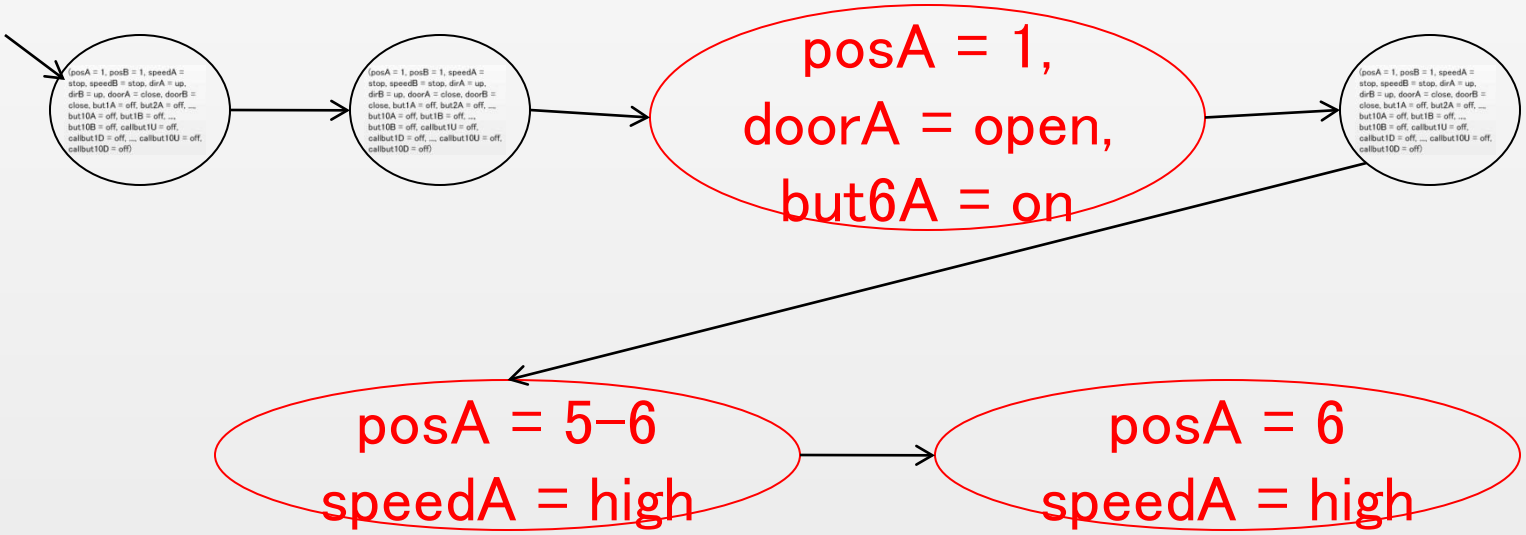
モデル検査: 反例となる経路

- 動いているときには扉は閉まっている



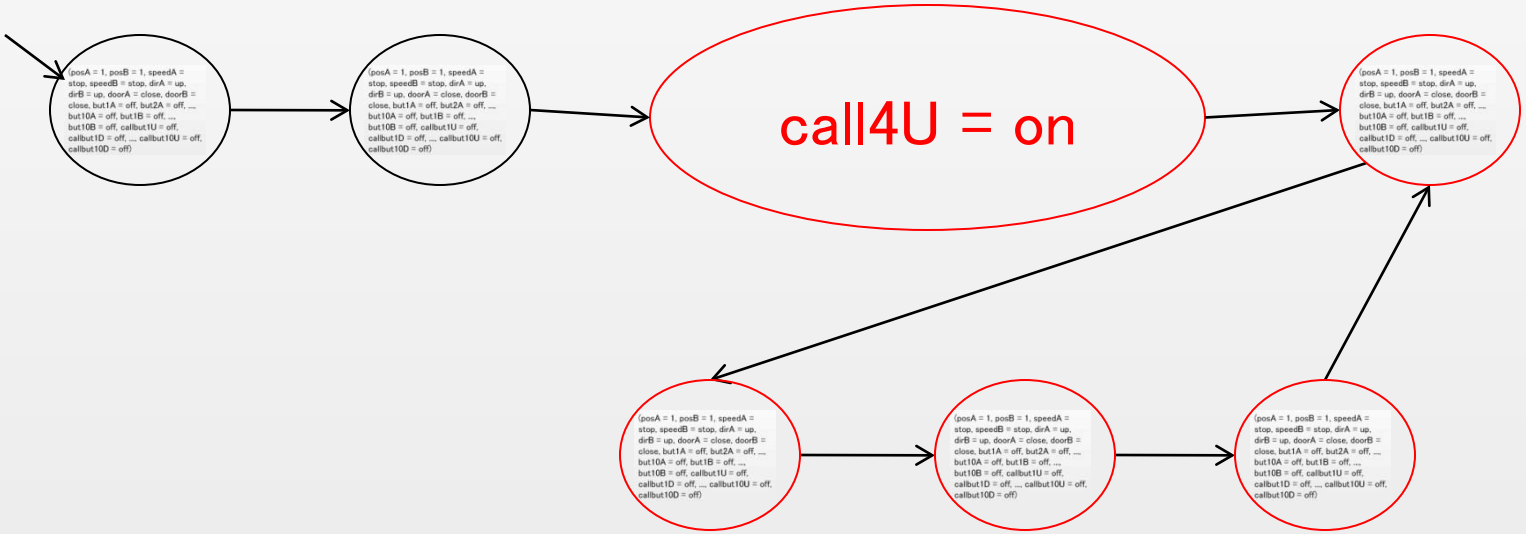
モデル検査: 反例となる経路

- 扉が空いているうちにエレベータ内のボタンが押されたら，その階に停車する.



モデル検査: 反例となる経路

- 各階の呼出ボタンが押されたら，いつかはそちら向きのエレベータがやってくる



このループ中のどこにも
posA = 4, doorA = open や
posB = 4, doorB = open が，無い

時相論理

- 「経路が持つべき性質」を表すための道具
- 時相論理で書かれた性質について，
  モデル検査アルゴリズムが開発されている．
- いくつかの種類がある
  - CTL, LTL, CTL*, ...
- 例:「エレベータが上向きに動いていたら，
  いつかは10階に到達する．」（LTL）

$$G\,(\,dirA = up \rightarrow F\,(\,posA = 10\,)\,)$$

モデル検査の技法

- 基本アルゴリズム
  - 時相論理で表された性質が，与えられた状態遷移系において成立するかどうかを確かめるアルゴリズム
  - 正確に，速く計算できるものが望まれる.
  - 無限語オートマトンの利用

（つづく）

モデル検査の技法 (つづき)

- 探索の高速化
  - Binary Decision Diagram の利用
  - SAT solverの利用 (有界モデル検査)
- 探索空間の圧縮
  - スライシング
  - Partial Order Reduction
  - 抽象解釈
  - Bit State Hashing

後半のまとめ

- モデル検査のアプローチ
  - システムを，状態遷移系として記述する.
  - システムの仕様を，状態遷移系の経路が持つべき性質として記述する.
  - 状態遷移系の全ての経路に対し，指定された性質をチェックする.
- 性質の表現 -- 時相論理
- キーポイント: 効率的なアルゴリズムと状態圧縮
- 技法: オートマトンの利用，BDD, POR, ハッシュ，抽象化，etc.