セッションS2-a
組込みソフト技術者のための形式手法入門 v2

2008/09/05

イーソル株式会社
リサーチ＆コンサルテーションサービス部

目次

1. 形式手法とは
   - フォーマルメソッドの中のモデル検査の位置づけ
2. LTSAについて
   - おそらく一番簡単なモデル検査ツールLTSAの紹介
3. モデル検査の利用方法のアウトライン
   - LTSAの色々な使い方
4. 適用上の問題点
5. RCSのロードマップ

1. 形式手法とは

- 形式手法(Formal Methods)
  - 論理学、集合論、代数学など**数学**を用いて数式的にシステムを記述し、検証を行う手法の総称
    - モデル検査はその様な手法の一つ
  - **論理的に厳密な検証が可能**なため、**高信頼性**が要求されるものに適している
- なぜそうしたか説明したい人が使う
  - 理由を客観的に説明する

1-1. 形式記述と形式検証のタイプ

■ 記述

- ❖ 自然言語でがんばる
  - ▶ 客観的に判断できることが重要
  - ▶ 限界がある
  - ▶ でも全ての基本 (validation)
- ❖ モデルを使う
  - ▶ 条件記述: 形式仕様記述
    - ✓ VDM
    - ✓ Z
    - ✓ OCL
    - ✓ ・・・
  - ▶ 振る舞い記述: 状態遷移系
    - ✓ ステート図
    - ✓ プロセス代数式

■ 検証

- ❖ 定理証明（超ムズイ）
  - ▶ ・・・
- ❖ 充足器
  - ▶ SAT solver
    - ✓ ・・・
  - ▶ SMT solver
    - ✓ Yices
    - ✓ ・・・
- ❖ モデル検査機
  - ▶ SPIN
  - ▶ NuSMV
  - ▶ LTSA
  - ▶ ・・・

ボタン一発系

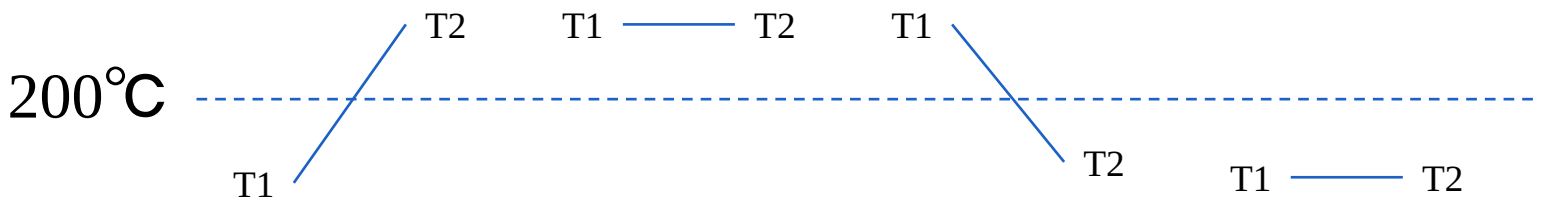
1-2. 形式的と言う意味

- 一定間隔で温度を測定して、温度が上がりすぎないように管理をする話

- 以下の仕様は同じか? ・・・ (レビューで長い議論になったりする)
  - 前回測定温度が200℃以上で，その次は200℃未満，だったら安全である
  - 二回連続して200℃以上だったら危険である

- 論理式で書くと ・・・ (これがモデル化、形式仕様記述)
  - $(T_1 \geqq 200) \rightarrow (T_2 < 200)$ なら安全
  - $(T_1 \geqq 200) \wedge (T_2 \geqq 200)$ なら危険

1-3. 検証: SATソルバー的アプローチ



A **B** **C** **D**

安全 **危険** **安全** **安全**

状態	T1≧200	T2＜200	T2≧200	(T1≧200) ->(T2＜200)	(T1≧200)∧(T2≧200)
A	0	0	1	1	0
B	1	0	1	0	1
C	1	1	0	1	0
D	0	1	0	1	0

1で安全 **1で危険**

同じ事を言っている

1-4. 検証: 定理証明的アプローチ

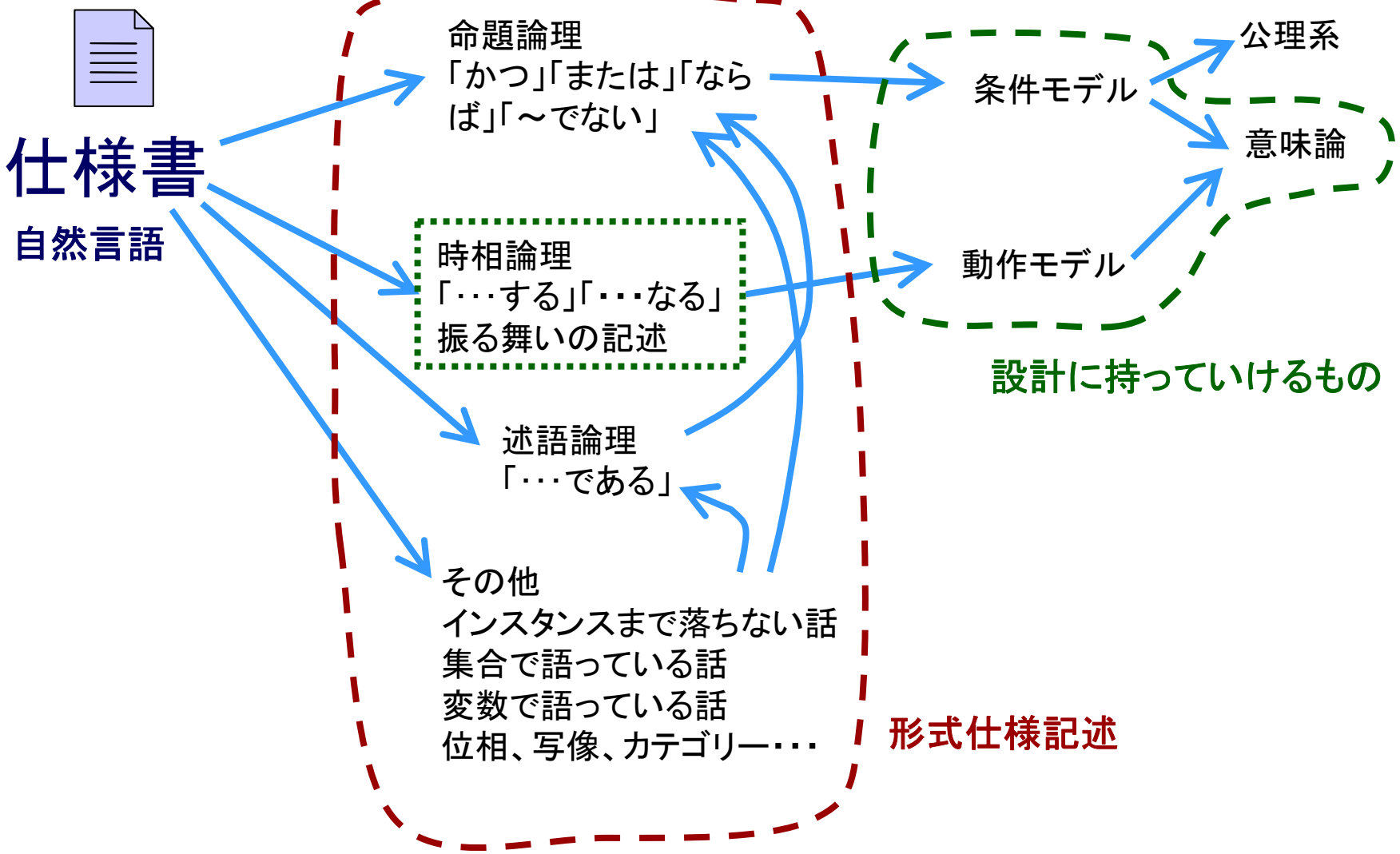
$P = (T_1 \geqq 200)$
$Q = (T_2 < 200)$ とする

(1) P → Q であれば安全
(2) P∧¬Qであれば危険

証明

P→Qは　¬P∨Qと同じ
安全を否定すると危険だから
¬(¬P∨Q)であれば危険
ド・モルガンの定理から、これは
P∧¬Qとなる
よって(1)と(2)は同じ事を言っている

1-5. 仕様書の形式化

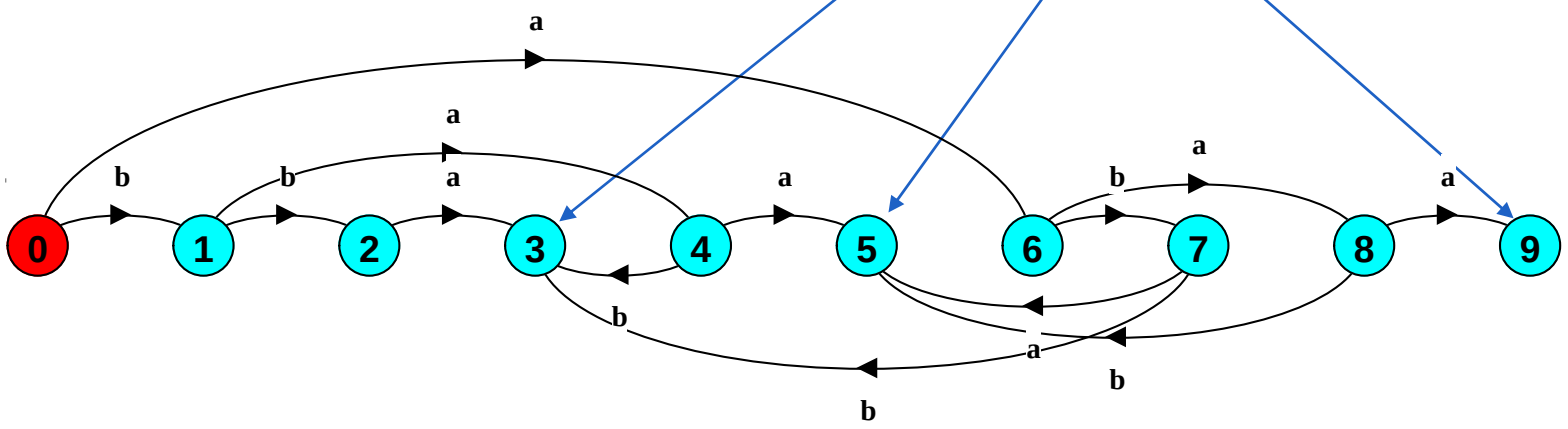


1-6. 動作モデルの例

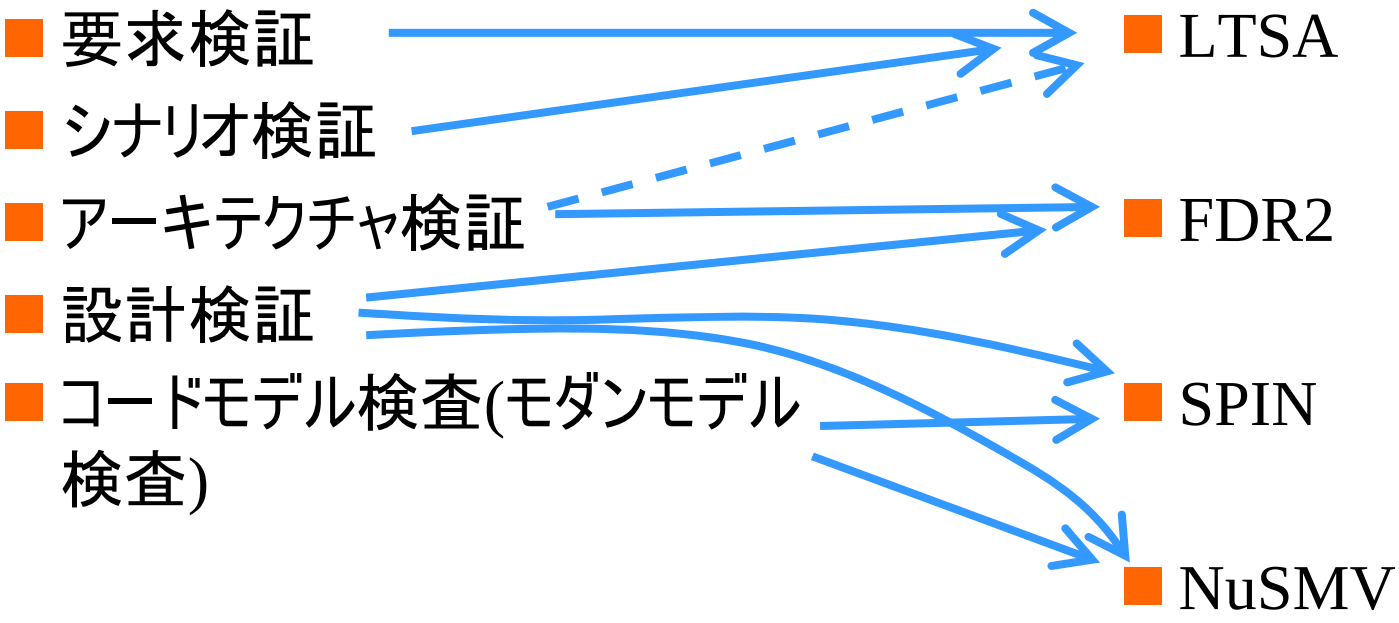
- aが3個とbが2個から、合計で3個取り出す、取り方は

A = (a->a->a->END).
B = (b->b->END).
T = ({a,b}->{a,b}->{a,b}->END).

$$\frac{3!}{1!2!} + \frac{3!}{2!1!} + \frac{3!}{3!} = 7$$



1-7. モデル検査とモデル検査ツール



2. LTSAについて

- フリーのモデル検査ツール
  - 入手先
    - http://www-dse.doc.ic.ac.uk/concurrency/
  - 簡単に使える
    - 動かすことも、モデルを作ることも簡単にできる
    - モデル検査初心者にお勧め

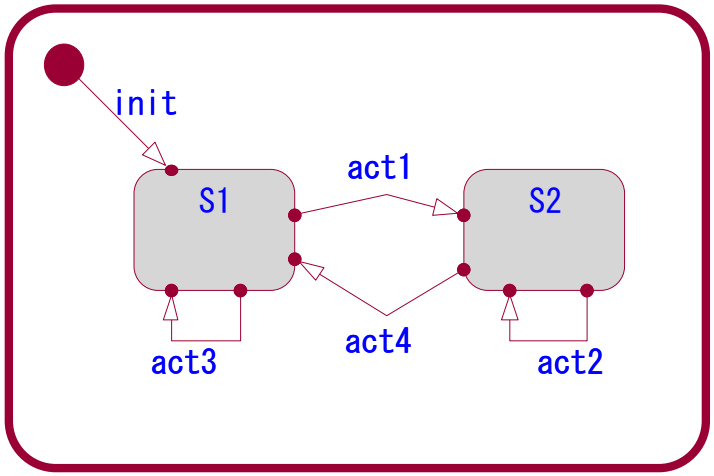
- LTSAで用いる形式
  - テキスト：有限状態プロセス（ FSP ：Finite State Processes）という式を書く
    - ✓ FSP Quick Reference
      - http://www.doc.ic.ac.uk/~jnm/book/ltsa/Appendix-A-2e.html
  - 状態図：ラベル付き遷移システム（ LTS ：Labelled Transition Systems）という状態マシンを生成する

- LTSAの使い方
  - http://www.graco.c.u-tokyo.ac.jp/~tamai/concurrency/slides/
  - http://www.cqpub.co.jp/hanbai/books/33/33441/33441_16syo.pdf
  - http://www.esol.co.jp/rcs/rtos_fw.html
  - http://www.esol.co.jp/rcs/uml_tool.html

2-1. LTSAモデルについて

検査対象

UMLステートチャート



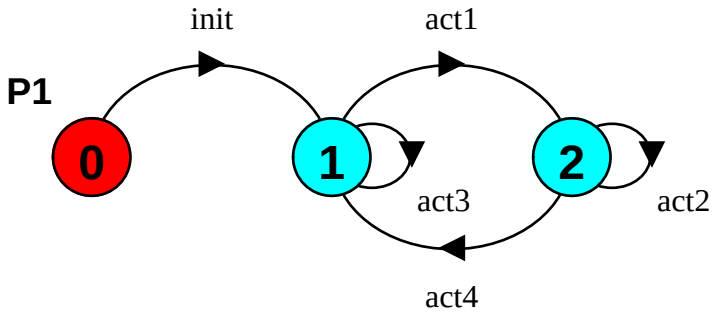
LTSAの入力

FSP形式

P1 = (init->S1),
S1 = (act1->S2 | act3->S1),
S2 = (act4->S1 | act2->S2).

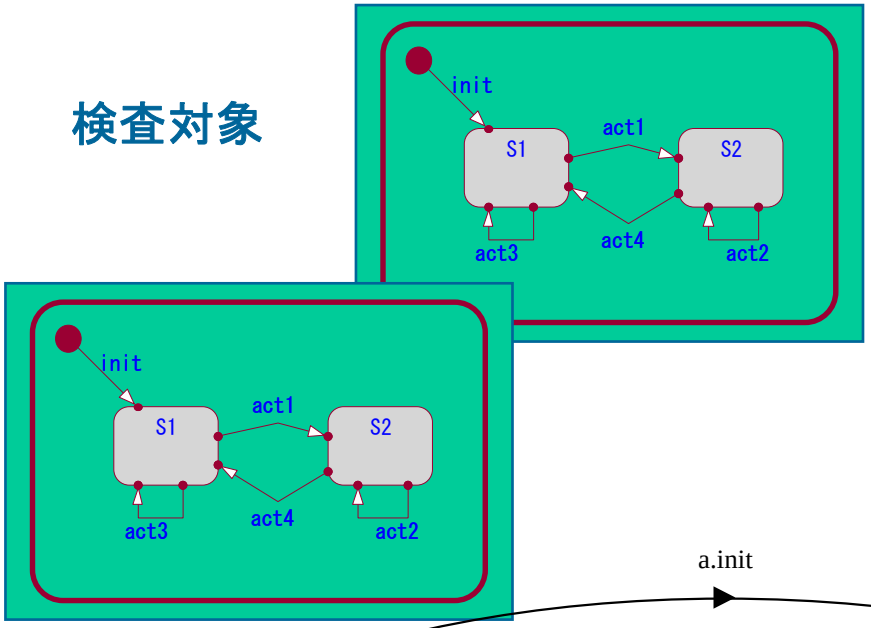
LTSAの出力

LTS形式



2-2. 状態マシンの合成

検査対象

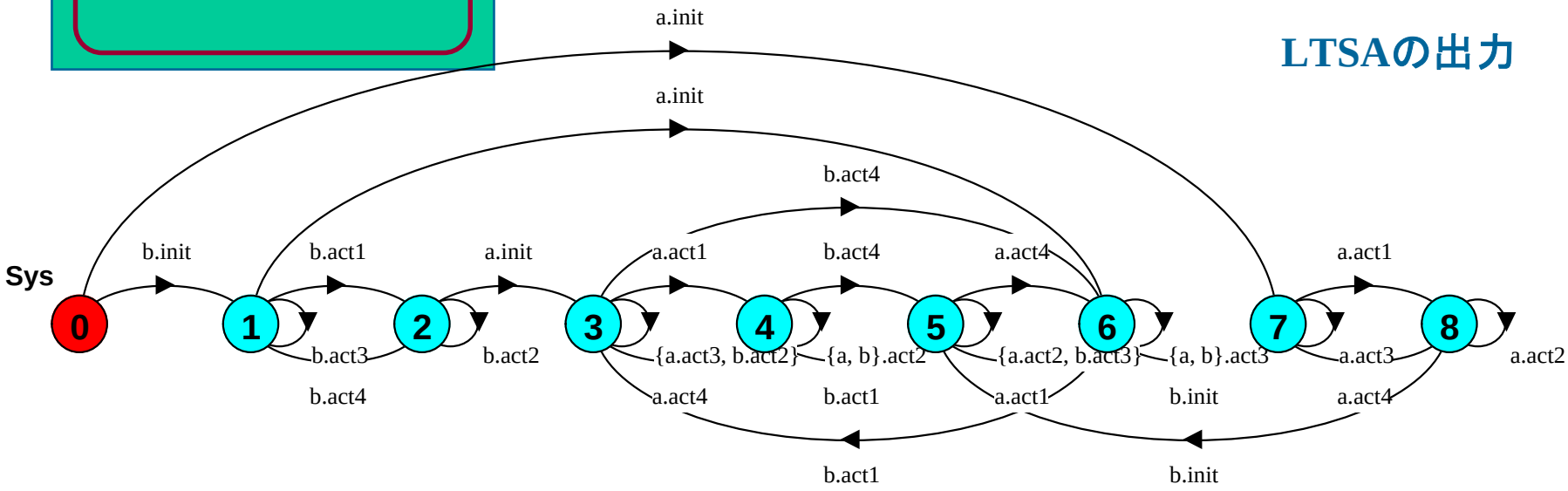


LTSAの入力

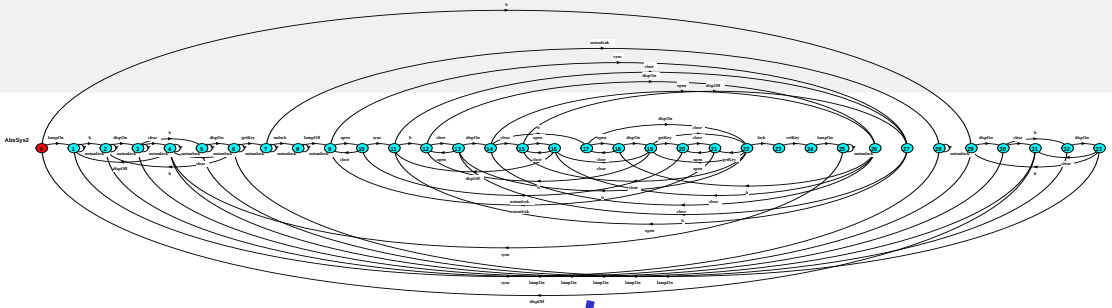
FSP形式
P1 = (init->S1),
S1 = (act1->S2 | act3->S1),
S2 = (act4->S1 | act2->S2).

||Sys = (a:P1 || b:P1).

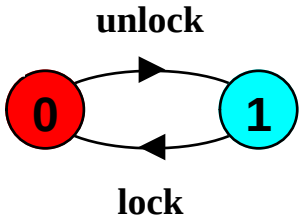
LTSAの出力



2-3. 動きの抽出



[](lock → <>unlock)
[](unlock → <>lock)



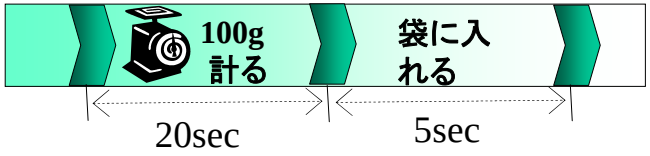
弱模倣等価

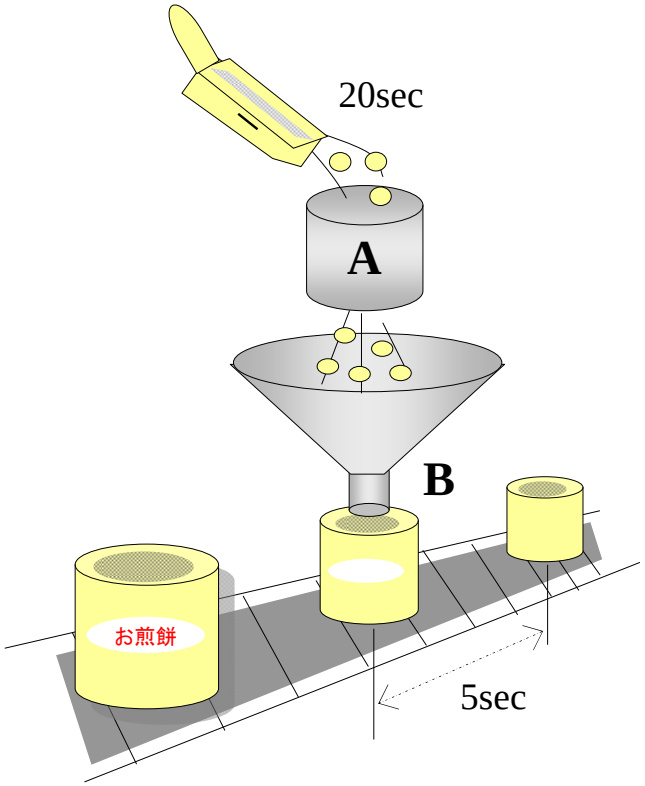
- 特定の動作の関係のみを抽出して見ることもできる。
  - この機能はタスク分割に応用できる。
  - テストケース生成
  - テストドライバの生成
  - ログ解析器の生成

3.モデル検査の利用方法のアウトライン

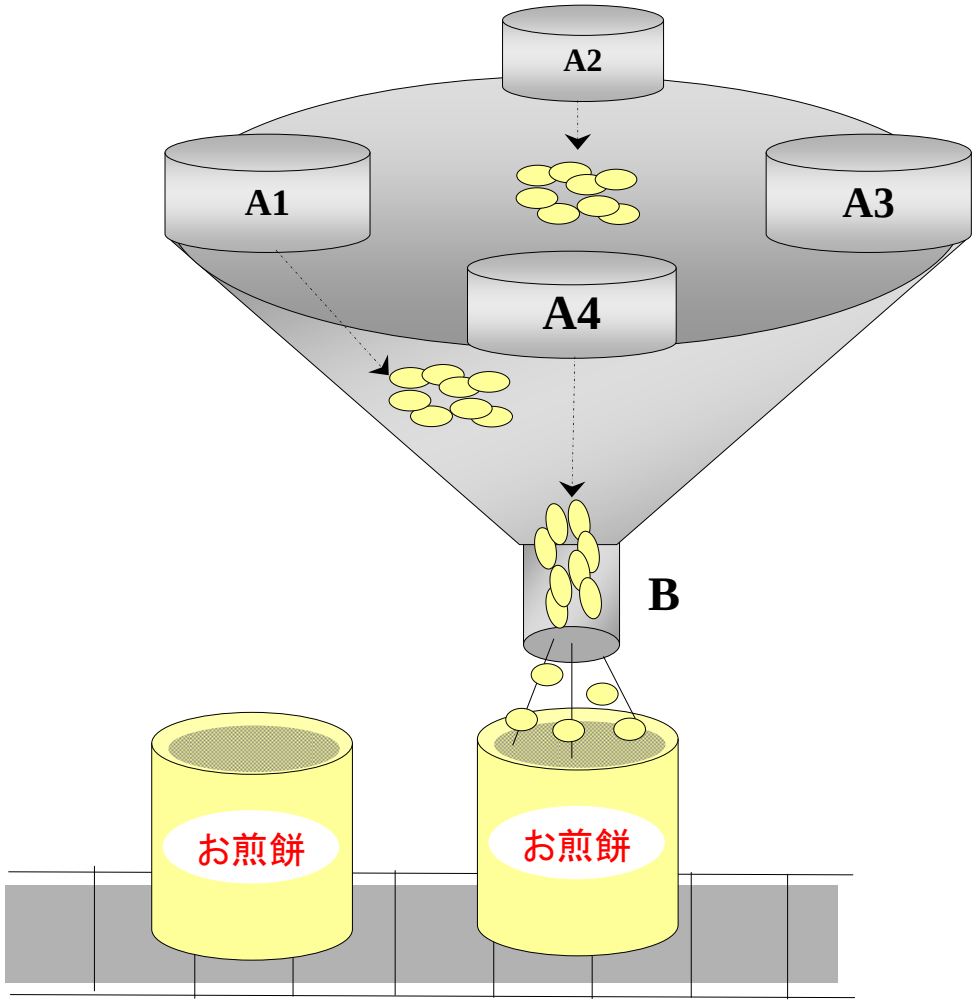
a. 制御システムの例
b. タスク検証の考え方
c. シナリオ検証
d. 形式仕様記述との連携

3-a. 制御システムの例



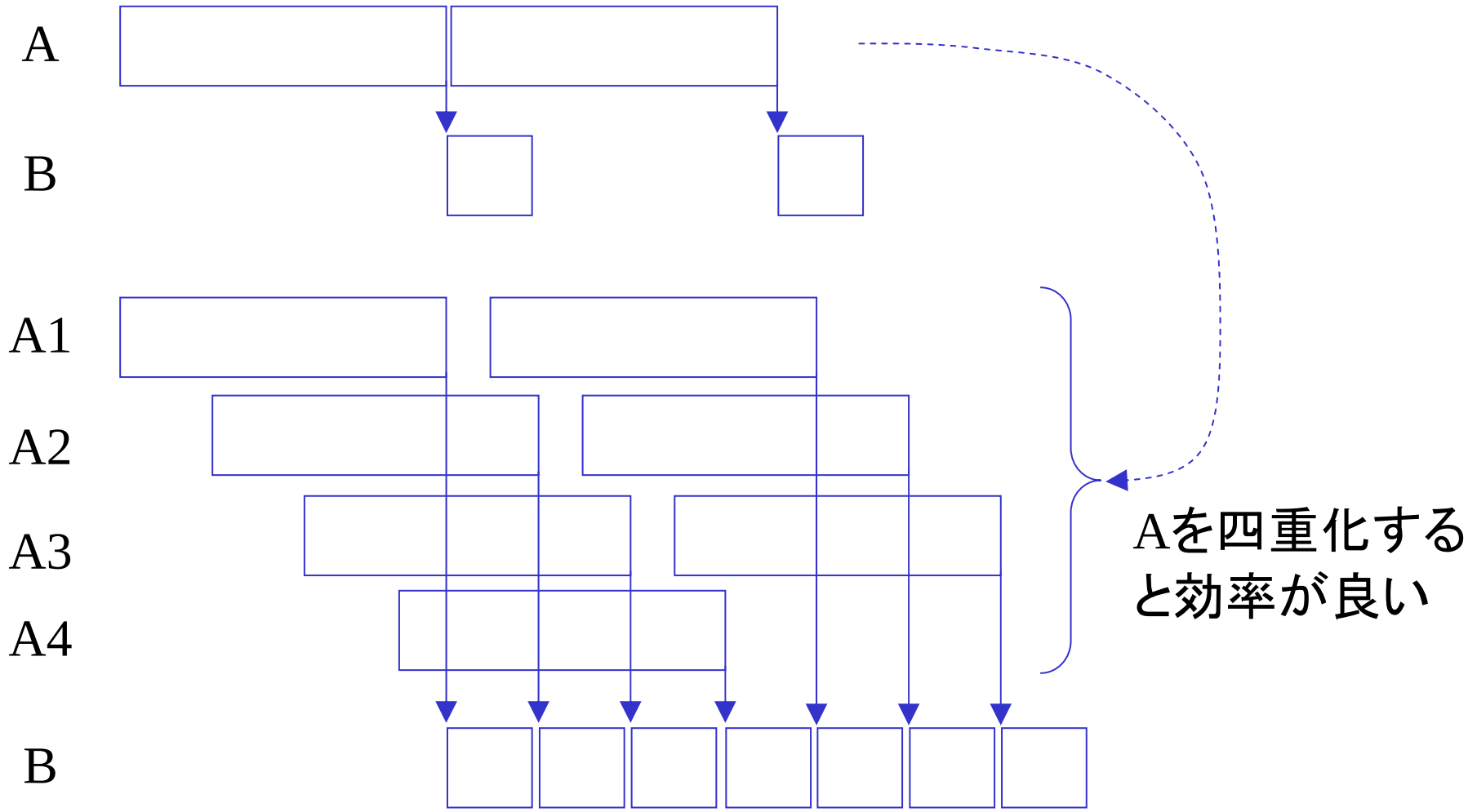


20sec に1ヶ

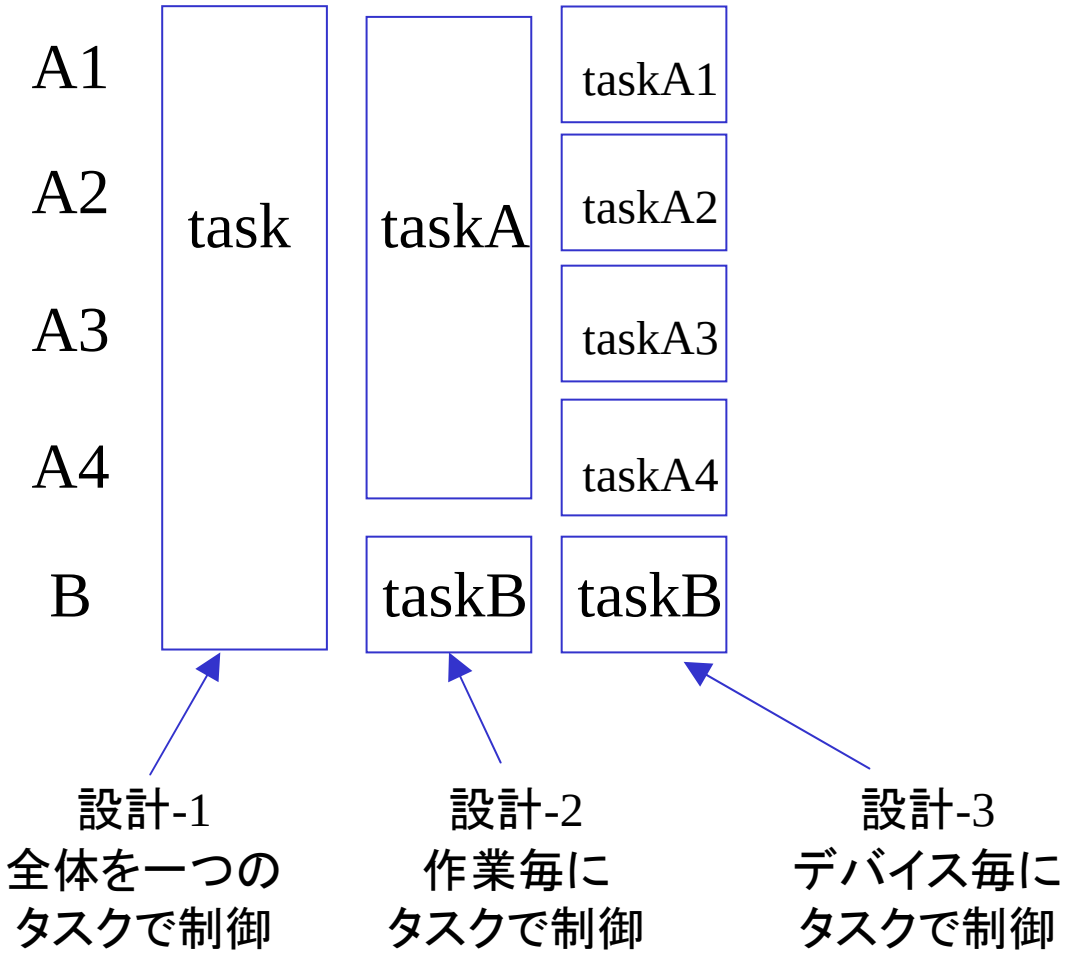


5sec に1ヶ

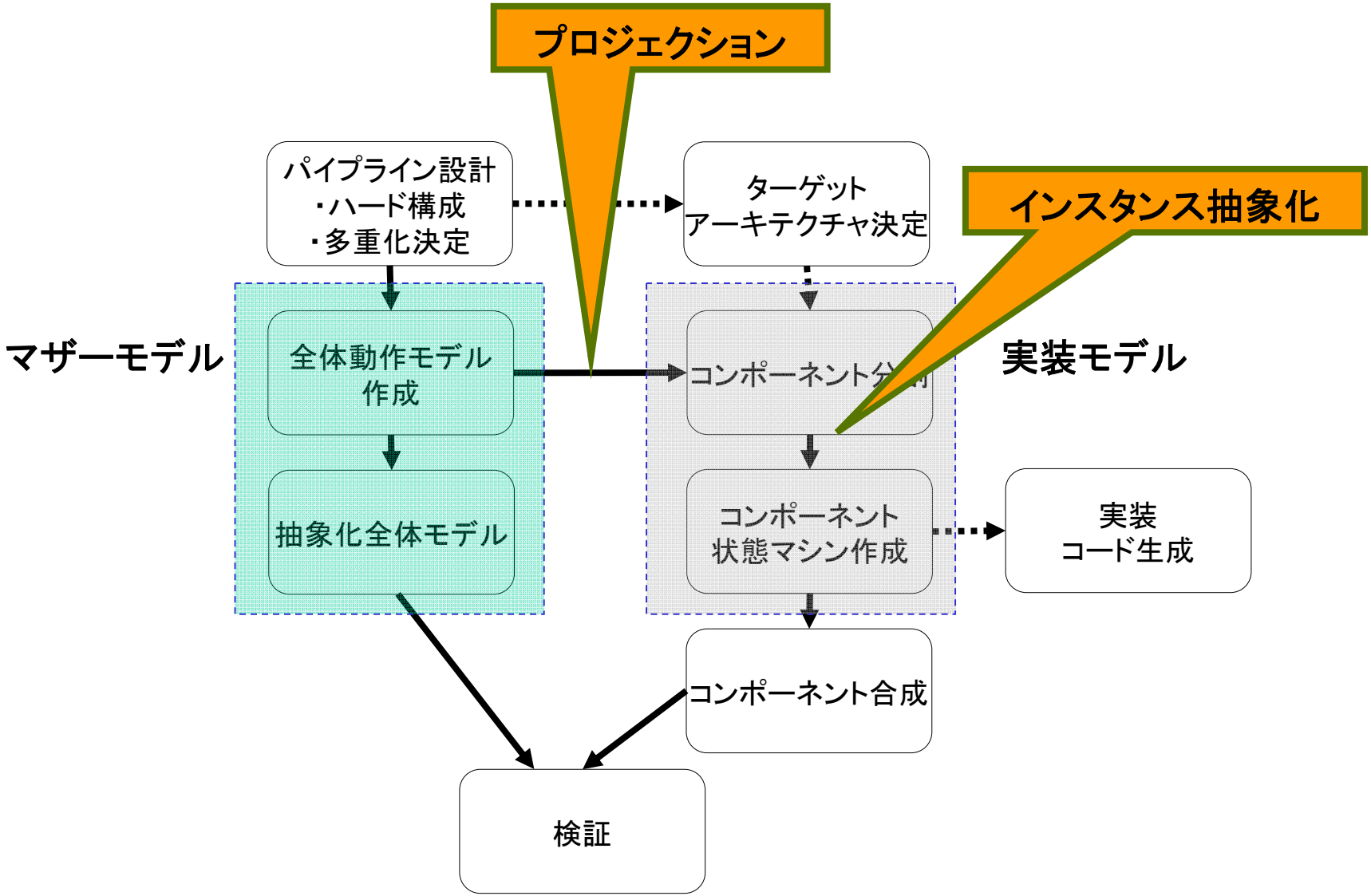
3-a-1. 動作タイミング



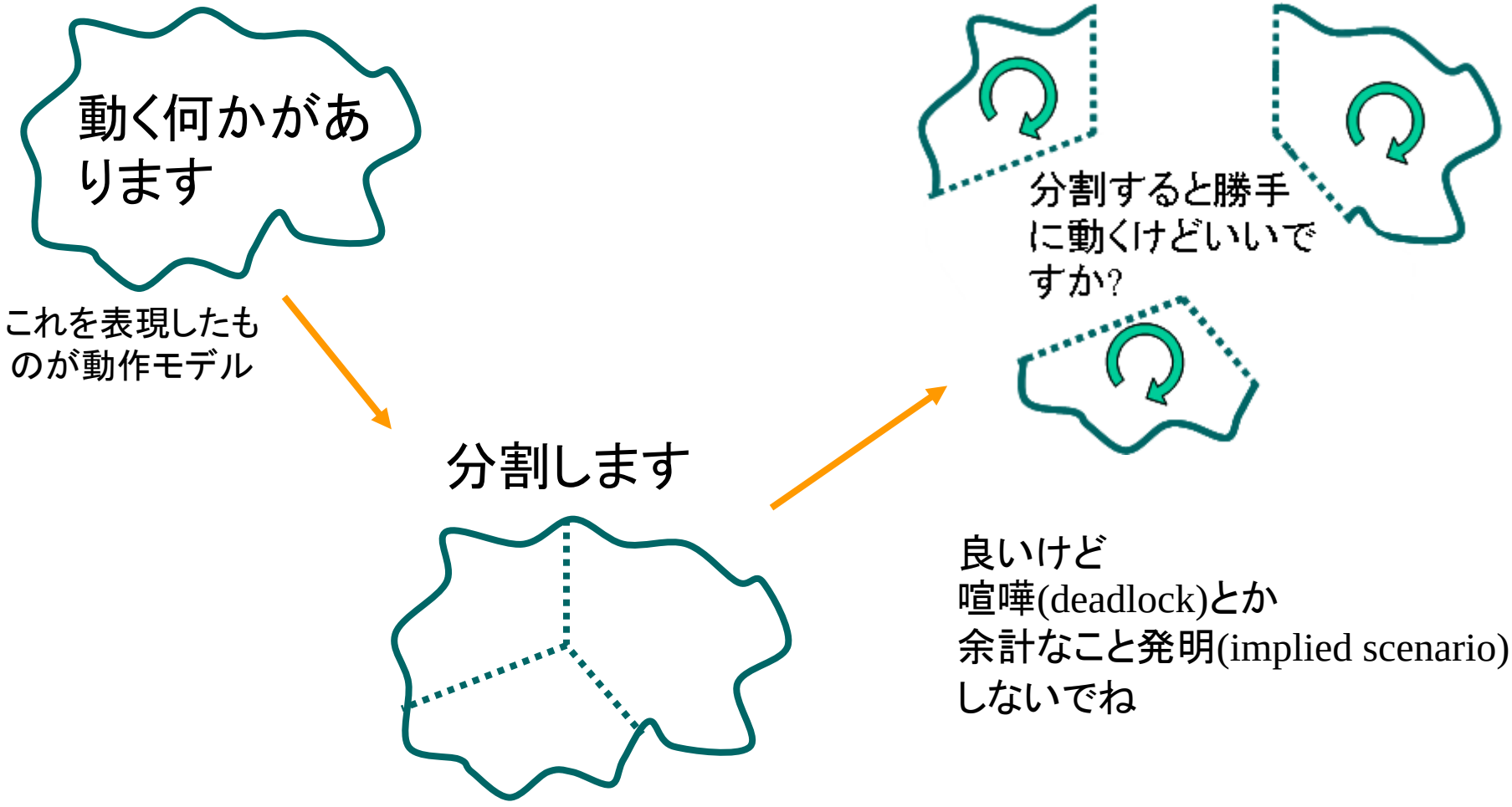
3-a-2. ターゲットアーキテクチャの選択



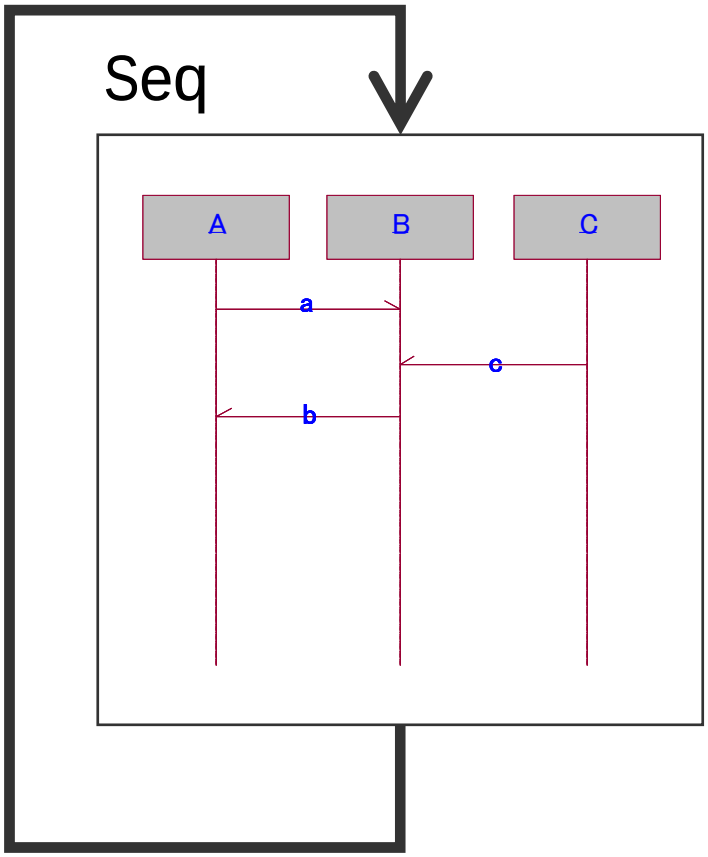
3-a-3. 設計手順の概要

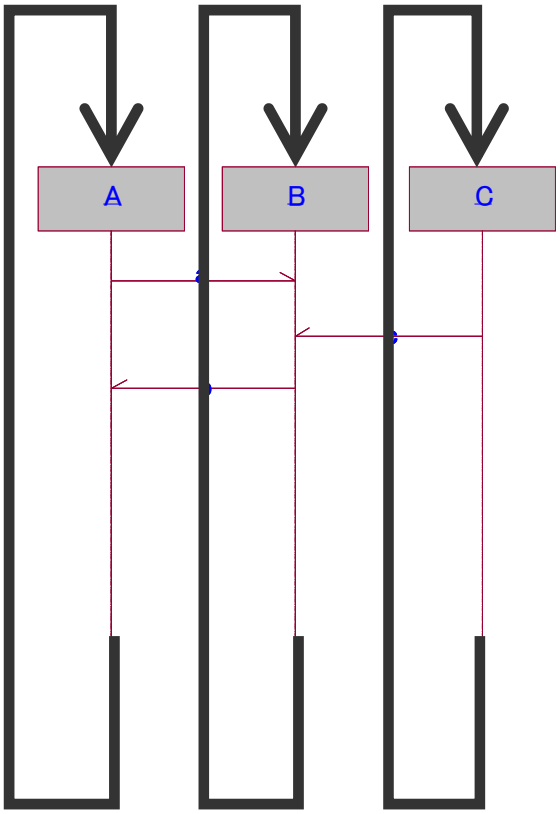


3-b. タスク検証の考え方



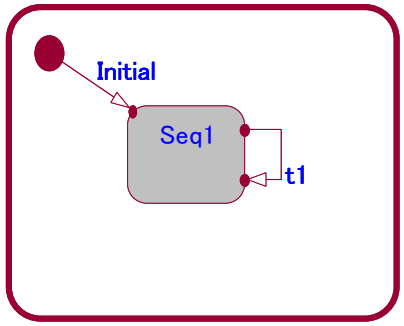
3-b-1. 二つの動きの違いは?

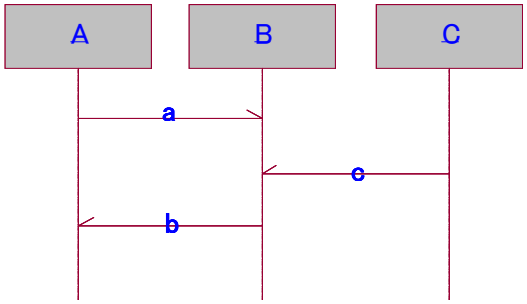




(A||B||C) ⊆ Seq

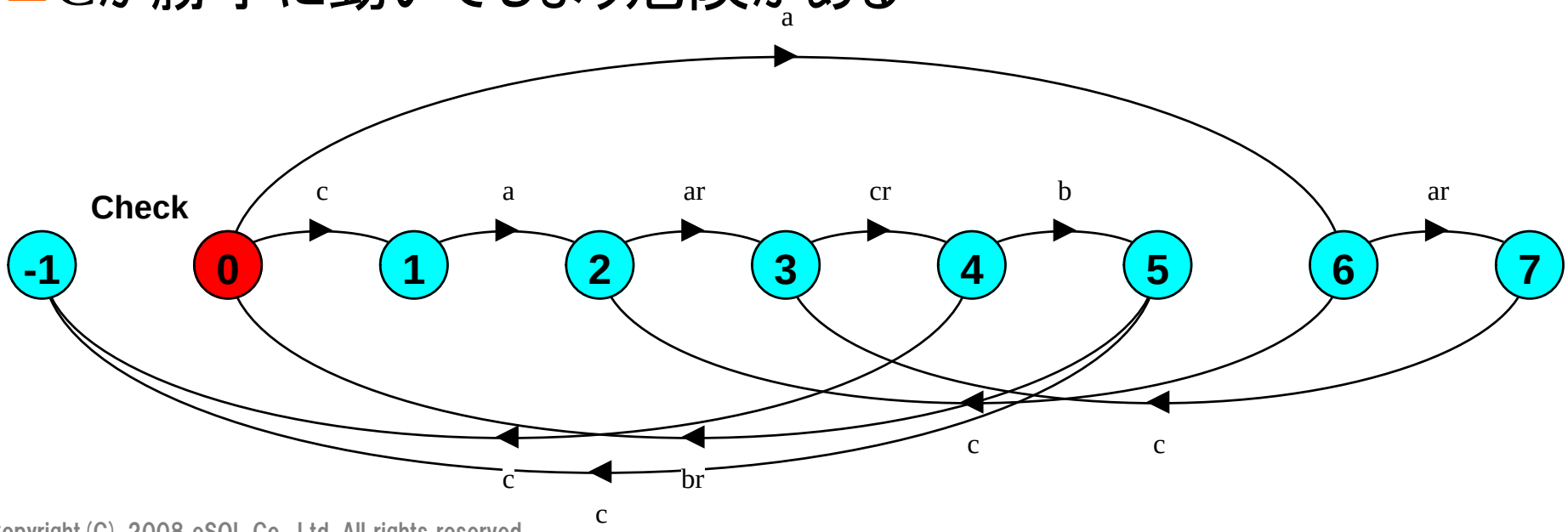
3-b-2. 検証例





Seq1

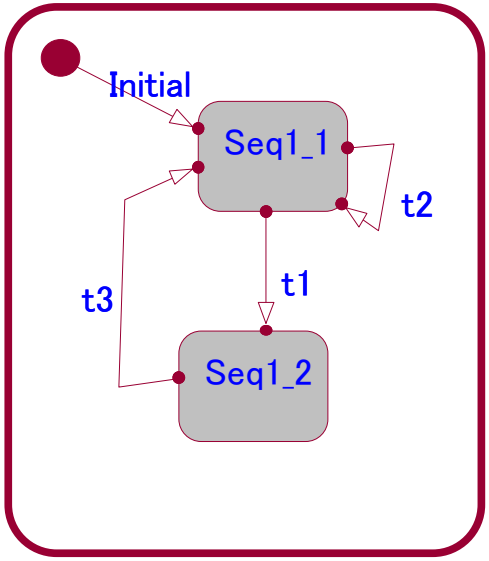
■Cが勝手に動いてしまう危険がある

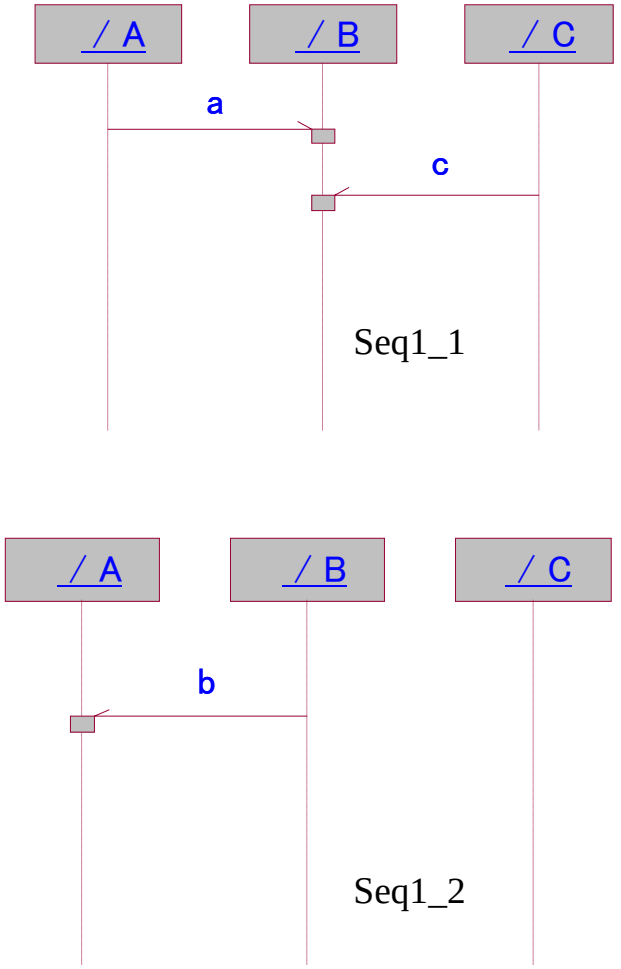


3-c. シナリオ検証

■状態0に接続する場合

- ❖c→a→c→aを繰り返すシナリオがある

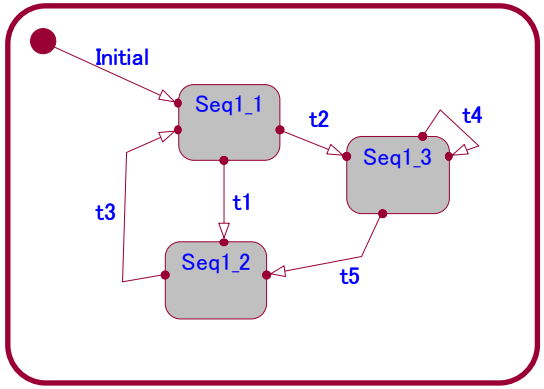


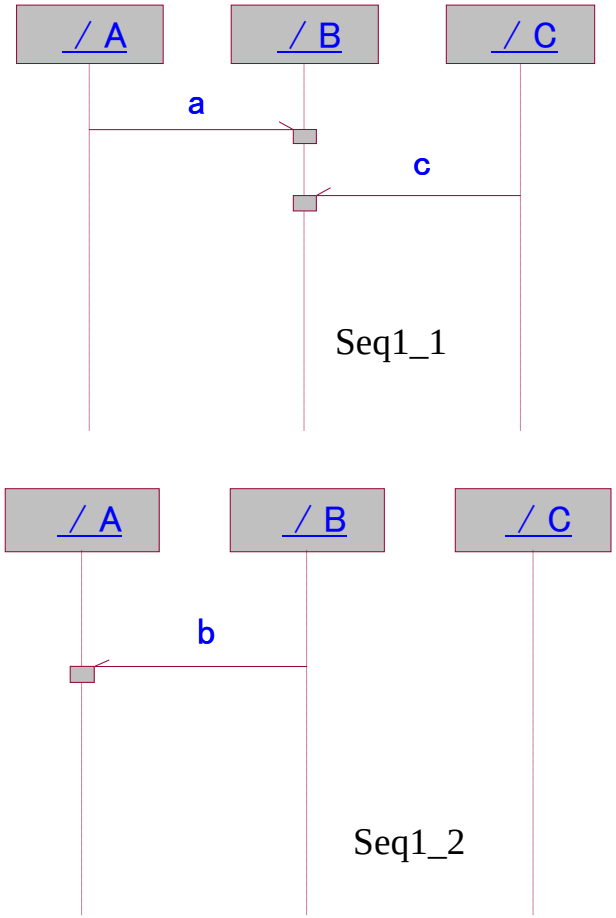


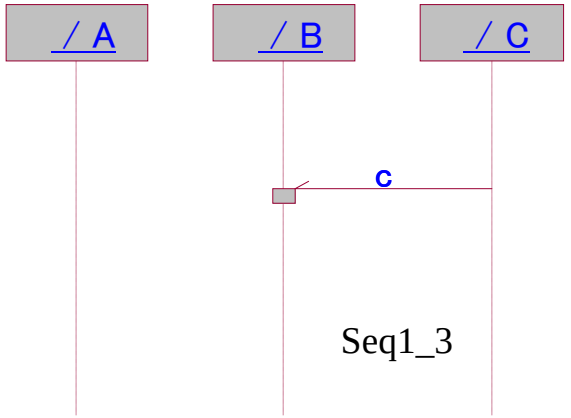
3-c-1. シナリオ検証2

■状態3に接続する場合

❖c→c→c→cを繰り返すシナリオがある





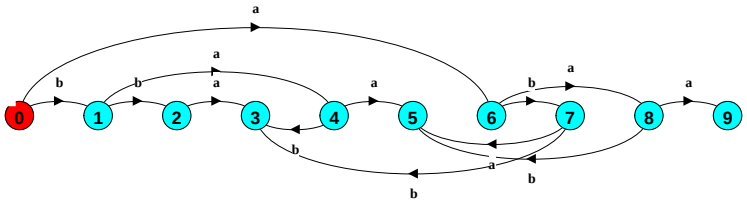


3-d. 形式仕様記述との連携

仕様書

操作的記述
形式仕様記述

動作モデル



条件の記述

- 不変条件
- アクション条件
  - 事前条件
  - 事後条件

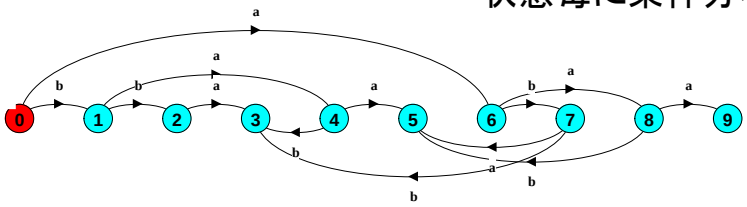
検証

実際には表示的記述が不足するので補いながら検証をする

3-d-1. 検証のレベルを上げる

FSP動作モデル

状態毎に条件分析が可能



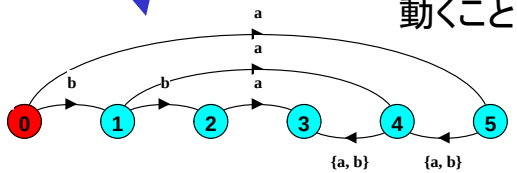
条件の記述

- 不変条件
- アクション条件
  - 事前条件
  - 事後条件

どのレベルで記述するのかまず決める。実際の条件は、
　何を証明したいのか
　何を保証したいのか
等を考えながら作り出すことが多い。

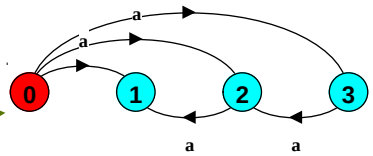
模倣等価

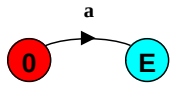
外部観測上区別不能なレベル
何処に組込んでも同じように
動くことを保証したい



トレース等価

出来ることが同じものは同じ
チャンと動けば同じ機能を
提供する。外部環境を指定
して再利用



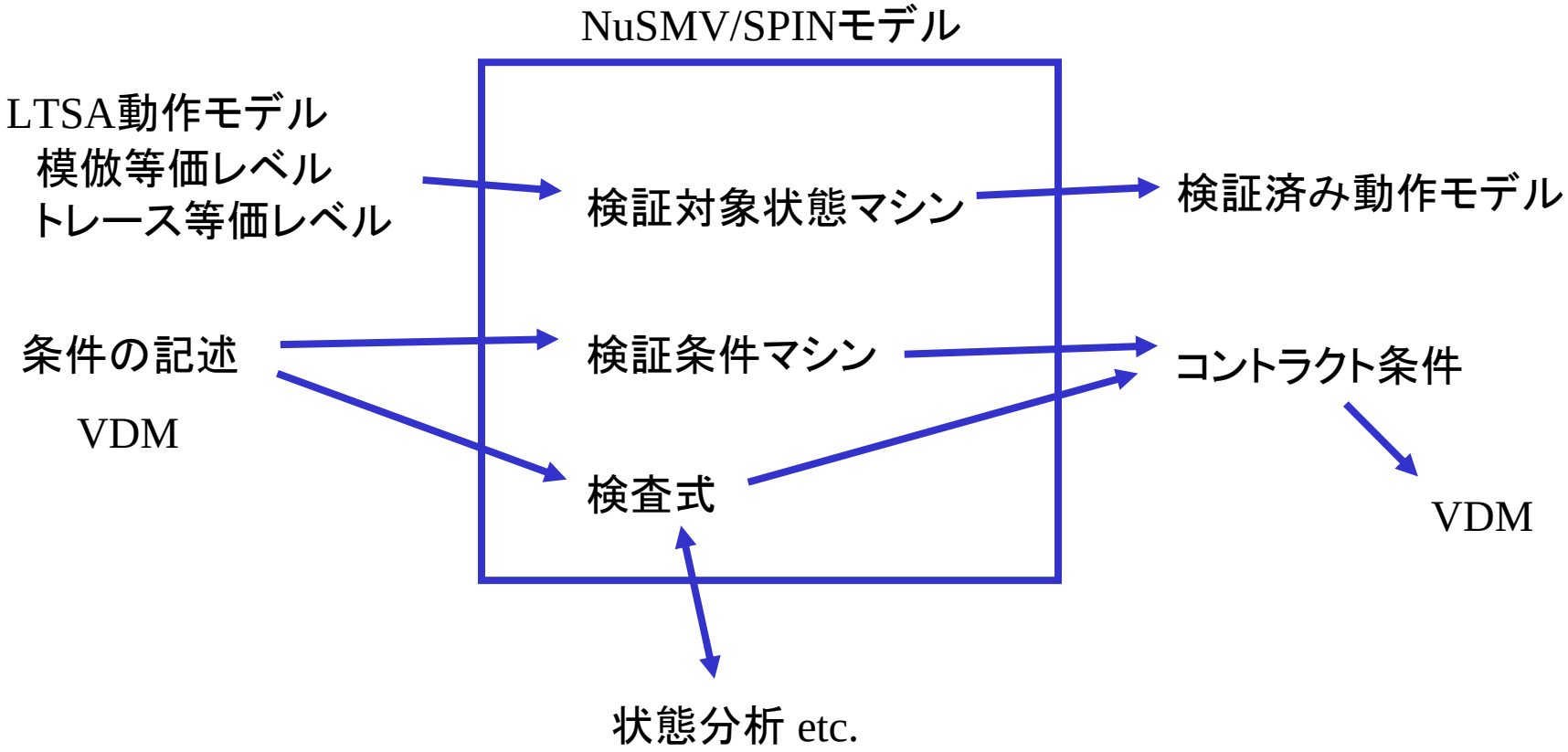


単なる関数呼出しレベル

単体テスト条件

3-d-2. LTSA以外のツール/手法との連携

解析的に条件を確定するのは困難なので、モデル検査により試行錯誤で決定する。



4. 適用上の問題

- 仕様書をモデル化して検査する
  - 仕様書は現状のソースコードと一致していない
- ソースコードをモデル化すると状態爆発
  - リバースして抽象化
- 検証用モデルのデバッグにはまってしまう
  - MDAではないので、モデルのデバッグには意味がない場合が多い
- モデルはできたが検査式を作れない
  - 要求仕様が無い
  - 何を検査したらいいのか

5. RCSのロードマップと方針

- 方針
  - ボタン一発系の技術しか普及しないだろう
  - ツールサポートを重視
  - 合わせ技
- ロードマップ
  - モデル検査技術のタスク設計への適用を研究・実践してきた
  - 次のステップ
    - 動的構造のリバース →モデル検査
    - VDMによる形式仕様記述 + SAT/SMT solver
    - 自然言語からの動作モデル生成
    - 自然言語仕様書の形式化

イーソル株式会社

リサーチ＆コンサルテーションサービス(RCS)部

〒164-8721 東京都中野区本町1-32-2 ハーモニータワー

Tel 03-5302-1360　Fax 03-6203-8339

E-mail : consult@esol.co.jp

URL: http://www.esol.co.jp/rcs/index.html