ケーススタディで学ぶ仕様の書き方

2018.03.08

川口　順央

@masateruk

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

仕様とは

端的にいえば、

そのシステムは何か？を表したもの

- 何か？を記述するにもいろいろな抽象度がありうる
- どの抽象度が適切か？
  - 抽象度を決めるには、なぜ仕様を書くのか？を考えると良い

なぜ仕様を書くのか？

端的にいえば、

システムを作る前にそのシステムは何か？を知るため

なぜ知りたいのか？

端的にいえば、

そのシステムがもたらす効果で何かを得たいと期待するから

システムに関わる人がもつ期待

- そのシステムを使う人は、そのシステムを使うことで自分の問題を解決したい
- そのシステムを売る人は、そのシステムがもつ魅力でもって売り上げをあげたい
- そのシステムのサポートをする人は、既存のシステムに比べてサポートの工数を削減したい

この写真 の作成者 不明な作成者 は CC BY のライセンスを許諾されています

期待から生じる関心や疑問

- そのシステムを使う人は、そのシステムを使うことで自分の問題が解決できるのか？
- そのシステムを売る人は、そのシステムは売り上げをあげられるほどの魅力を持つのか？
- そのシステムのサポートをする人は、既存のシステムに比べてサポートの工数は少ないか？

仕様の抽象度は、“ステークホルダーが観測できる”

- 仕様は、ステークホルダーの疑問に答えられる抽象度で記述する
- ステークホルダーが観測できるかどうかで判断するとわかりやすい

ステークホルダーの疑問に答えられる
＝ ステークホルダーが関心がある
≒ ステークホルダーが観測できる

ふたたび、仕様とは

仕様とは

ステークホルダーが観測できるシステムの特性を記述したもの

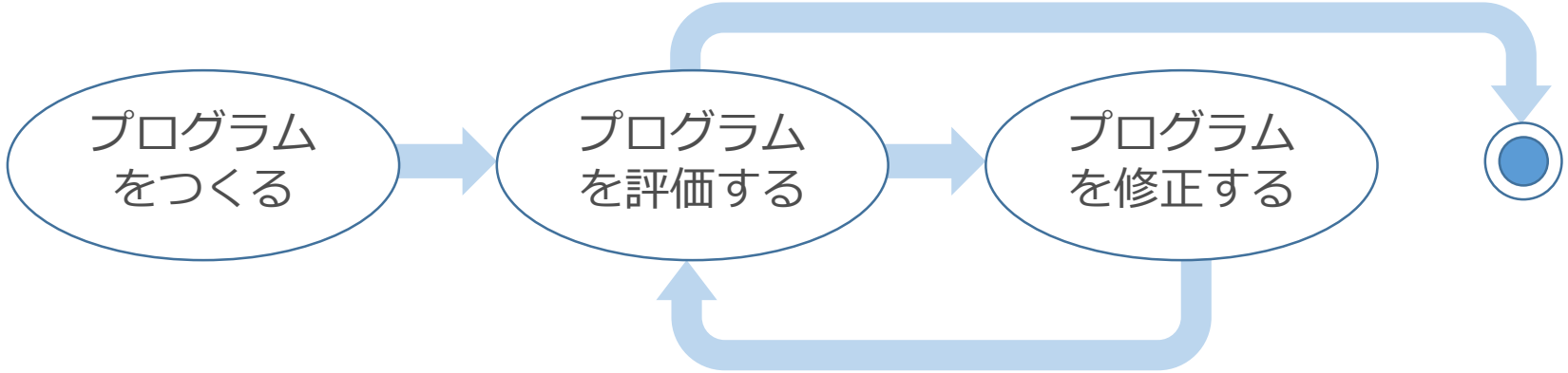
なぜ作る前に知りたいのか？

早い段階でシステムを理解したいから

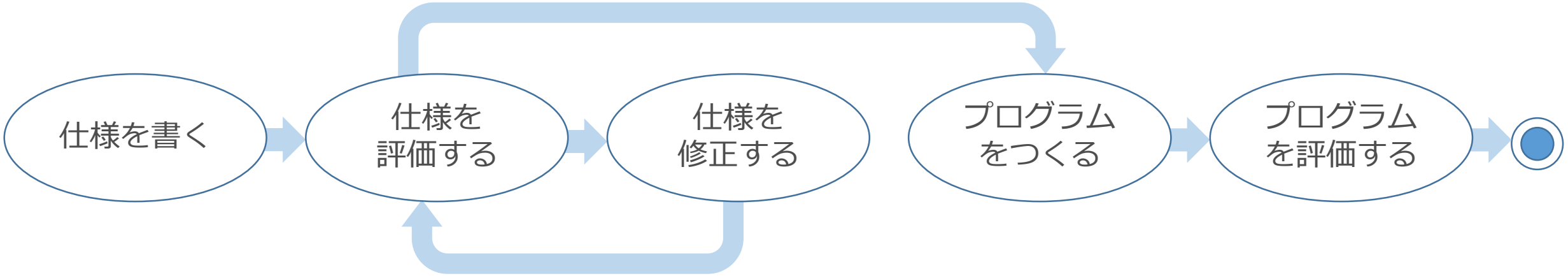
- システムを作る時間より、仕様を記述する時間の方が短い
- 作る前に仕様を確認することで、誤った仕様に基づいて作る無駄を小さくできる

単純なモデルでの工数比較

(A) 仕様を書かない場合



(B) 仕様を書く場合



単純なモデルでの工数比較

- プログラム、仕様のそれぞれの修正が一回として(A) – (B)を計算すると、

(A) – (B)
= （<プログラムを修正する> + <プログラムを評価する>）
- （<仕様を書く> + <仕様を評価する> * 2 + <仕様を修正する>）

- 差分としてあらわれる項の大小関係を考えて仕様段階でどこまで議論するか・仕様に書くか決める

工数比較して仕様に書くか決める

- 根幹となる機能やアーキテクチャに関わる機能など、変更になると<プログラムを修正する>工数が大きくなりそうなものは、仕様段階で議論して<仕様に書く>
- 画面上のGUI要素については、ガイドラインを決めておけば、各画面でのGUI要素の配置やアイコンの絵柄の修正については<プログラムを修正する>工数を小さくできるので、仕様にはどのGUI要素があって、それが何を表すか、それで何ができるかは書くが、細かい配置や絵柄については記述しないで<仕様に書く>工数を小さくする
- 営業チームが<仕様を評価する>（≒仕様を読む）のは時間かかる、または、難しいので、<プログラムを修正する>工数を小さくするために大雑把な方針については合意をとっておくが、評価はデモで行う

ソフトウェアシステムの仕様は何を書くのか？

ソフトウェアシステムの仕様では、おもに振る舞いを書く

- ソフトウェアの価値は、“動く”ことにある
- ステークホルダーは動き、すなわち、振る舞いに関心をもつ
- 振る舞いのほかには、性能や保守性など非機能要求と呼ばれるものも書く

仕様とは何か？ – まとめ

- 仕様とは、そのシステムとは何か？を規定したもの
- 抽象度は、ステークホルダーが観測できるもの
- ソフトウェアシステムの場合は、振る舞いと非機能要求を書く

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

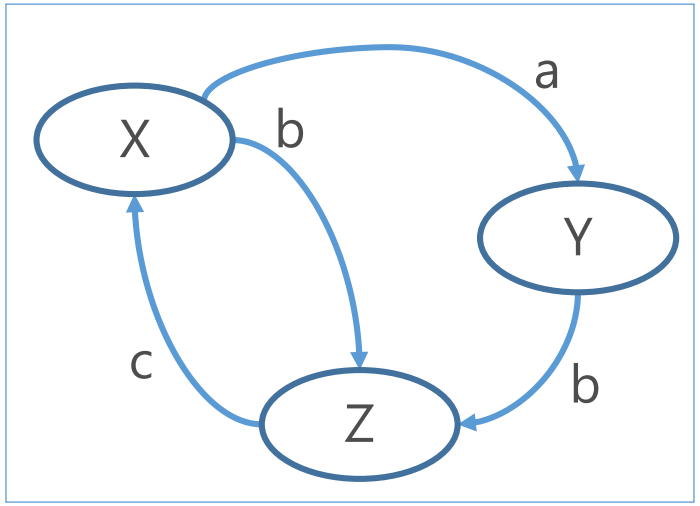
振る舞いを表す記法

- **シーケンス図**
  - 振る舞いのひとつの例に過ぎず網羅的ではない
- **コラボレーション図**
  - （主に内部の）構成要素の相互作用を書いたものである。外から見てどう振る舞うか？と抽象度が異なる
- **状態遷移図**
  - 外からみた振る舞いを網羅的に記述可能

状態遷移図から状態遷移モデル

- 状態遷移"図"である必要はなく、状態遷移を表すものであればよい。
  こういったものを総称して**状態遷移モデル**という

状態遷移モデル ー 図と表



状態遷移図

遷移先が明確で、手で追ってシミュレーションしたりするのに便利

	X	Y	Z
a	Y		
b	Z	Z	
C			X

状態遷移表

一覧性がよい。網羅性検査がしやすい

非機能要求の記述

- 非機能要求は、自然言語で書く
- テストで仕様を満たしているかどうか判定できるようにできるだけ定量的に書く
  - 定量的に書けない場合もあるが、その場合もどうなっていたら仕様を満たしているとするかどうかを合意しておくことが重要

仕様をどう書くのか－まとめ

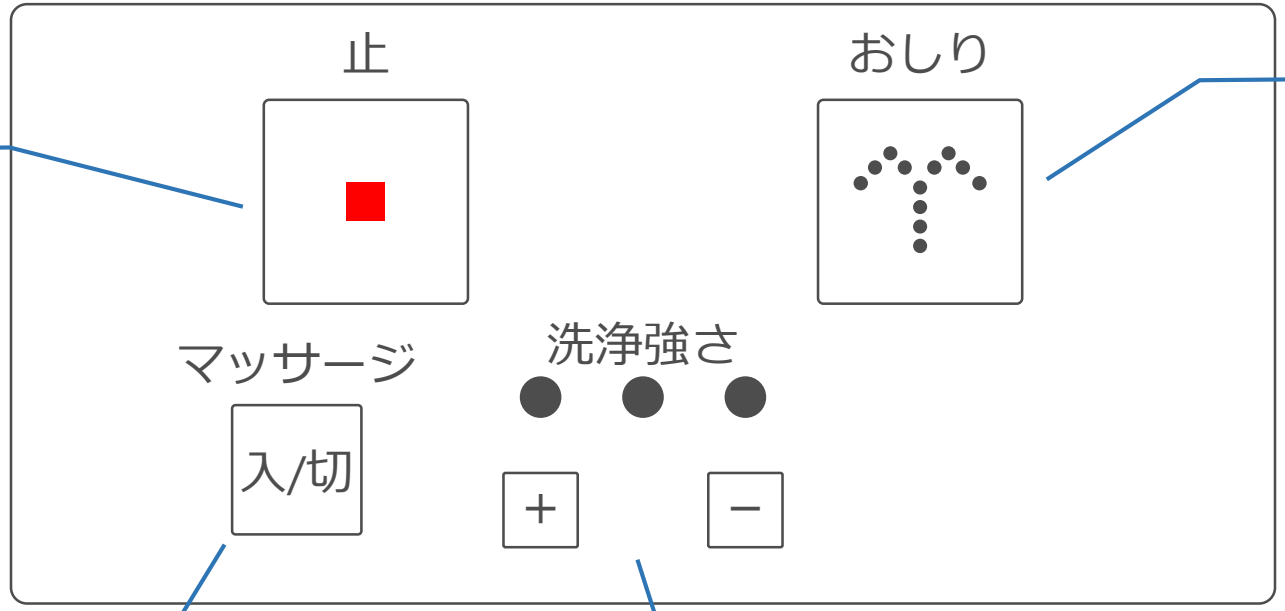
- 振る舞いは、状態遷移モデルで記述する
- 非機能要求は自然言語で書く

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

題目 ー ウォシュレットに新機能を追加する

- 基本的な機能は実現済みのウォシュレットにマッサージ機能を追加する
- 仕様書はないとする



止まるボタンを押すと
水が止まる

おしりボタンを押すと
水が出る

マッサージボタンを押すと
一定時間ごとに水圧の強弱
が切り替わる **NEW!**

+/−ボタンで水圧の強弱が切り替わる
水圧は弱・中・強

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

イベントを列挙する

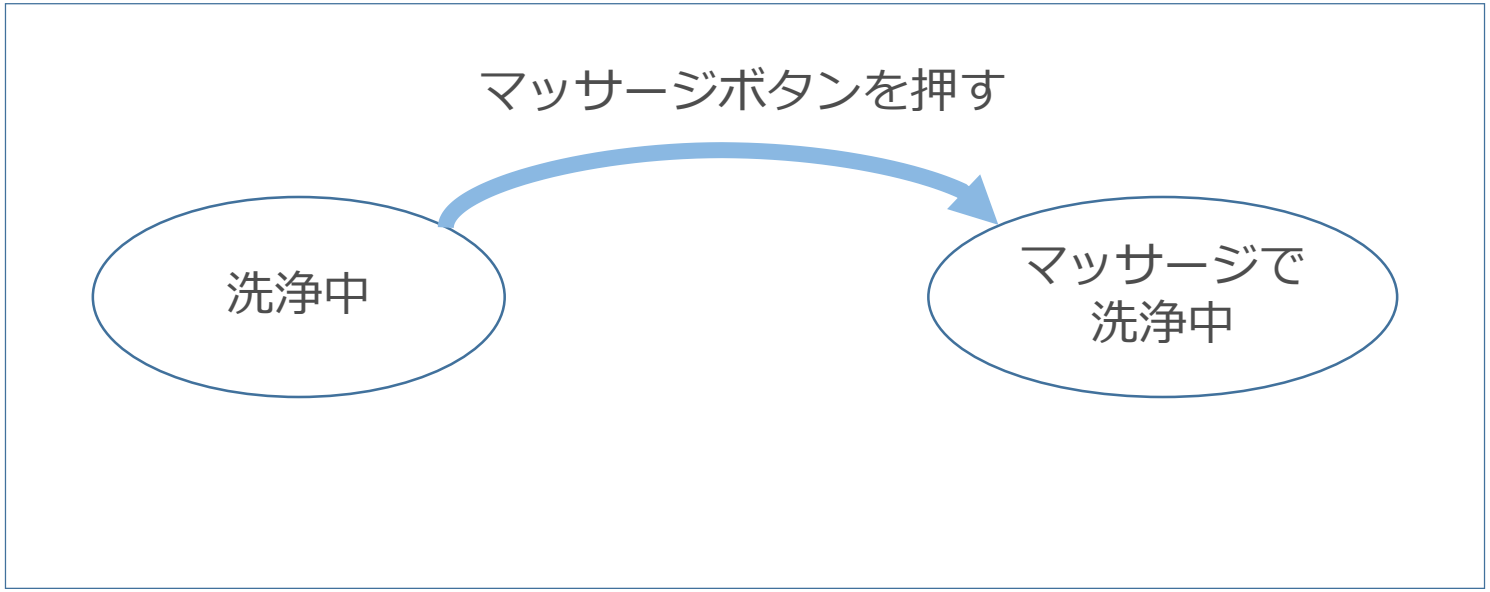
- 状態遷移モデルで記述するときに一番最初にやること
- 新機能を実現するイベントを列挙する

a. マッサージボタンを押す

挙げたイベントに対する遷移を決める

- イベントが発生するタイミングで振る舞いが異なるかもしれないが、まずは中心的な遷移から取り掛かる

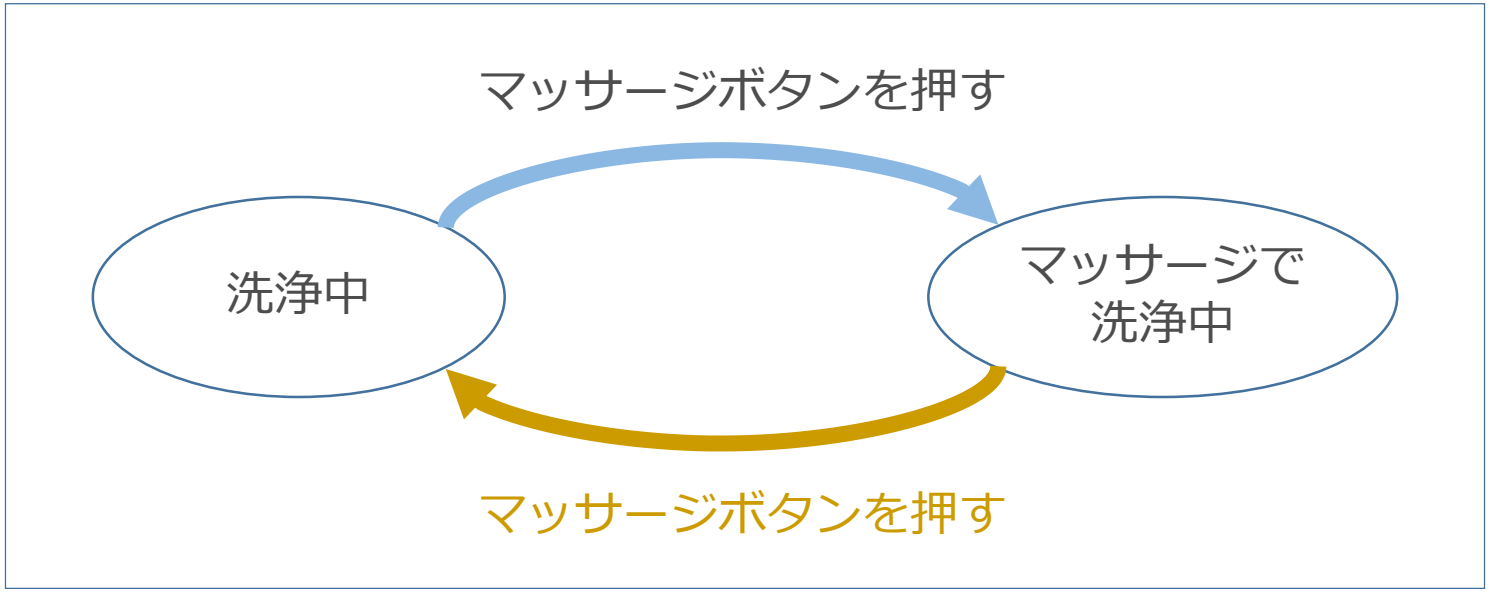
洗浄中にマッサージボタンを押したら、マッサージしながら洗浄する



すべての状態でイベントが発生したときの遷移を決める

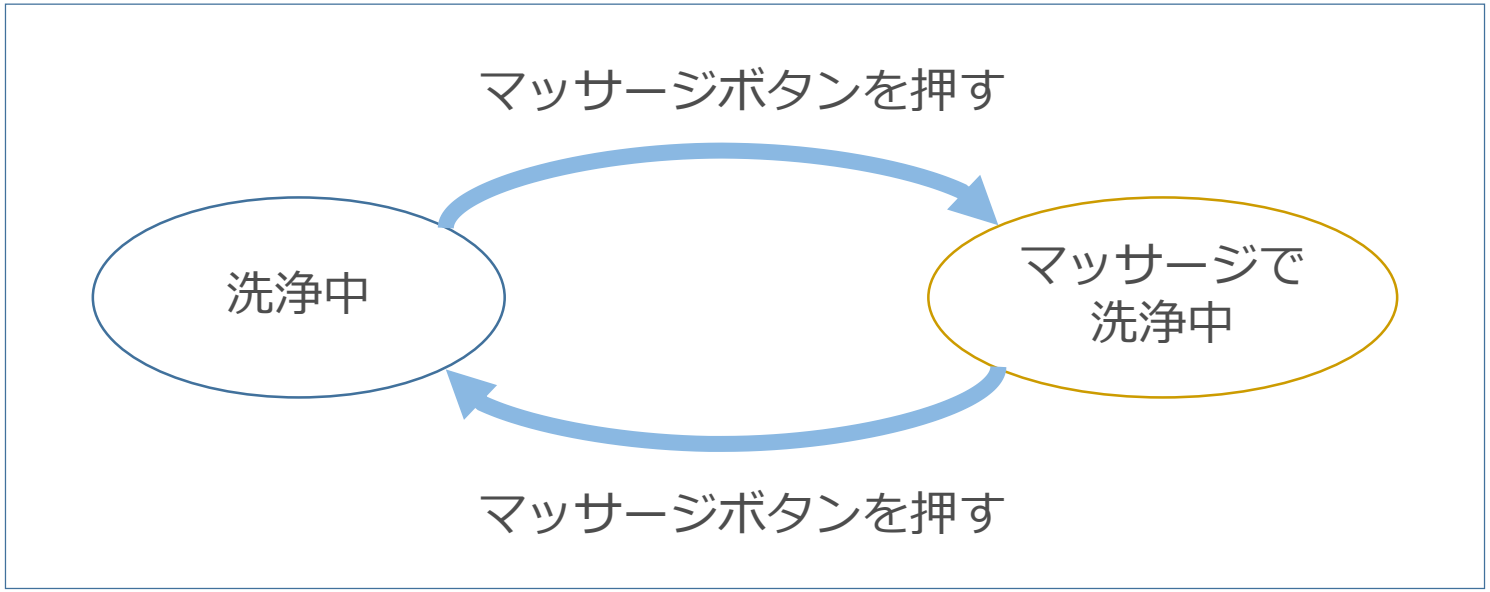
- マッサージで洗浄中にマッサージボタンを押したらどうするか？

マッサージで洗浄中にマッサージボタンを押したら、マッサージをやめて通常の洗浄中に戻る



新しい状態での既存イベントの遷移を決める

- マッサージで洗浄中状態は、既存システムにはなかった状態である
- 機能追加で増えた状態については、既存システムのイベントに対する遷移を決めて書く
- まずは既存イベントを列挙する



a. マッサージボタンを押す

b. おしりボタンを押す

c. 止めるボタンを押す

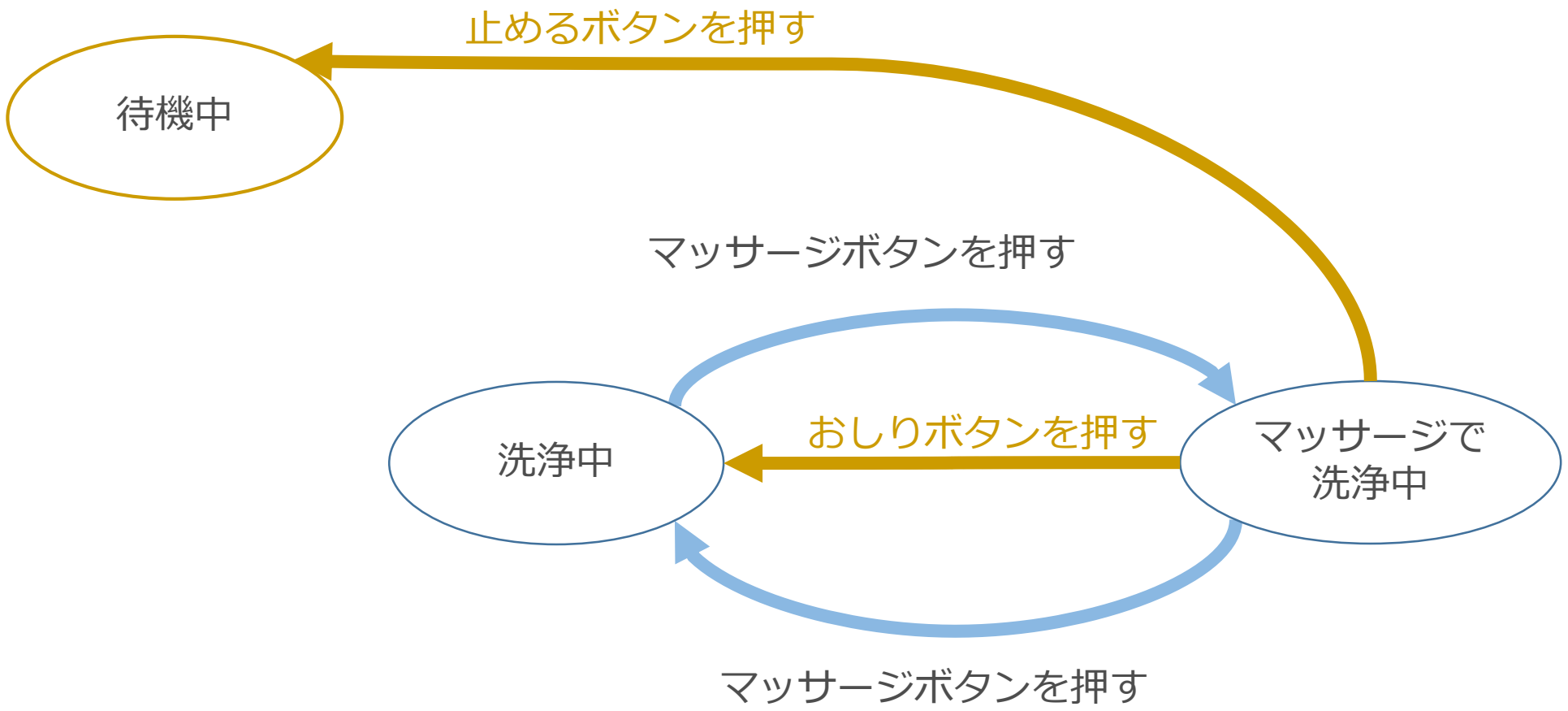
d. ＋ボタンを押す

e. －ボタンを押す

マッサージ中のおしり・止めるボタンの遷移

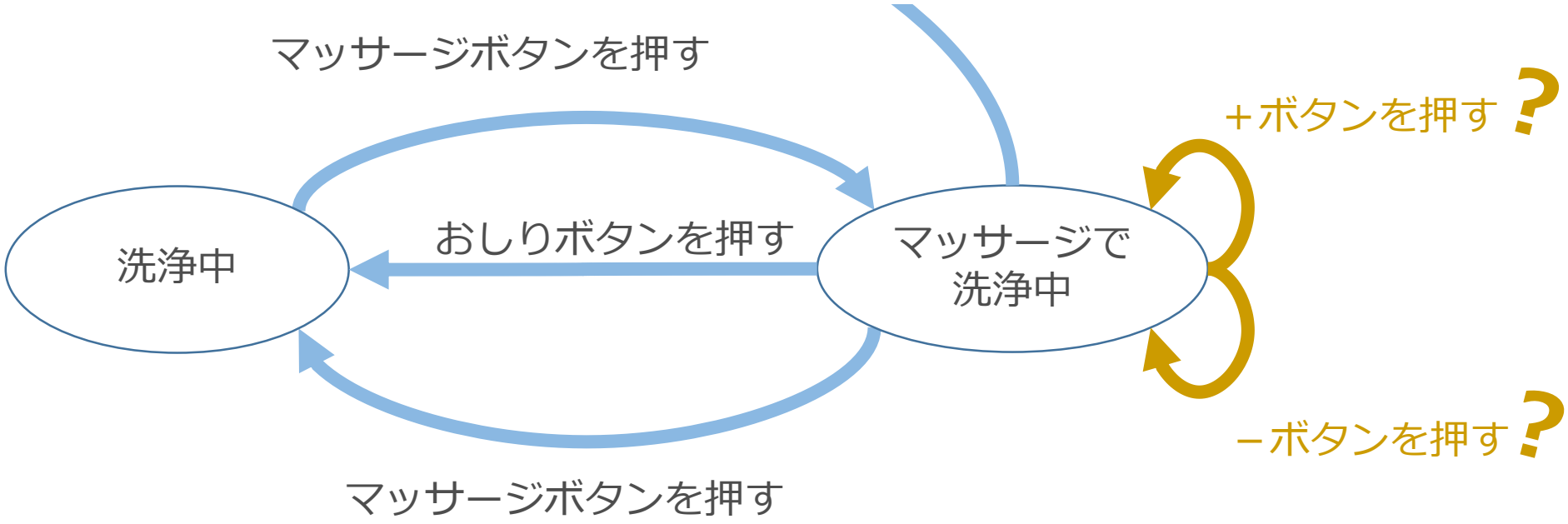
- おしりボタンを押したら通常の洗浄中に
- 止めるボタンを押したら、洗浄そのものをやめる

マッサージ中のおしり・止めるボタンの遷移



マッサージ中の+/−ボタンの遷移

- マッサージ中の+/−ボタンを押したあともマッサージで洗浄しているはずなので、マッサージで洗浄中である。ただし、水圧は変わってほしい
- しかし、水圧は変化するので前の状態とまったく変わらないでよいのか？



水圧について非形式的に整理する

- 洗浄中にマッサージボタンを押したときの水圧はどうするか？

  - 水圧が**弱**の状態で、マッサージボタンを押したときは、**弱**→**中**→**弱**→**中**を繰り返す
  - 水圧が**中**の状態で、マッサージボタンを押したときは、**中**→**強**→**中**→**強**を繰り返す
  - 水圧が**強**の状態で、マッサージボタンを押したときは、**中**→**強**→**中**→**強**を繰り返す

水圧について非形式的に整理する

- マッサージ中に＋ボタンを押したときは以下のようにしたい
  - **弱→中→弱→中**を繰り返している状態で、＋ボタンを押したときは、**中→強→中→強**を繰り返す
  - **中→強→中→強**を繰り返している状態で、＋ボタンを押したときは、**中→強→中→強**を繰り返す（変わらない）
- マッサージ中に－ボタンを押したときは以下のようにしたい
  - **弱→中→弱→中**を繰り返している状態で、－ボタンを押したときは、**弱→中→弱→中**を繰り返す（変わらない）
  - **中→強→中→強**を繰り返している状態で、－ボタンを押したときは、**弱→中→弱→中**を繰り返す

既存状態での新しいイベントの遷移を書く

- 止めるボタンの遷移で新たに追加した待機中は、既存システムで実現済みの状態である
- 既存の状態については、まず新しく追加したイベントの遷移を書く

新機能の記述の仕方

1. イベントを列挙する
2. 挙げたイベントに対する遷移を書く
3. すべての状態でのイベントの遷移を書く
4. 新しい状態での既存イベントの遷移を書く
5. 既存状態での新しいイベントの遷移を書く

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

既存システムの振る舞いは書くべきか？

書いたほうが良い。ただし最初は最低限でよい

- 書くほど、状態遷移モデルを使ってできることが増える
- 書くには工数がかかる
- 最初は最低限にとどめて、徐々に追記していくのが良い

既存システムを書く基準

0. まったく書かない
1. デッドロックや到達不可能状態がない
2. 状態間の遷移を網羅する
3. 全部書く

基準0. まったく書かない

- 到達不可能状態などが残るため、新しい機能の呼び出し方が不明瞭
- できれば避けたいレベル

基準1. デッドロックや到達不可能状態がない

- すべての状態をめぐることができるようになる
- 本来は存在する状態間の遷移が記述されないため、状態遷移をつかったシナリオ検査がかなり限定される
- **まずはこの基準を目指す**

基準2. 状態間の遷移を網羅する

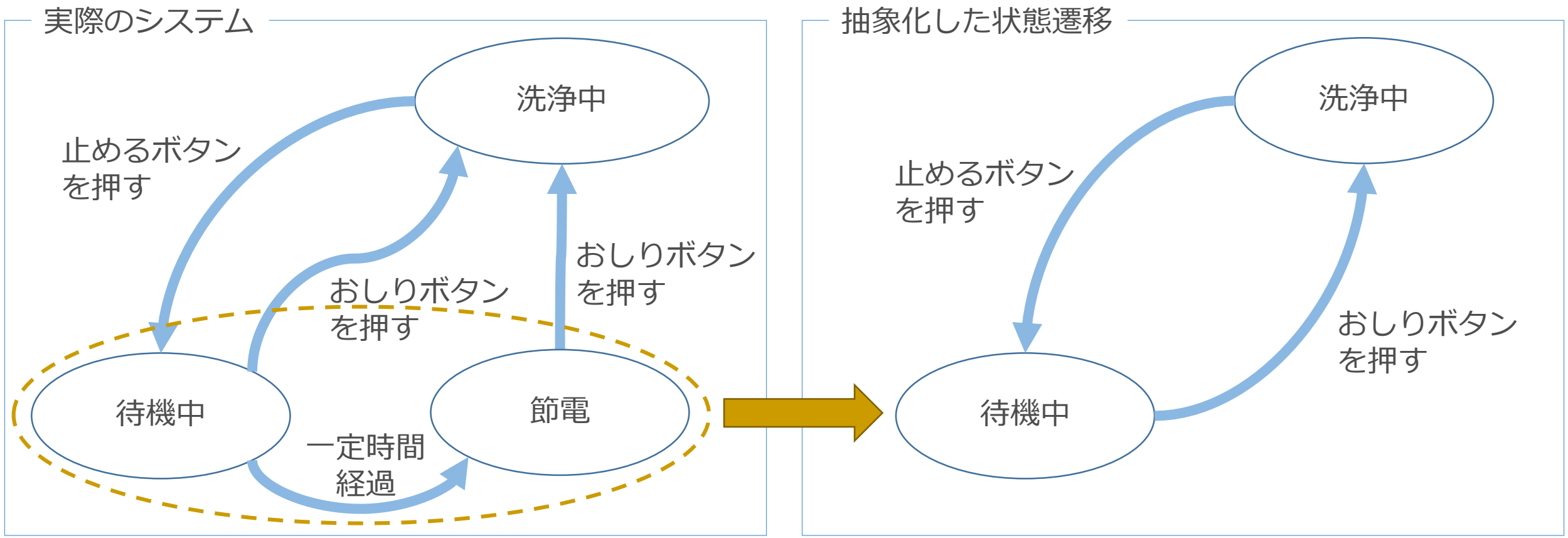
- 既存システムについては抽象的な状態遷移のみを記述して、新機能については十分な詳細度で記述したモデル
- 丸の状態で表現されない状態の変化が記述されないため、アクションに表れる振る舞いに関するシナリオ検査が十分にできない
- **できればこの基準までもっていきたい**

基準3. 全部書く

- アクション部分も含めて、既存システムについては適切な抽象度の記述で、新機能については十分な詳細度で記述したモデル
- 既存部分を含めたシナリオ検査が可能になる
- 最初から目指すと時間がかかる場合がある。徐々にこの基準を満たすようにするのがよい

既存システムの振る舞いは抽象化して書く

- 複数の状態、複数の遷移からなっていてもひとつの状態、遷移として表す
  - まずは何らかの手段で状態が遷移できることが表せれば十分とする



既存部分の記述の仕方

- 状態間の遷移を網羅することを目指す
- 抽象化して書く。新機能に注目するのに十分な詳細度でよい
- 最初は最低限でよい。徐々に書き足す
  - 仕様書を見て“既存システムの遷移が書いてない” と言う人が出てきたら、それは仕様書が活用され出したよい兆候

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

非機能要求の記述

- 非機能要求は自然言語で記述する

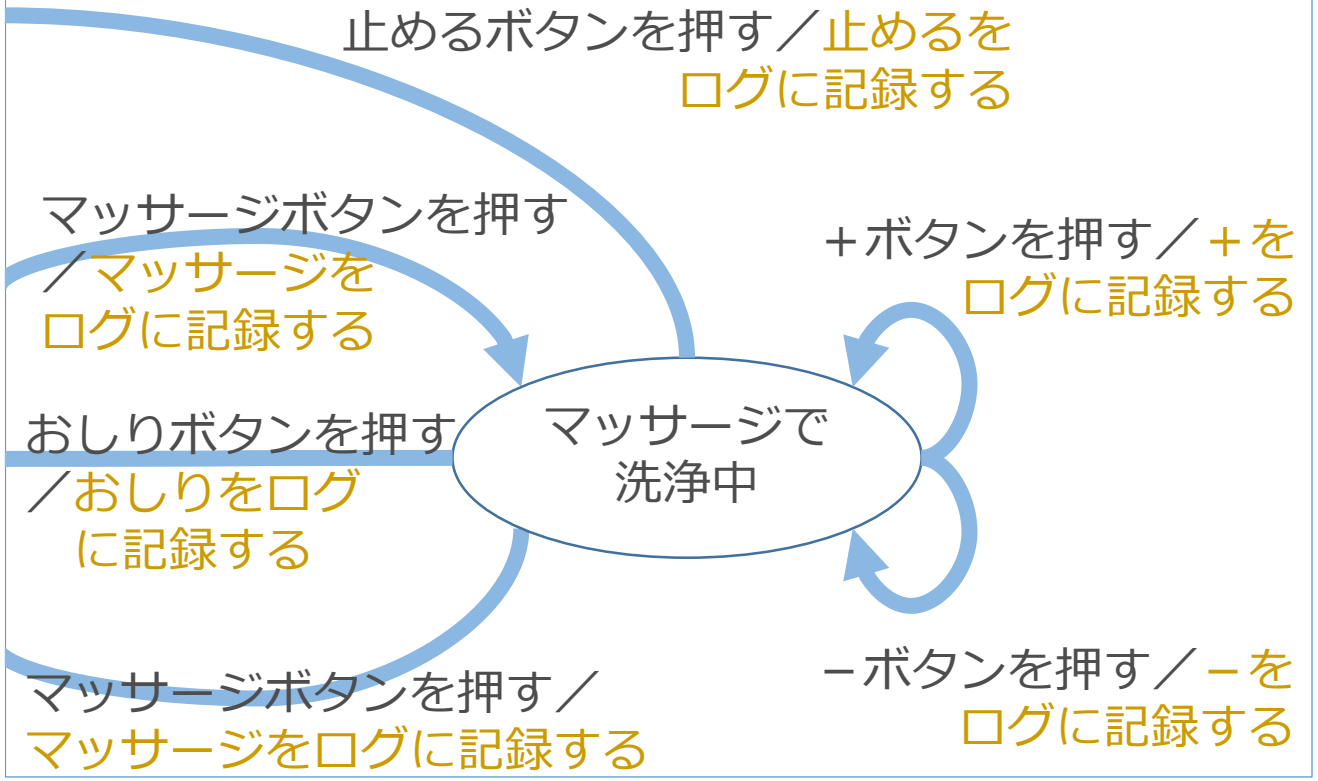
止めるボタンを押してから、1秒以内に水を止めること

- （単純な）状態遷移モデルは遷移が完了するまでの時間を表すことができない

横断的機能の記述

- 横断的機能は、状態遷移モデルで記述可能。同じ記述が頻出するのを避けるために自然言語で書いてもよい

アクションで記述する場合



自然言語で記述する場合

- すべての状態で起きたイベントをログに記録する

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

仕様の利用方法

- 仕様を入力とした成果物の作成
  - 仕様をもとに**プログラム**を設計・実装する
  - 仕様をもとに**テスト**を設計する
  - 仕様から**ユーザマニュアル**をつくる
- 仕様自身の評価
  - 状態の到達性検査
  - デッドロック検査
  - 網羅性検査
  - シナリオ検査

状態の到達性検査

- 初期状態から到達できない状態がない
- 既存部分の記述が基準１以上で使用可能

デッドロック検査

- その状態から出る遷移をひとつも持たない状態は**デッドロック**状態
- デッドロック状態がないことを検査する
- 既存部分の記述が基準１以上で使用可能

網羅性検査

- すべての状態ですべてのイベントの遷移を記述しているか？
- 抜け漏れチェックのひとつ
- 既存部分の記述が基準３で使用可能。ただし新たに追加した状態に限定すれば、基準０でも可能

シナリオ検査

- 利用方法を思い浮かべて、シナリオにする
- 遷移をたどりながらふたつのことが確認できる
  - 自分が表したかったものを表しているか？（記述に誤りがないか？）
  - 自分たちが欲しいシステムであるか？（妥当性検査）
- 既存部分の記述が基準２以上で使用可能。基準が高いほど検査できるシナリオが多い

ケーススタディ前半まとめ

- 新機能の遷移は網羅的に記述する
- 既存部分の遷移は抽象的に。最初は部分的な記述で構わない
- 仕様を書いたら、仕様自身を評価して記述の誤りと妥当性を確認する

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

形式的仕様とは

形式的仕様とは、数学的な記述で書かれた仕様のこと

プリミティブな状態遷移モデル ― 遷移

定義

- プリミティブな状態遷移モデルは、**遷移の集合**
- 遷移とは次の3つ組

  (遷移元の状態，イベント，遷移先の状態)

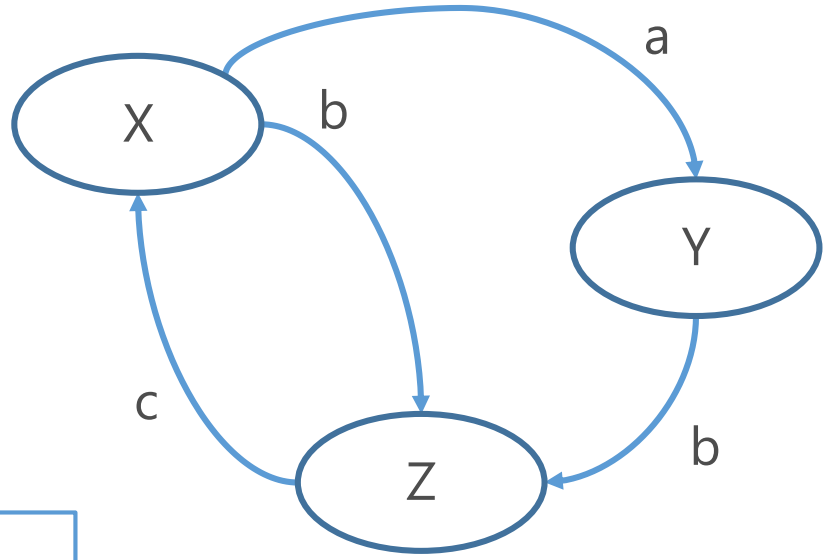
例

- 状態 X でイベント a をうけると状態 Y に遷移する遷移

  (X，a，Y)

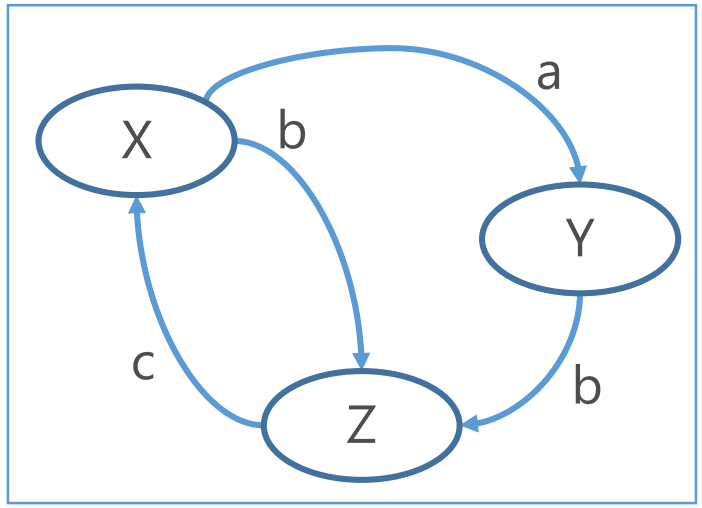
- 状態遷移

  {(X，a，Y)，(X，b，Z)，(Y，b，Z)，(Z，c，X)}



例の状態遷移図

プリミティブな状態遷移モデル ー 図と数学



状態遷移図

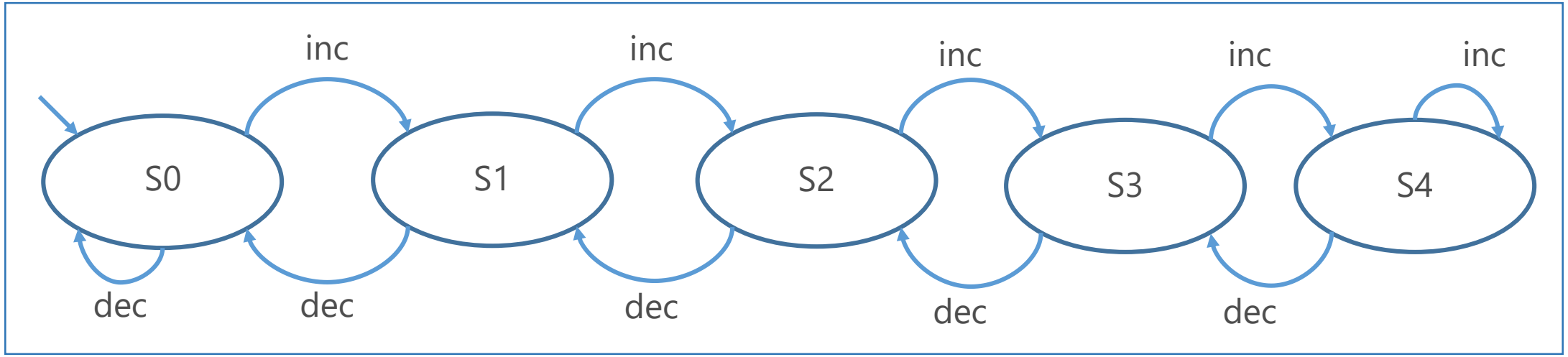
遷移を一覧したり、手で追ってシミュレーションしたりするのに便利

```
{(X, a, Y), (X, b, Z),
 (Y, b, Z), (Z, c, X)}
```

数学的記法

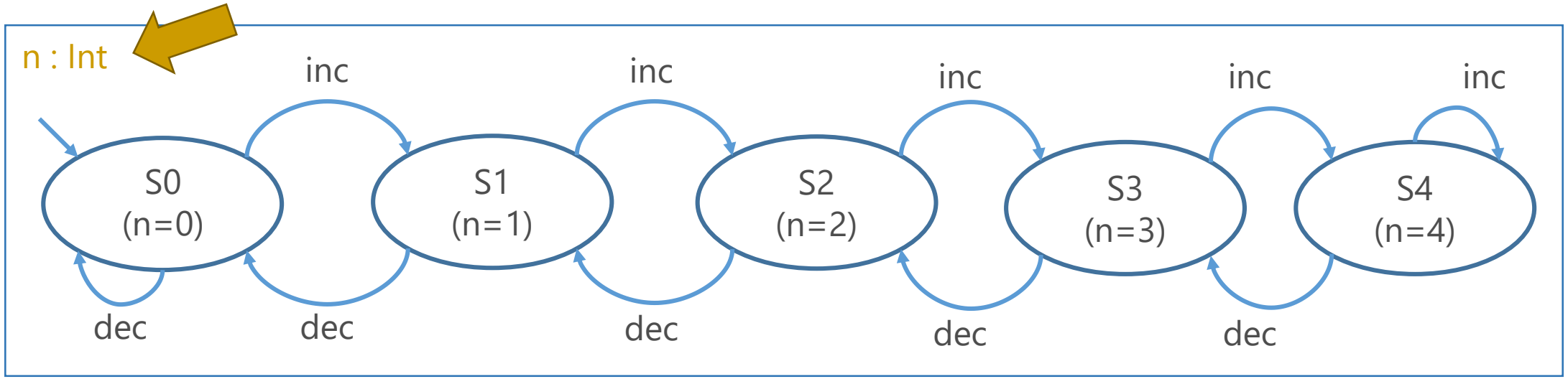
厳密な検証や分析するのに便利

状態遷移モデルに変数を導入する ― 変数なし



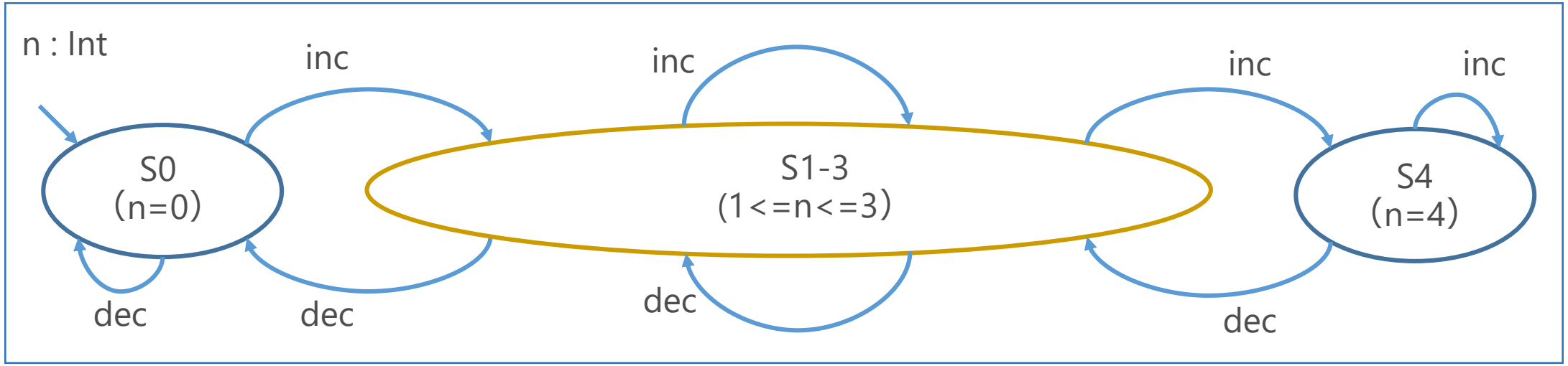
- 初期値が 0 でイベント inc で 1 つインクリメント、dec でデクリメントする状態遷移図（ただし、数値は 0 以上、5 未満）
- 状態（丸）が 5 個で、遷移が 10 個

状態遷移モデルに変数を導入する ― 変数追加



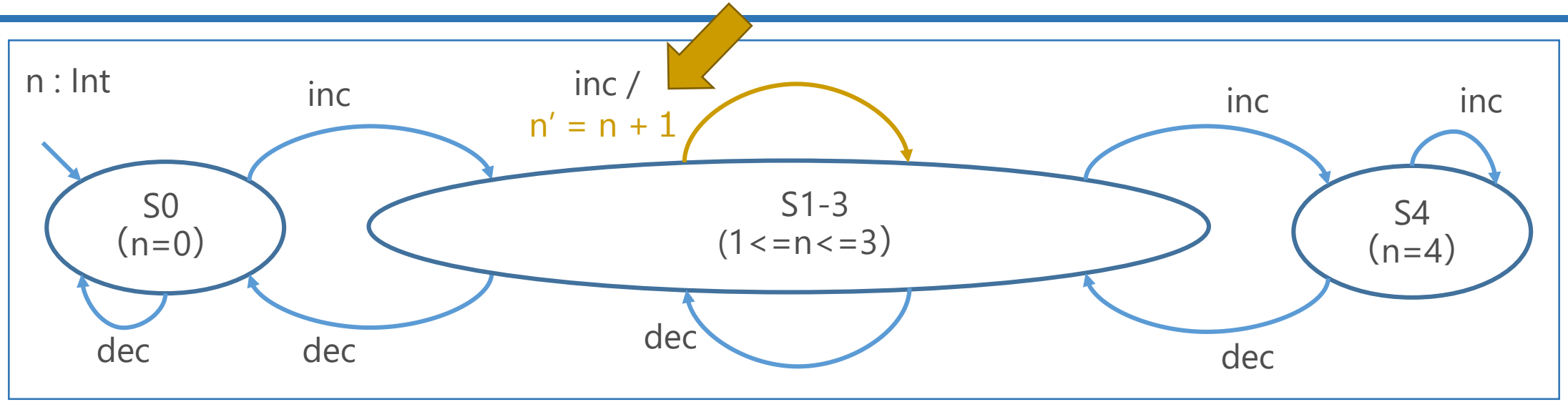
- 0 から 4 ではなく、0 から 99 だったら？数値が取りうる範囲が大きくなると図では表しづらい
- 整数型の変数 n を導入する

状態遷移モデルに変数を導入する ― グルーピング



- S1 から S3 の状態をグルーピングしてひとつにまとめる
- ただし、こうすると n = 1 のときにイベント inc をうけたときに n が 2 になることを表せない（1 <= n <= 3 であることはわかるが）

状態遷移モデルに変数を導入する ― 事後条件#1

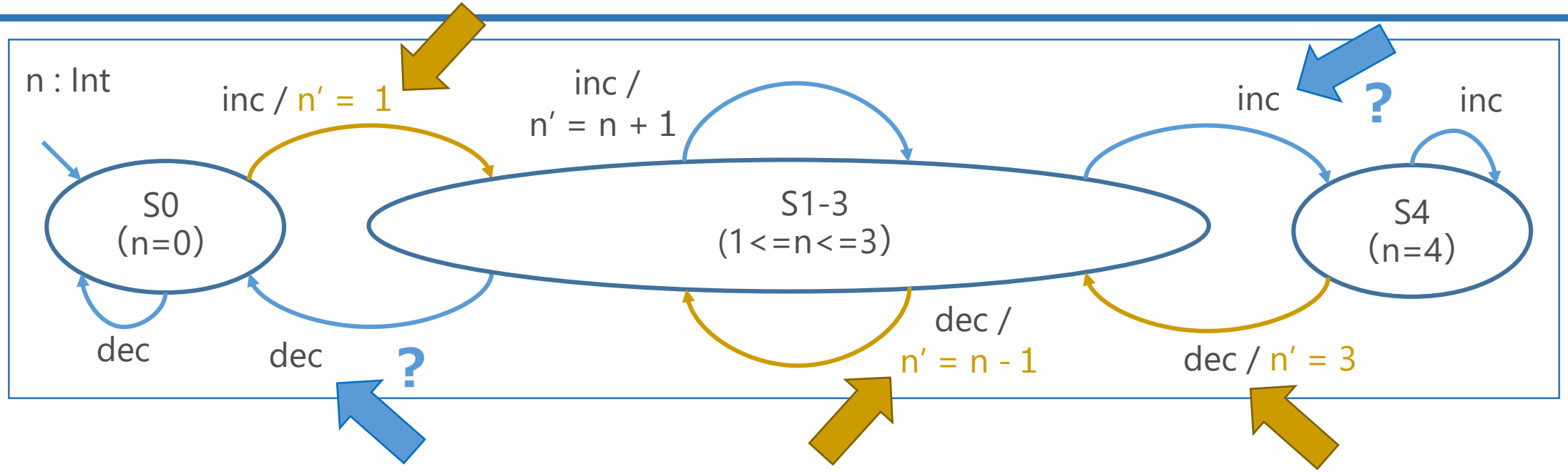


- n = 1 のときにイベント inc をうけたときの n の変化を条件で表す
- 遷移後の n は遷移前の n よりひとつ大きいものに等しい

n′ = n + 1	（ n′ は遷移後の n, = は代入ではなく等しいことを表す等号）

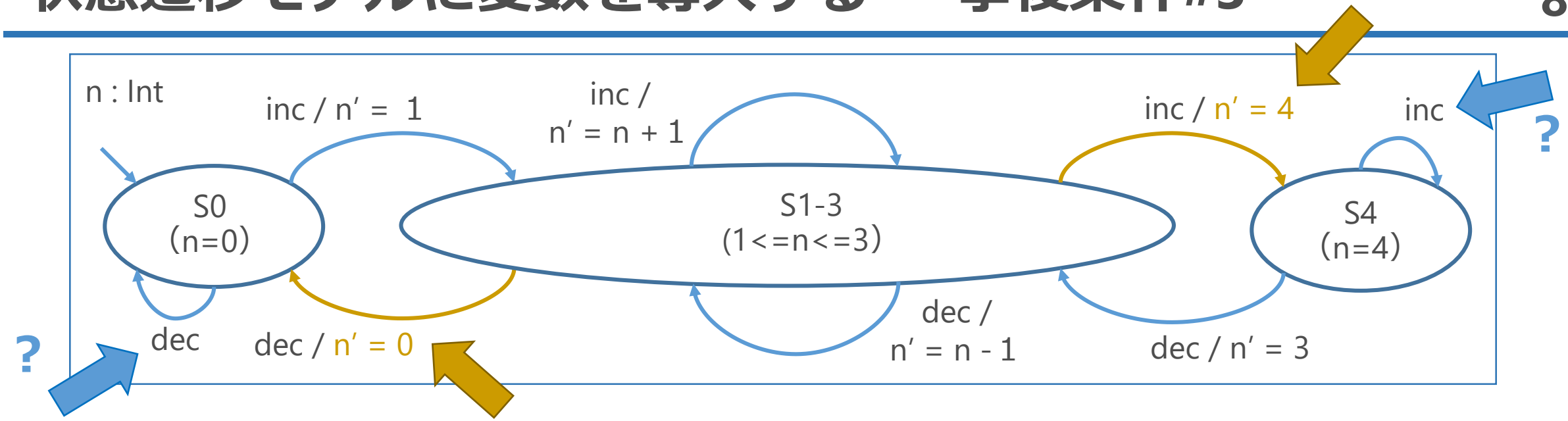
- これを**事後条件**と呼ぶ。事後条件は事前状態と事後状態の関係を表す

状態遷移モデルに変数を導入する ー 事後条件#2



- S1-3 では n は複数の値を取りうるので事後条件で表現
- S1-3 での dec, S1-3 での inc はどうか？

状態遷移モデルに変数を導入する ― 事後条件#3

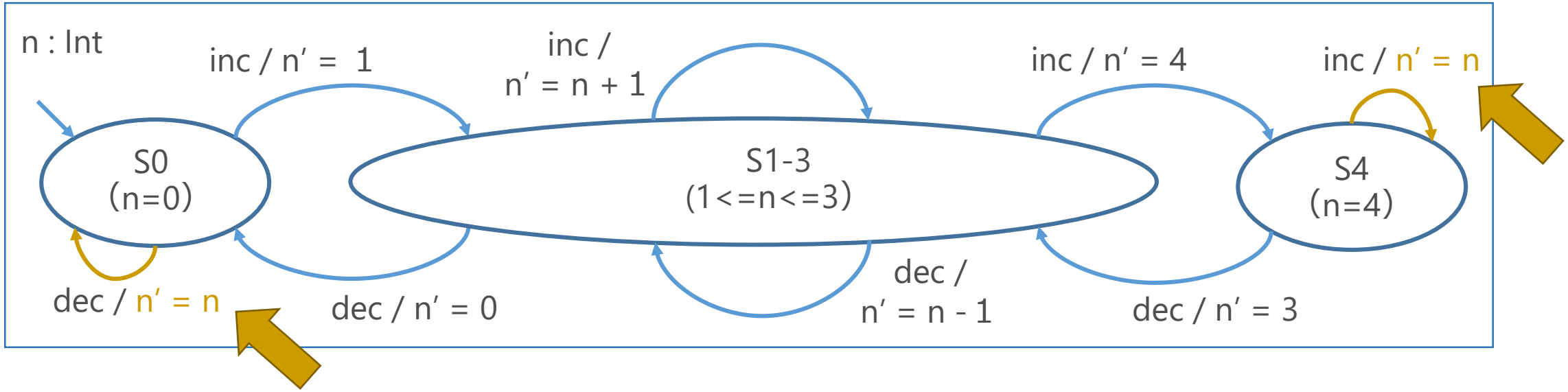


- S0, S4 ではそれぞれ n = 0, n = 4 であることが期待される

  それぞれの遷移で n がどのような変化（どのような値）をとっても良いわけではないので、事後条件を追記する

- S0 での dec, S4 での inc はどうか？

状態遷移モデルに変数を導入する ― 事後条件#4

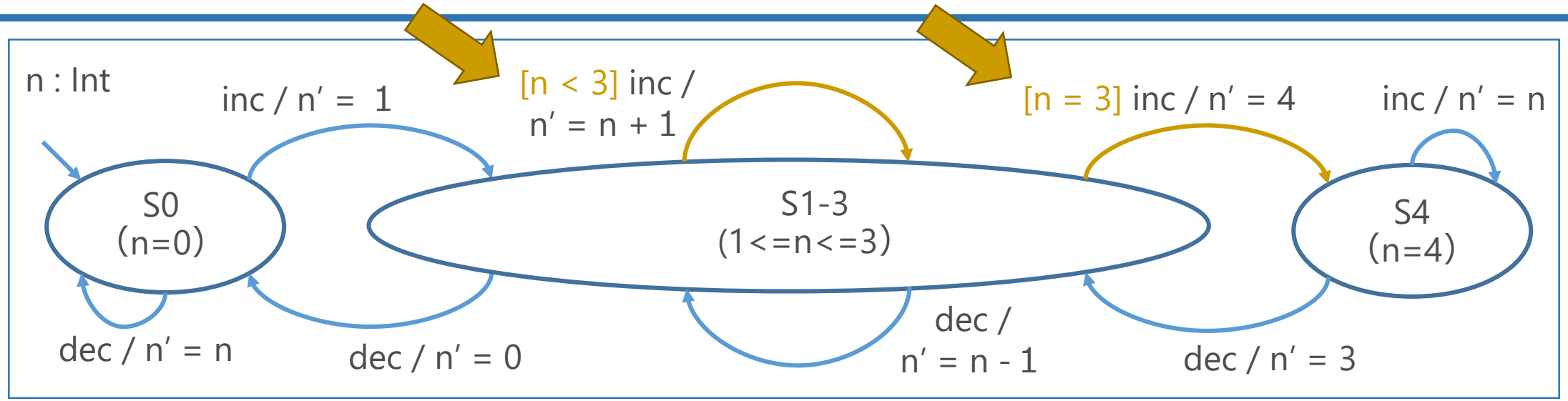


- S0, S4 ではそれぞれ n = 0, n = 4 であることが期待される

  それぞれの遷移で n がどのような変化（どのような値）をとっても良いわけではないので、事後条件を追記する

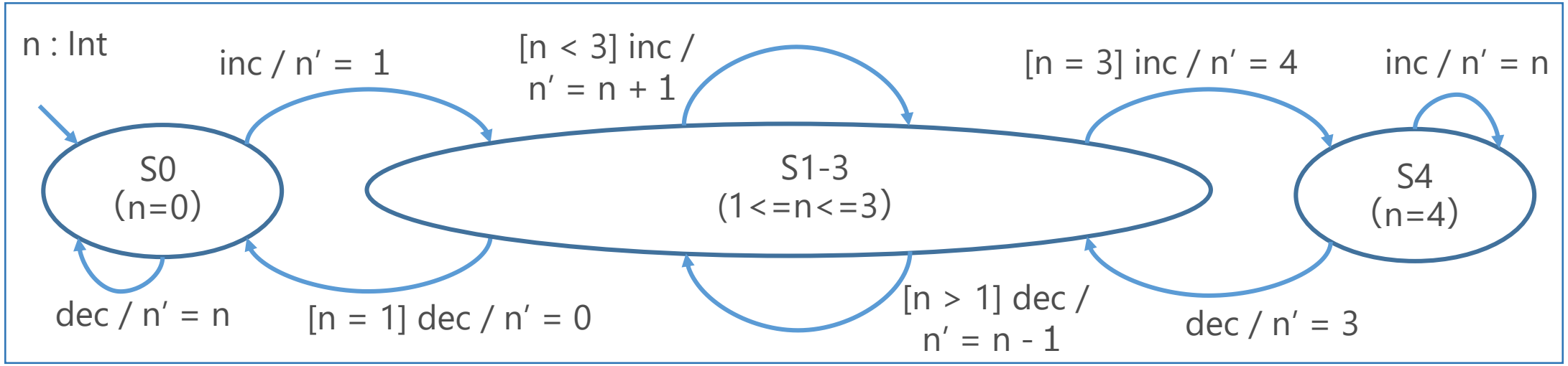
- **変わらないことも事後条件で明記する**

状態遷移モデルに変数を導入する ― ガード条件



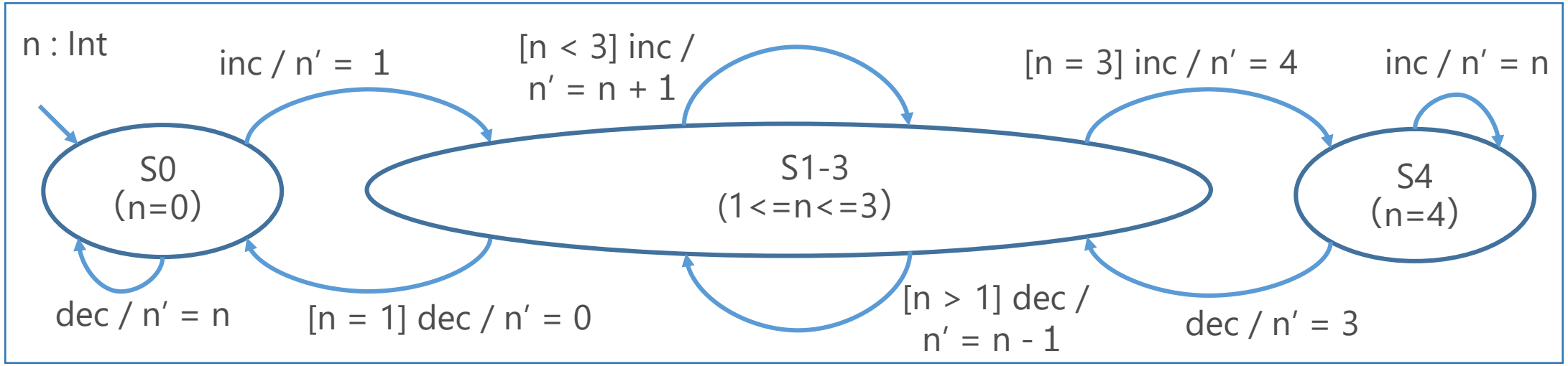
- また、n < 3 のときと n = 3 のときで、同じイベント inc の遷移後の状態が異なる
- これを表現するのに遷移に条件を付与する。イベントが発生したとき、条件が成り立っている遷移を遷移可能とする
- この条件を**ガード条件**と呼ぶ

状態遷移モデルに変数を導入する ― 完成



- 変数を導入すると、丸を少なくして状態遷移図を書くことができる
- 同じようにしてイベントもパラメータ抽象してまとめて扱うことができる
- イベントのパラメータはガード条件や事後条件にも表れうる

状態遷移モデルの形式的記述



- 状態遷移モデルの事後条件、ガード条件を数式（論理式）で表した記述
- システムの振る舞いを曖昧なく表現できる

形式的な仕様記述のまとめ

- 形式仕様記述とは、数学的に記述された仕様
- 形式的な状態遷移モデルとは、数学的な構造をもった状態変数を導入し、ガード条件、事後条件を論理式で表したもの
- 厳密な推論や検証が可能になる

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

アクションから状態変数を見つける

- アクションの記述で変化する対象を状態変数とする

アクションから事後条件へ

- 事後条件に状態変数の変化を論理式で書く
- アクションでは手続き的な表現になりがちだが、条件式にすることで適切な抽象度で記述ができる

形式的仕様の検査方法

- 非形式的な仕様で行える検査はすべて可能
- 厳密に正確に検査できる
- ツールを使うことができる。自動化も可能

記述の抜け漏れ検査

- すべての遷移で状態変数の事後条件が定められているか？
- **状態変数**と**事後条件**を記述した状態遷移モデルで使用可能

網羅性検査

- すべての状態ですべてのイベントに対する遷移が決められているか？
- ある状態のひとつのイベントに対する遷移が全域的か？
  - ガード条件の論理和をとって真になれば全域的
- ガード条件が論理式で表されていると可能ある状態のひとつのイベントに対する遷移が全域的か？

シナリオ検査

- 非形式的な仕様と同様の手段が使える
- 状態変数の変化を観測することで仕様の妥当性や間違いに気づくことができる

非形式的な状態遷移の形式化の仕方

1. アクション記述から状態変数を見つける
2. アクションを事後条件に変換する
3. ガード条件を論理式で表す
4. 書いた仕様を検査する。ツールも使用可能

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

仕様に関するQ&A

- 仕様とは？

  ステークホルダーが観測できるシステムの特性を記述したもの

- ソフトウェアシステムの仕様には何を書くのか？

  振る舞いと非機能要求

- 振る舞いと非機能要求はそれぞれどう書くのか？

  振る舞いは状態遷移モデルで、非機能要求は自然言語で書く

- 仕様の利用方法は？

  プログラムやテストの設計の入力とするほかに、早い段階でシステムを評価できる

仕様に関するQ&A

- どの程度の詳細度で書けばよいか？
  - ステークホルダーが観測できる振る舞いを書く
  - 状態遷移モデルにすることで振る舞いを適切な抽象度に保つことができる。アクションより事後条件の方が抽象的に記述できる
  - 既存システムなどはまずは抽象度をあげて書く
- 抜け漏れがない仕様を書くには？

  状態変数を導入して事後条件を書く。遷移の事後条件を調べれば抜け漏れを見つけられる

仕様に関するQ&A

- 仕様を導入するには？

  まずは最低限の記述から始める。徐々に書き足せば良い

- だれが仕様を書くべきか？

  - システムに関する知識と仕様記述のスキルの両方を持つ人がふさわしい。多くの場合エンジニアがそれにあたる
  - あなたが仕様を欲しいと思っているのであれば、あなたが書くのが良い

- 形式化のメリットは？

  非曖昧な仕様が手に入る。誰が何度読んでも同じ意味で読み取ることができる

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	

発表の構成

仕様とは	仕様とは何か？	どう書くのか？
ケーススタディ前半	（題目）ウォシュレットに機能追加	
	新機能の記述	既存部分の記述
	非機能要求の記述	仕様の利用方法
ケーススタディ後半	形式的な仕様記述とは	
	非形式から形式仕様記述へ	
まとめ	仕様に関するQ&A	