経営情報システム学特論１
１０．時間の離散モデル化

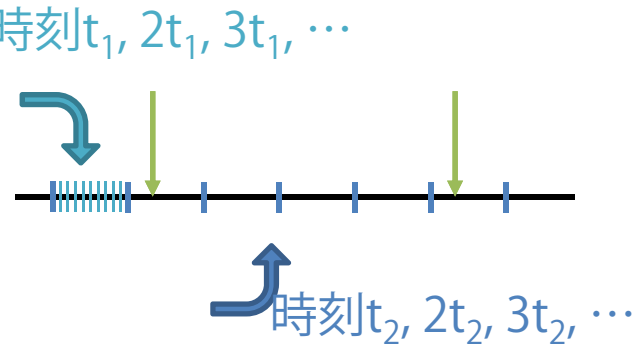
SS専攻 経営情報システム学講座 客員
石川 冬樹
f-ishikawa@nii.ac.jp

目次

- 時間のモデル化
- 時間を考慮した検証
- モデルの表すプログラム・意味
- 例題：荷物の仕分け

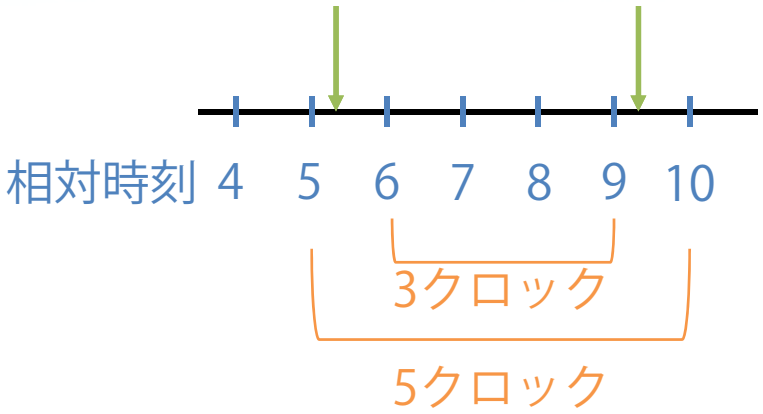
時間のモデル化

- 「最初のマウスクリック後，2秒以内に次のクリックが起きるとダブルクリックとして扱う」
  - 物理世界での絶対時間（連続値）
    - 「最初のマウスクリック後，1.825041521…（秒後に次のクリックが起きた」
  - 実際の観測機構あるいは分析モデル
    - クロックにより計測
    - クロックの精度はさておき，結局離散値
    - 実際のクロック間隔は十分小さい，分析モデルでは荒く考えるかも



時間のモデル化

- 「2秒以内だとダブルクリック」
  - 1クロック＝0.5秒だとすると次の図の場合は？



「時刻5」と「時刻9」のクリック：
時間差0.5 × 4 = 2秒?

実際に断言できることは
1.5秒 < クリック間の間隔 < 2.5秒
（クロックの誤差を除く）

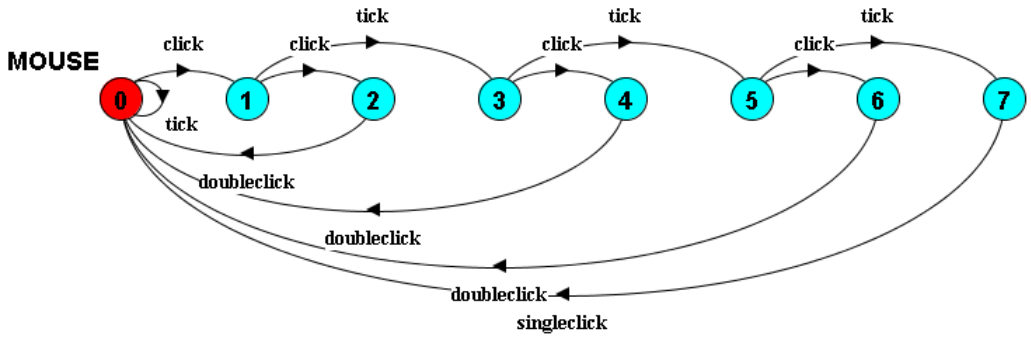
  - 真に（絶対時間で）2秒以内だったかはわからない
  - 現実的には「相対時刻の差が4以下ならダブルクロック（最悪2.5秒離れてるかも）」でも気にならない

今回の例のLTSA版

Dクロック未満ならダブルクロックと見なす

```
MOUSE(D=3) = (
  tick -> MOUSE
  | click -> COUNT[1]
),
COUNT[t:1..D] = (
  when (t==D) tick -> singleclick -> MOUSE
  | when (t<D) tick -> COUNT[t+1]
  | click -> doubleclick -> MOUSE
).
```

tickするタイマーを合成してもよいが
今回はプロセスが1つなので
あってもなくても同じ



典型的な時間動作のモデル例

- 所要時間の見積もりとして，最小所要時間，最大所要時間が与えられたタスク

```
TASK(Min=2,Max=4) = (
  start -> COUNT[0]
 | tick -> TASK
),
COUNT[t:0..Max] = (
  when (t>=Min && t<=Max) end -> TASK
  | when (t<Max) tick -> COUNT[t+1]
).
```

典型的な時間動作のモデル例

- タイムアウト動作

```
TIMEOUT(D=3) = (
  startwait -> COUNT[0]
  | {tick, come} -> TIMEOUT
),
COUNT[t:0..D] = (
  when (t<D) tick -> COUNT[t+1]
  | when (t==D) timeout -> TIMEOUT
  | come -> TIMEOUT
).
```

目次

- 時間のモデル化
- **時間を考慮した検証**
- モデルの表すプログラム・意味
- 例題：荷物の仕分け

時間を考慮した検証

- これまでのモデルは，時間を考慮せずすべての可能性を考えていた
- 実際には起きない遷移がモデルに表現されている可能性がある（いい場合も悪い場合も）
  - 実際にありえない速さでリクエストが次々と来て詰まることがある
  - 実際にあり得ない速さでリクエストが処理されてうまくいくことがある
  - 他プロセスがどんなに進んでも，あるプロセスは一歩も進まない可能性がある（現実には考える必要がないことが多い）
  - ．．．

時間なし生産者と消費者（同期）

```
CONSUMER = (
 get -> CONSUMER
 | skip -> CONSUMER).

PRODUCER = ( put -> PRODUCER ).

||SYSTEM = ( CONSUMER || PRODUCER )
           /{item/{put,get}}.
```

消費者は時折消費

生産者はとにかく生産

失敗しようがない

時間あり生産者と消費者（同期）

```
CONSUMER(Tc=3) = (
  get -> COUNT[1]
  | tick -> CONSUMER
),
COUNT[t:1..Tc] = (
  when (t==Tc) tick -> CONSUMER
  | when (t<Tc) tick -> COUNT[t+1]
).

PRODUCER(Ts=3) = ( put -> COUNT[1] ),
COUNT[t:1..Ts] = (
  when (t==Ts) tick -> PRODUCER
  | when (t<Ts) tick -> COUNT[t+1]
).
```

消費後最低Tcクロック経過後，
次の消費が発生しうる
（すぐにしなくてもよい）

生産後Tsクロック経過したら，
次の生産が発生する
（しなければならない）

時間あり生産者と消費者（同期）

```
||SAME = ( CONSUMER(2) || PRODUCER(2) )
          /{item/{put,get}}.

||SLOWPRODUCE = ( CONSUMER(2) || PRODUCER(3) )
 /{item/{put,get}}.

||FASTPRODUCE = ( CONSUMER(3) || PRODUCER(2) )
 /{item/{put,get}}.
```

生産者が速い場合のみデッドロック（パスはitem tick tick）

時間あり生産者と消費者（バッファ）

```
STORE(N=3) = STORE[0],
STORE[i:0..N] = (
  put -> STORE[i+1]
  | when(i>0) get -> STORE[i-1]
).

||SYSTEM = ( CONSUMER(1) || PRODUCER(1)
             || STORE).
```

オーバーフローによるERRORあり

バッファ付きバージョン

速さは揃えているが，消費者は
待つかもしれないので，
オーバーフローは起きうる

時間あり生産者と消費者（バッファ）

- 状態遷移モデルは「こういう遷移を起こすことができる・起きる可能性がある」ことを書く
  - ずっと起こさない可能性も含む
- 「できるときには待たずにすぐに起こす」という意図は明示的に書く必要がある

```
||FASTSYSTEM = SYSTEM>>{tick}.

progress TIME = {tick}
```

発火できるときには待たずに発火
→オーバーフローはなくなる

時間が経過せずに無限に行動できる，
ようにはなっていないことを確認

目次

- 時間のモデル化
- 時間を考慮した検証
- モデルの表すプログラム・意味
- 例題：荷物の仕分け

モデルの表すプログラム・意味

- そもそも下記は，実際には何を意味するのか？

```
tick -> act1 -> act2 -> tick
```

  - act1とact2が終わるまで「時計が待たされる」？

モデルの表すプログラム・意味

- そもそも下記は，実際には何を意味するのか？

```
tick -> act1 -> act2 -> tick
```

  - act1とact2が終わるまで「時計が待たされる」？

可能性１．実際のプログラム・システムは，act1，act2を順々に実行するだけだが，それらは十分に早く，考えているクロックを消費しないと見なしてしまって問題ないという仮定を表現している

- どういうことが可能な世界を表現しているのか，より注意が必要

モデルの表すプログラム・意味

- そもそも下記は，実際には何を意味するのか？

```
tick -> act1 -> act2 -> tick
```

  - act1とact2が終わるまで「時計が待たされる」？

可能性２．シミュレーターやゲームの中で，指示された通りに時間経過を再現するコードは，act1とact2が終わってから（その仮想世界の中の）クロックを進めるという可能性もある

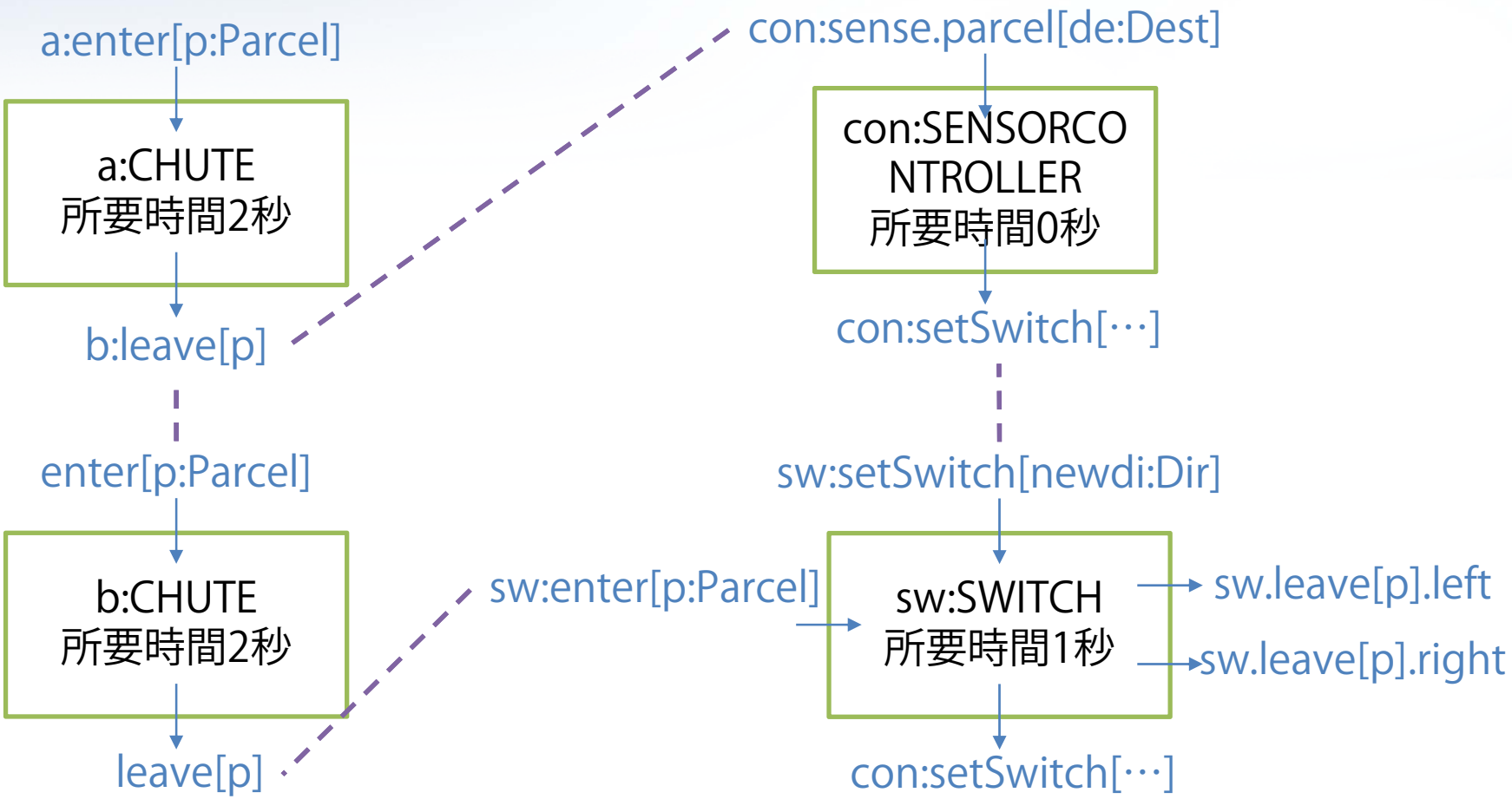
目次

- 時間のモデル化
- 時間を考慮した検証
- モデルの表すプログラム・意味
- 例題：荷物の仕分け

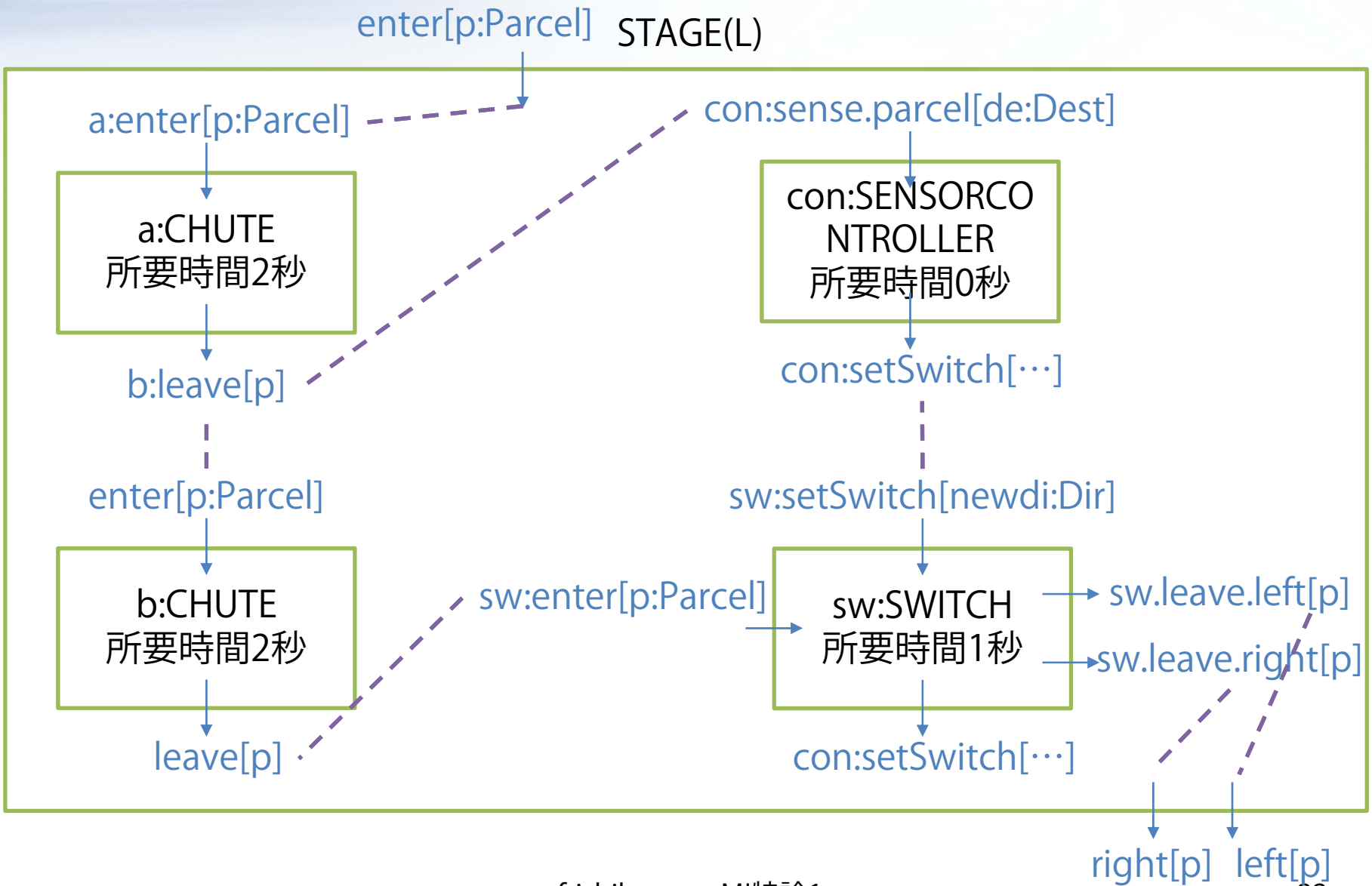
荷物の仕分け

- 滑り台を通し荷物を流していく（parcelrouter.lts）
  - 荷物は一定間隔で送られる
  - 途中にはセンサーがあり，荷物の行き先を見てその先にあるスイッチを切り替える
  - スイッチの結果に基づき，左右の分かれ道のどちらかに進んでいく
  - 荷物がスイッチを通過するまで，スイッチの向きを変えることはできない
- 正しく仕分けができるか？
  - 荷物を流す間隔を狭めたいが，短くし過ぎるとスイッチが切り替えられない

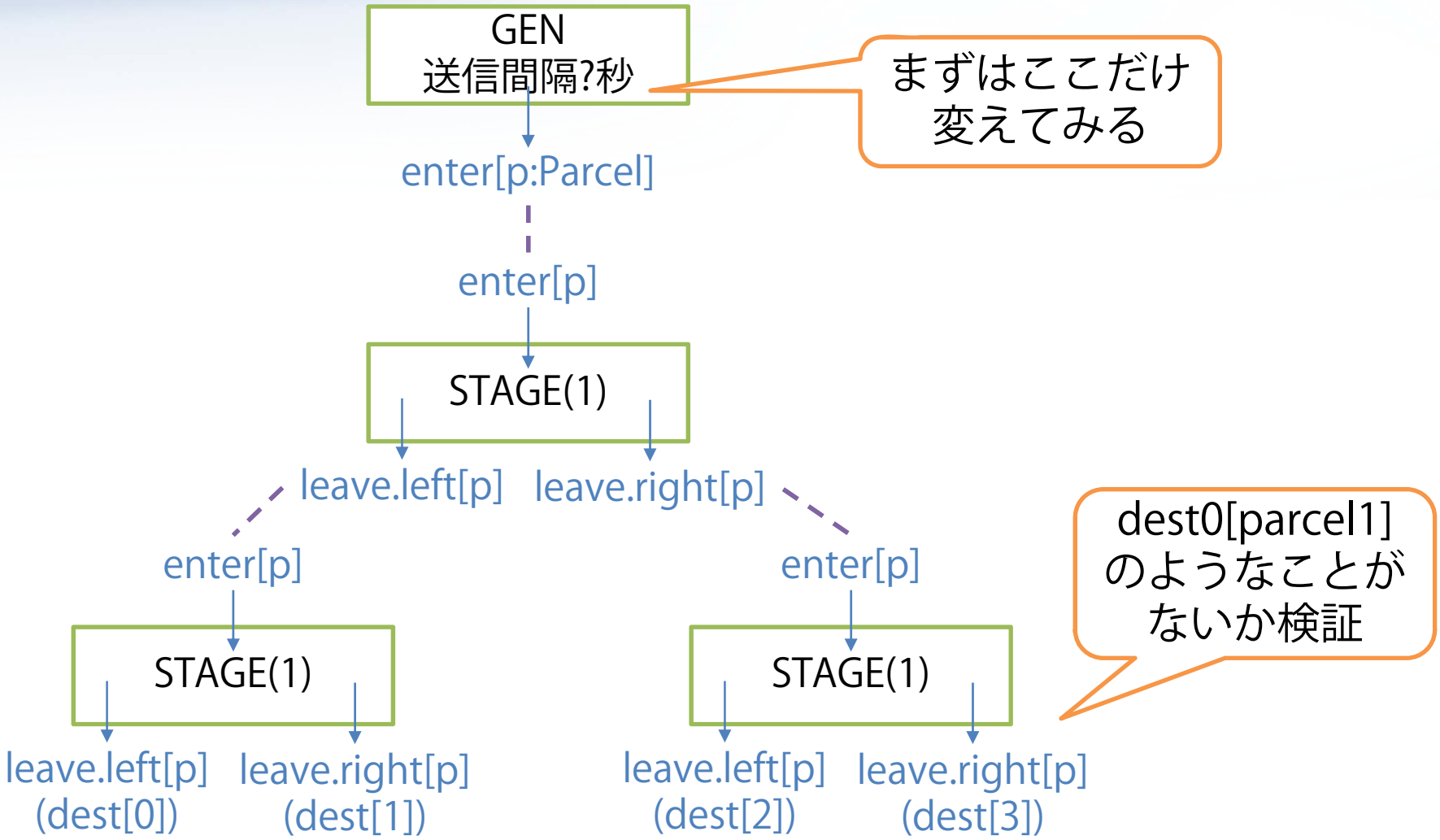
部品構造（1レベル内）


a:enter[p:Parcel]
a:CHUTE
所要時間2秒
b:leave[p]
enter[p:Parcel]
b:CHUTE
所要時間2秒
leave[p]
con:sense.parcel[de:Dest]
con:SENSORCONTROLLER
所要時間0秒
con:setSwitch[…]
sw:setSwitch[newdi:Dir]
sw:enter[p:Parcel]
sw:SWITCH
所要時間1秒
sw.leave[p].left
sw.leave[p].right
con:setSwitch[…]

部品構造（ブラックボックス化）



部品構造（全体）


GEN
送信間隔?秒
まずはここだけ
変えてみる
enter[p:Parcel]
enter[p]
STAGE(1)
leave.left[p]
leave.right[p]
enter[p]
enter[p]
STAGE(1)
STAGE(1)
dest0[parcel1]
のようなことが
ないか検証
leave.left[p]
(dest[0])
leave.right[p]
(dest[1])
leave.left[p]
(dest[2])
leave.right[p]
(dest[3])

演習：荷物の仕分け

- 配布ファイルparcelrouter.ltsにおいて，荷物を送り出す間隔など様々な所要時間を変化させてみて検証を行い結果を確認してみよ

おまけ

- このようにクロックをモデル化すると状態爆発が起きやすい
- 複数のクロックをまとめて1つの状態と見なしてくれるようなツールもある（例：UPPAAL）
  - 例：
    プロセス1： a （最短2秒後，最長4秒後）
    プロセス2： b （最短1秒後，最長3秒後）
  - 起きうるのは結局，
    - 1〜2秒にbが起き，2〜4秒にaが起きる
    - 2〜3秒にaもbも起きる（aが先か，bが先か）
    - 2〜3秒にbが起き，3〜4秒にaが起きる

まとめ

- 時間のモデル化
  - 離散モデルにより表現することになる場合，適切な粒度を定める必要があるが，離散であること自体には実用上の問題はない
  - 時間に関する仮定・制約を考慮すると，しない場合には隠れていた不整合が顕在化することが多い
  - クロックとの同期を自らモデル化する場合，何が起きることとしているのか，何が起きえないとしているのか，注意深く検討する必要がある
- 次回：Java並行プログラミングのノウハウ