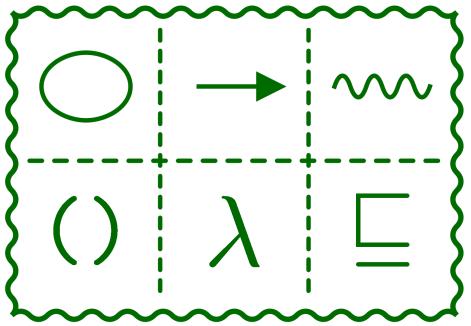
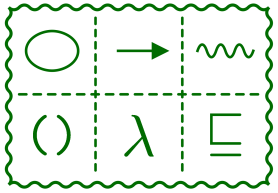


並行システムの検証と実装

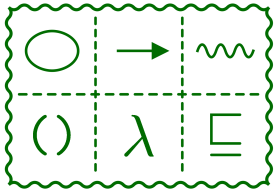
第0章 並行システム

PRINCIPIA Limited
初谷 久史



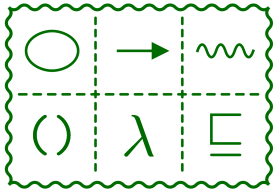
ゴール

- 要求を満たす信頼性の高い並行システムを設計できるようになること
  - 対象とする設計の作業
    - 並行システムの振る舞いを表すモデルを作成すること
    - 作成したモデルが与えられた要求や仕様を満たしていることを検証すること
    - 検証されたモデルに基づいてシステムを実装すること



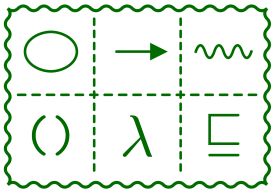
この講義で学習すること

- 並行システムについて学ぶ
- 並行システムの理論である CSP 理論を学ぶ
  - Communicating Sequential Processes
- 並行システムの振る舞いをモデル化し，検証するツールの使い方を学ぶ
  - 構造と振る舞いのモデリング
  - 検査と結果の分析
- 並行システムの実装方法について学ぶ
  - ~~プログラミング言語~~
  - OS提供のIPC，pthread ライブラリ，不可分操作



講義計画

第1, 2回	並行システム概説，状態遷移図によるプロセスのモデル化
第3回	式によるプロセスのモデル化，並行合成
第4回	逐次合成，デッドロック検査
第5, 6回	プロセスの動的生成と終了，隠蔽，非決定性，発散
第7, 8回	トレース方式による詳細化検査
第9, 10回	安定失敗方式による詳細化検査
第11, 12回	CSP 理論
第13回	CSP ライブラリによる実装
第14回	同期プリミティブによる実装
第15回	不可分操作によるロックフリーアルゴリズムの実装

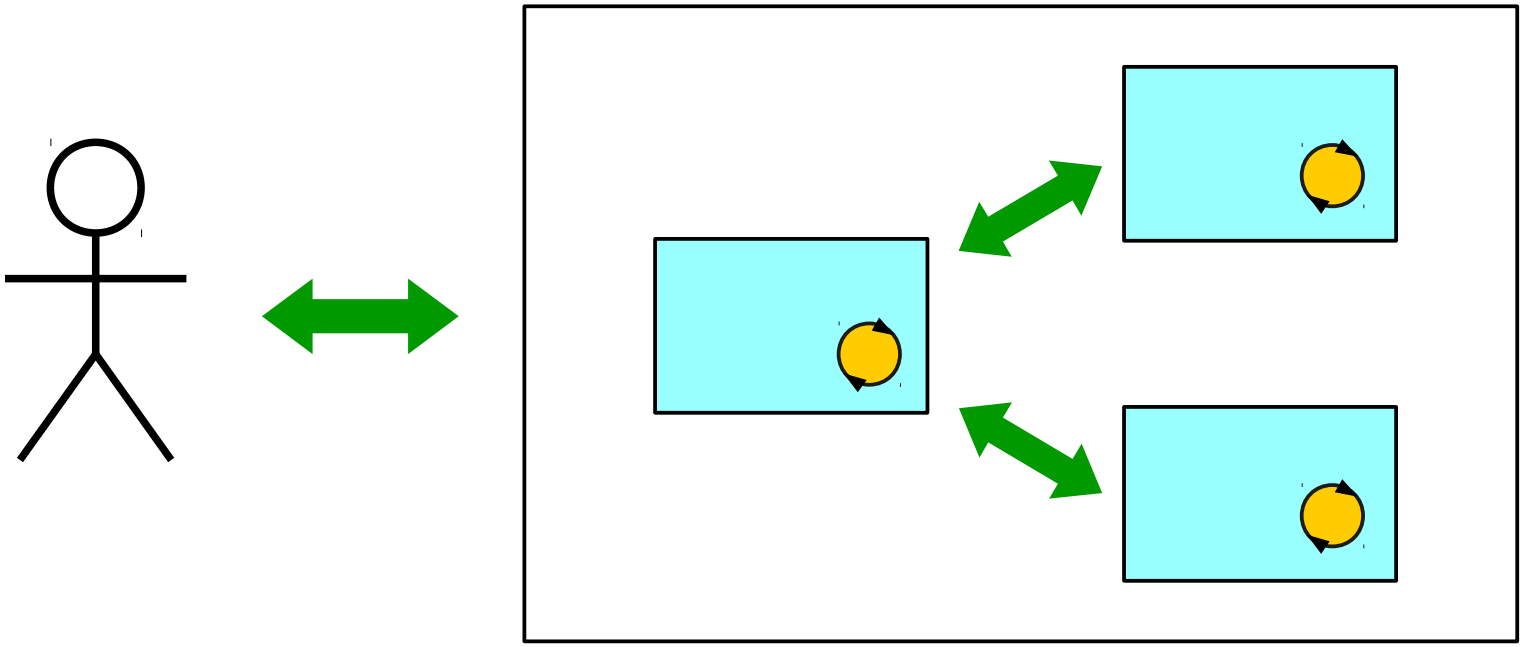


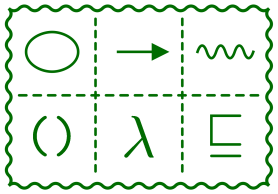
並行システム概要

- 並行システム
- 相互作用・振る舞い・プロセス
- 並行システムの正当性
- 並行システムの開発プロセス
- 並行システム開発における課題

並行システムとは

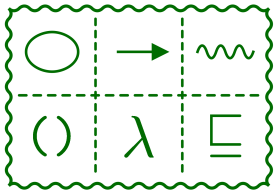
複数の構成要素からなるシステムで,
各構成要素が並行に動作し,
互いに作用を及ぼしあいながら
全体として目的の仕事をするシステムのこと





並行システムとは

- 並行
  - 各構成要素が自ら処理を行う能力を（実・仮想・疑似にかかわらず）持ち自律的に処理を行う
  - マクロ的な視点からは同時に複数の処理が行われているように見える
- 複数の構成要素
  - 複雑なシステムの構築技法：分割統治
  - 並行性を生かすため
    - 高速化
    - 応答性
- 相互作用
  - 構成要素間で情報の交換を行う．処理に影響する．
  - 並行システムと外部（ユーザ・他システム）とのやり取りも相互作用



並行システム：3つの問

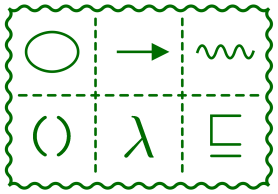
- 相互作用とは何か
- 並行システムの振る舞いとは何か
- 並行システムの正当性とは何か

並行システムとは

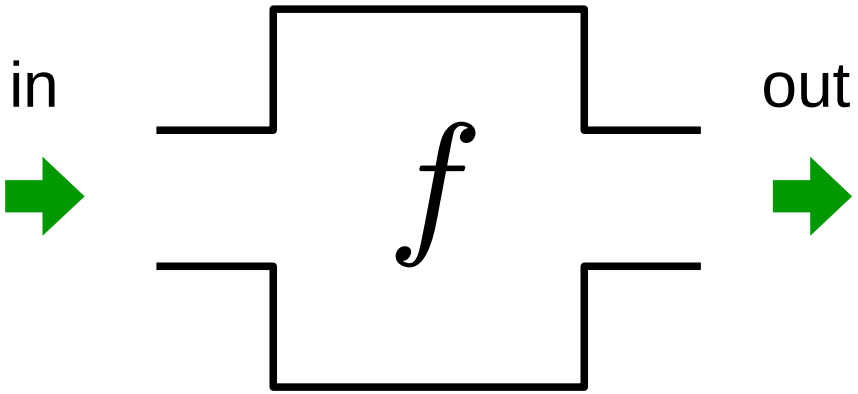
複数の構成要素からなるシステムで，各構成要素が並行に動作し，互いに作用を及ぼしあいながら全体として目的の仕事をするシステムのこと

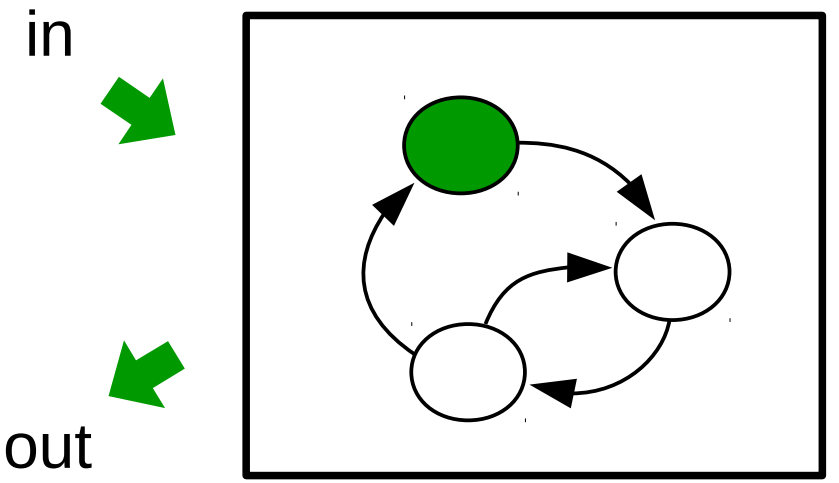
設計の作業 (2/3)

- 並行システムの振る舞いを表すモデルを作成すること
- 作成したモデルが与えられた要求や仕様を満たしていることを検証すること



計算システムとリアクティブシステム





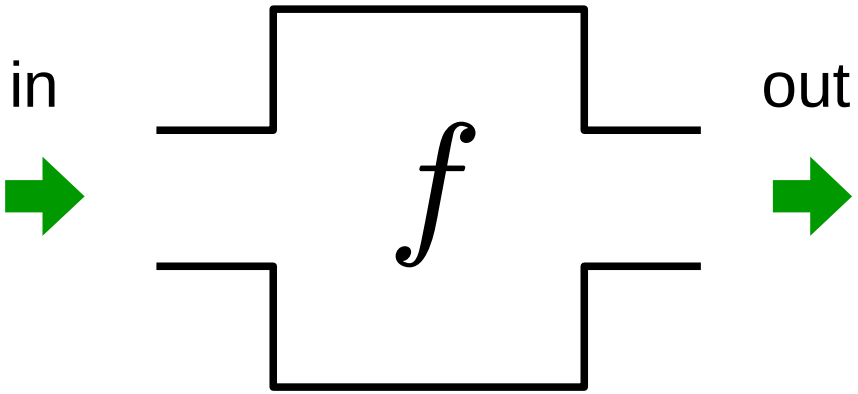
計算システム

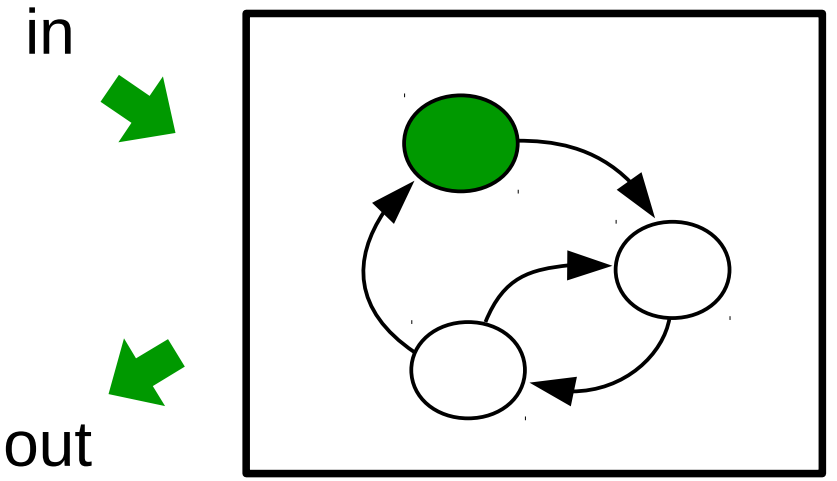
リアクティブシステム

「リアクティブシステムとは，システムの外部（システムの利用者や他のシステムなど）とのやりとりを行いながら計算の実行やサービスの提供を継続することを目的とするシステムである．」

–コンピュータサイエンス入門 ー 論理とプログラム意味論，岩波書店

計算システムとリアクティブシステム



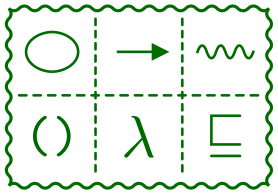


計算システム

リアクティブシステム

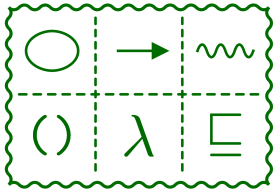
計算

計算 × コミュニケーション



並行システムとリアクティブシステム

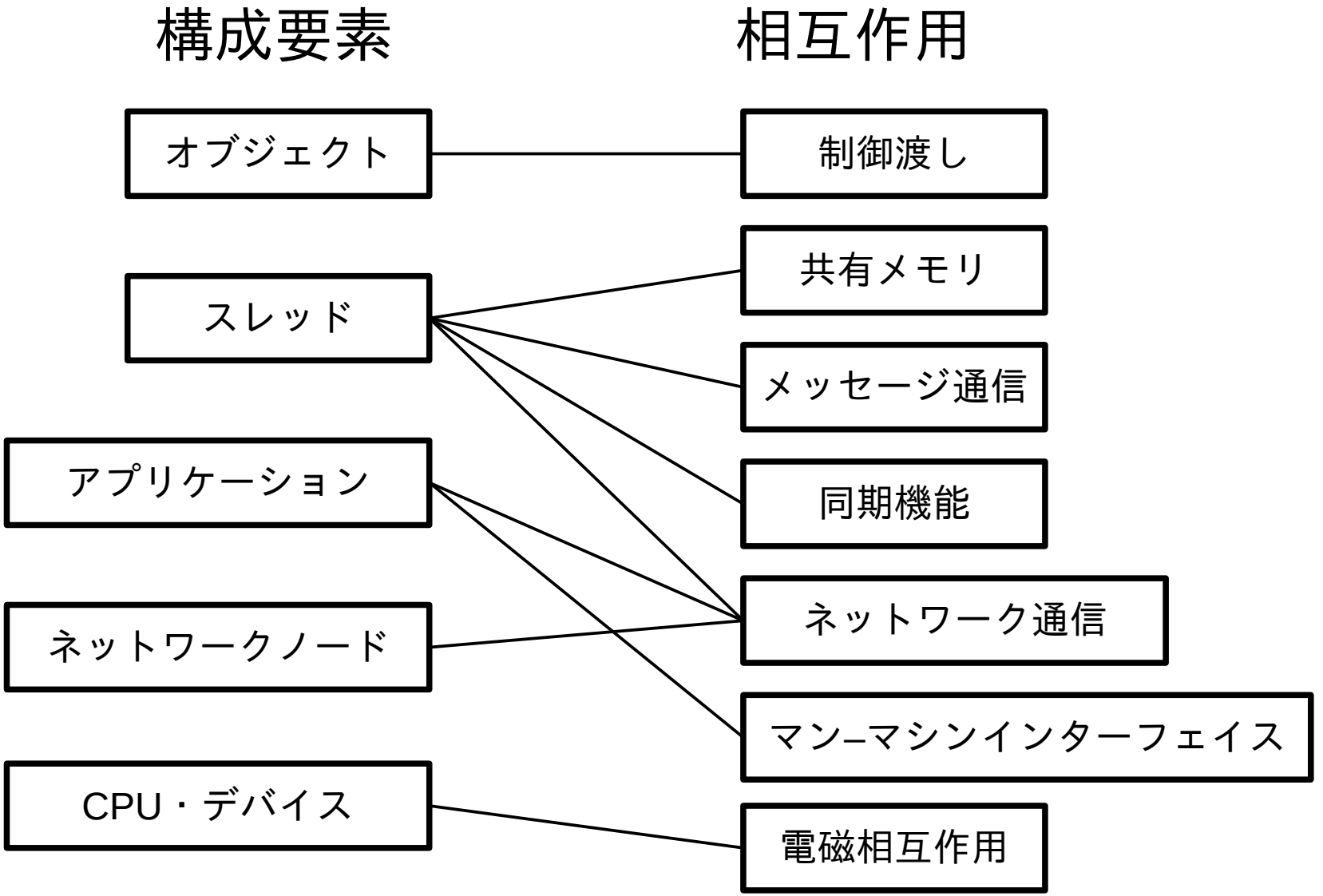
- 並行システムはリアクティブシステムの一種
  - 構成要素もリアクティブシステム：階層的・再帰的
- リアクティブシステムは内部状態を持つ状態遷移システムである
  - 内部状態によって同じ入力に対しても出力が異なる
- リアクティブシステムの2つの柱
  - 計算
  - コミュニケーション

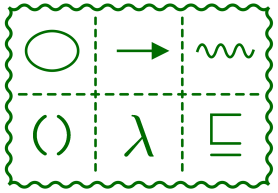


主な相互作用

- 制御渡し
- 共有メモリ
- メッセージ通信
- 同期機能
  - セマフォ
  - ミューテックス
  - 条件変数
- ネットワーク通信
- RPC
- ユーザとの対話
- CPU–デバイス間通信

システムの構成要素と相互作用



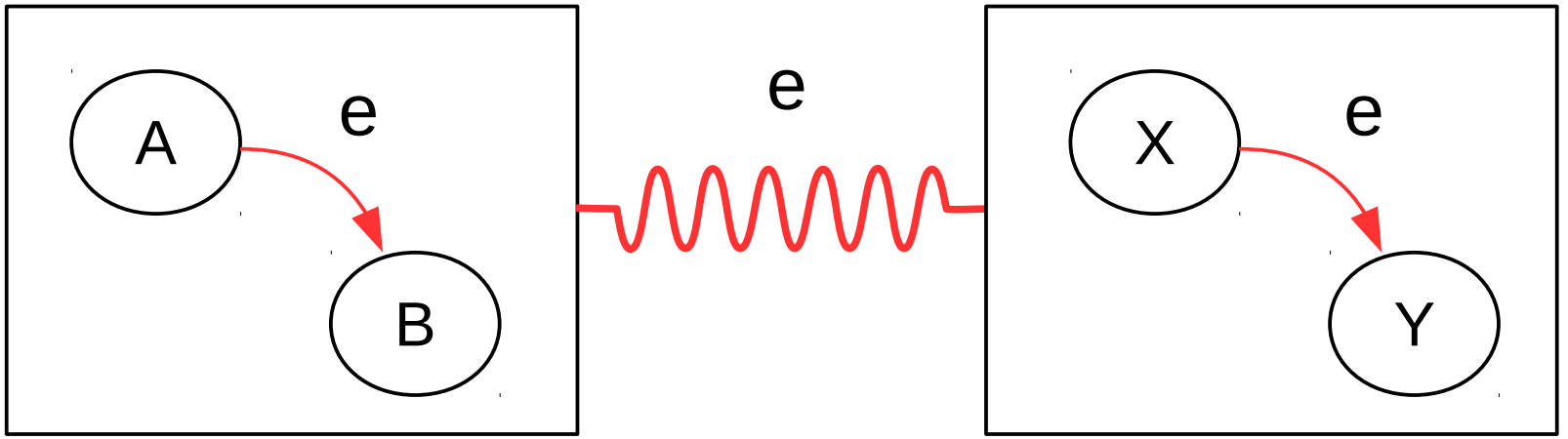


理論的相互作用

- 設計段階では，実装方法とは別に，様々なシステムを統一的に扱うことができる分析に適した相互作用で考えることが望ましい
- 基準
  - 意味が明確に定義されている
  - 振る舞いの記述が可能で，比較ができる
  - プリミティブとして他の相互作用を表現できる

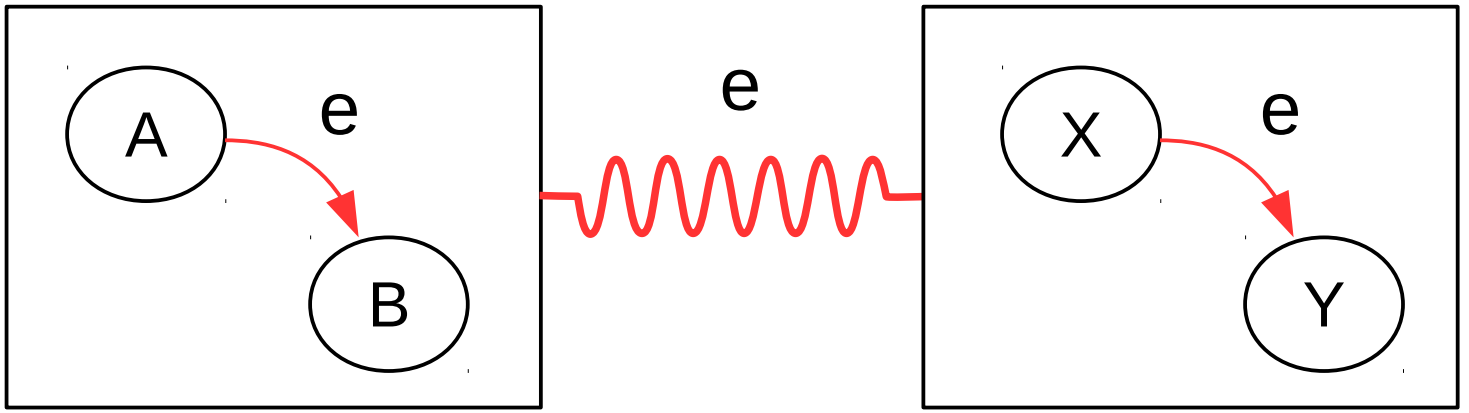
イベントによる同期型の相互作用

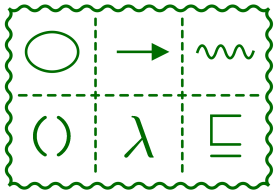
- 構成要素間で起こる相互作用の最小単位をイベントという
- イベントは名前を持ち，相互作用の種類を表す
  - 例：書き込み write，要求 req，応答 ack
  - 異なる遷移に同名のイベントをつけることができる
    - イベントは遷移の名前ではないということ



イベントによる同期型の相互作用

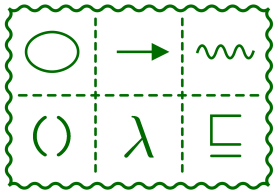
- 現状態からイベント e の付いた遷移があるとき，構成要素は「イベント e を提示している」という
- 2つの構成要素それぞれがイベント e を提示していて，対応する各遷移が同時に行われるとき，
  「イベント e が発生する」または
  「イベント e で同期する」という



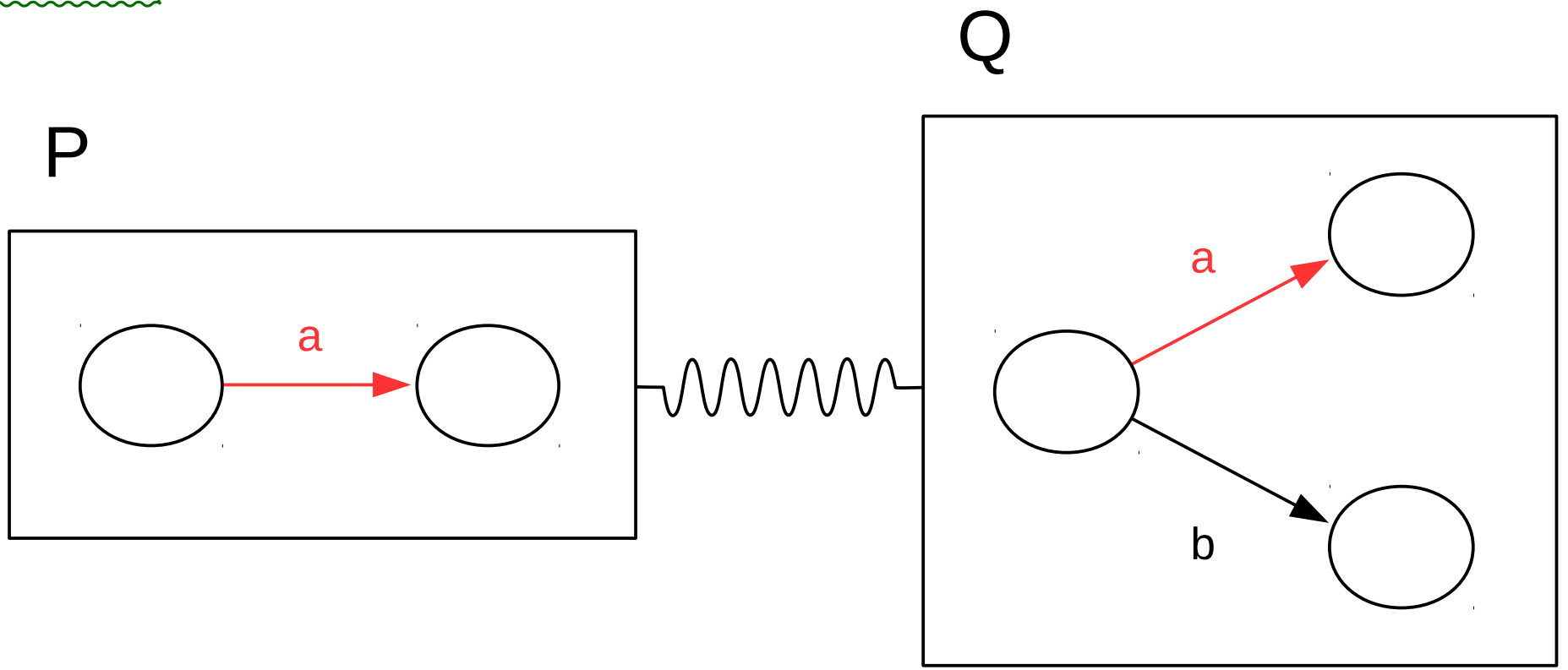


他の相互作用を表現する

- 選択
- 同期メッセージ通信
- 非同期メッセージ通信
- 共有メモリ
- 同期機能
  - ミューテックス
  - セマフォ
  - 条件変数



選択

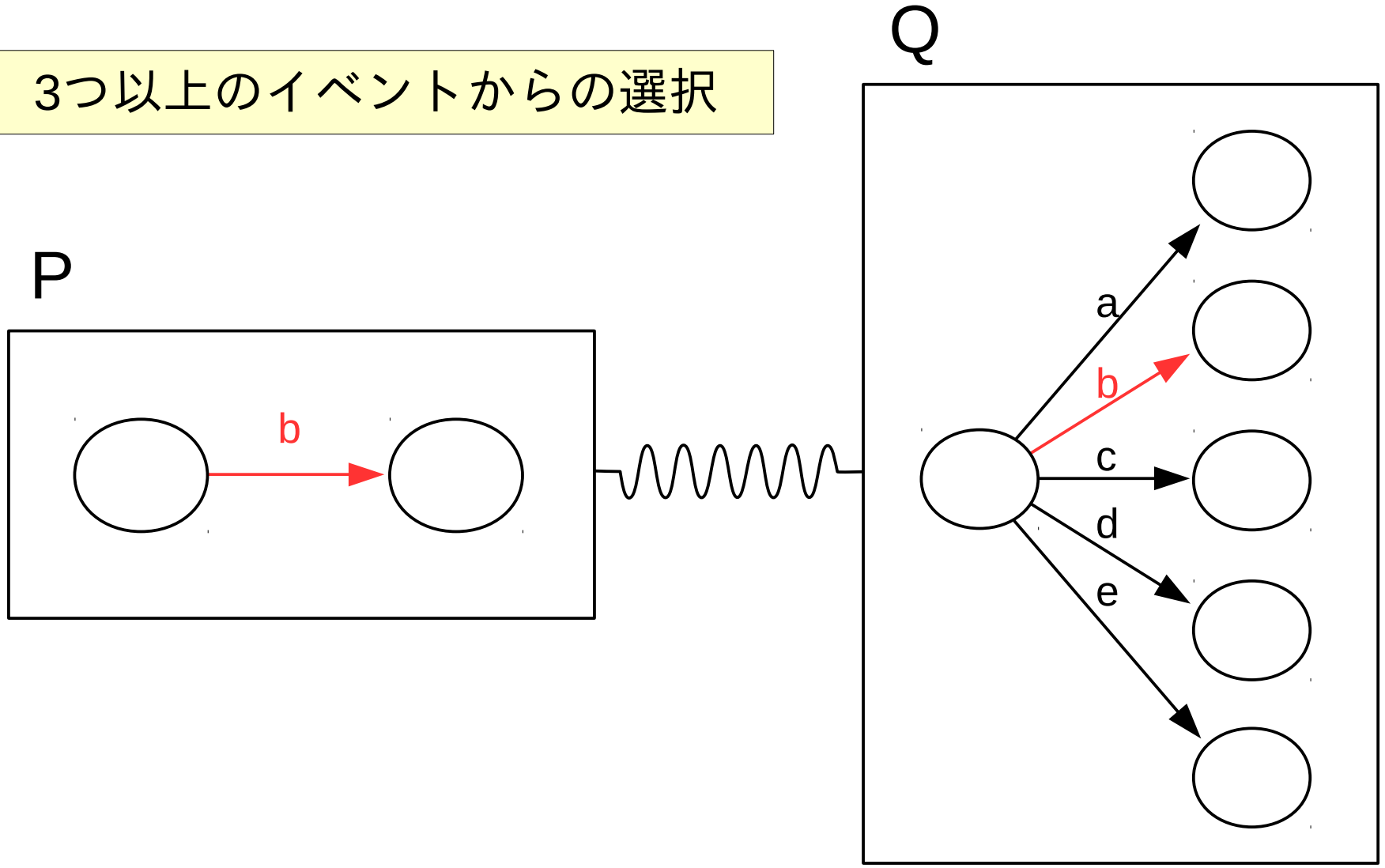


2つのイベント a, b を提示する

どちらが発生したかによって
相手が提示したイベントがわかる

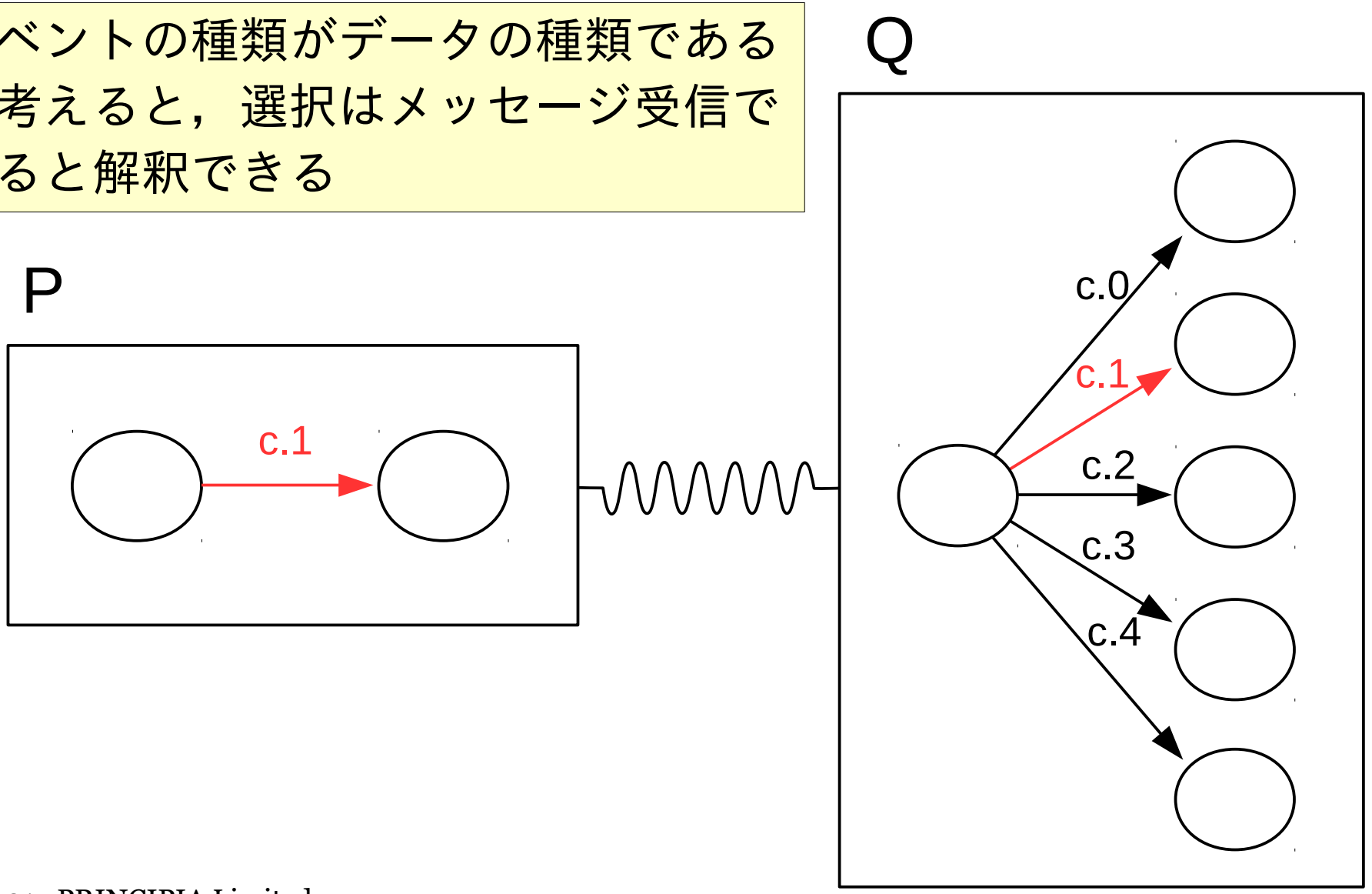
選択

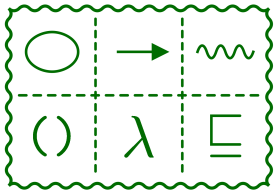
3つ以上のイベントからの選択



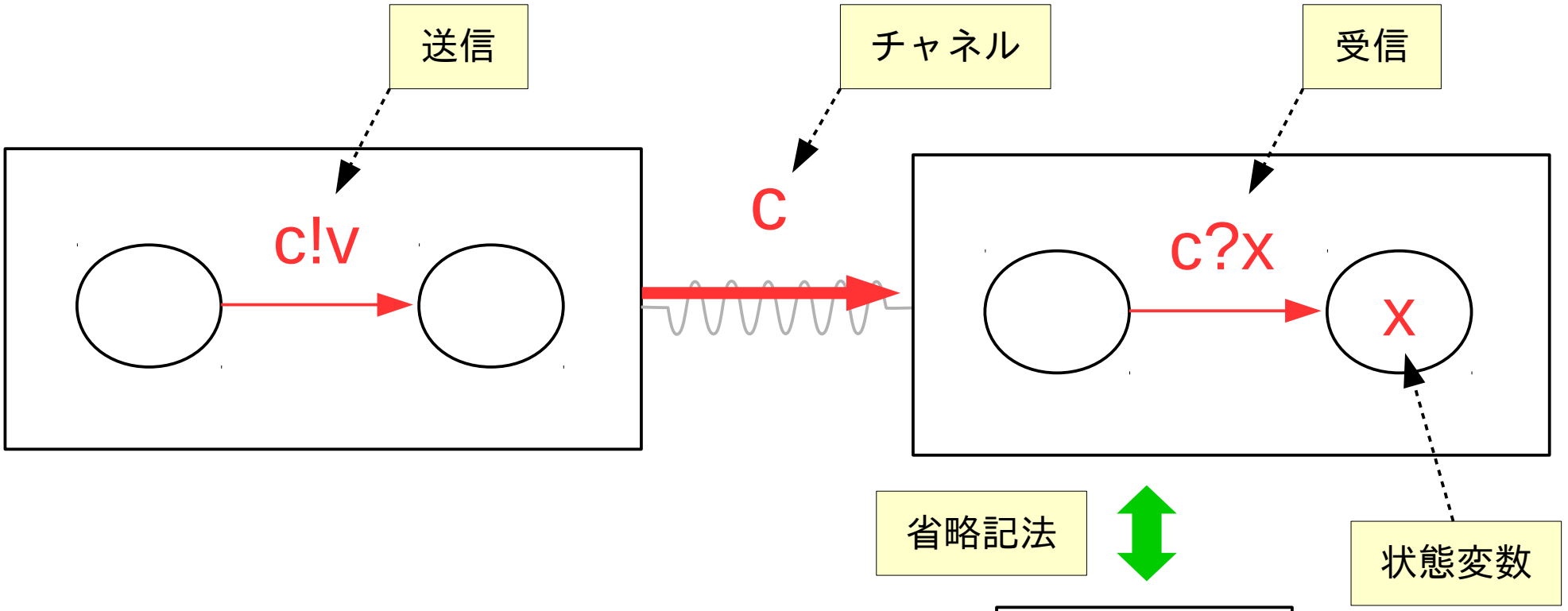
同期メッセージ通信

イベントの種類がデータの種類であると考えると，選択はメッセージ受信であると解釈できる

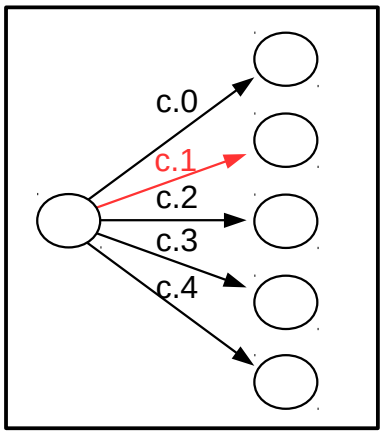


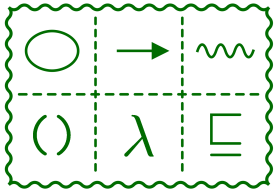


同期メッセージ通信とチャネル



チャネル c を通じてメッセージ送受信を行っていると解釈することができる.



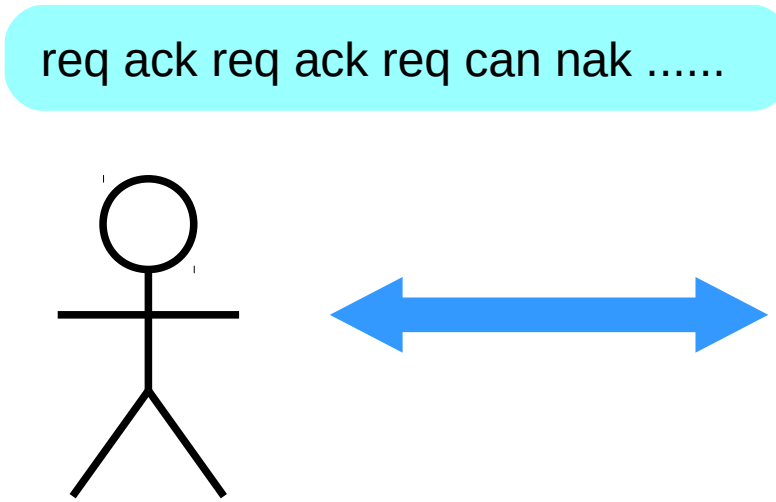


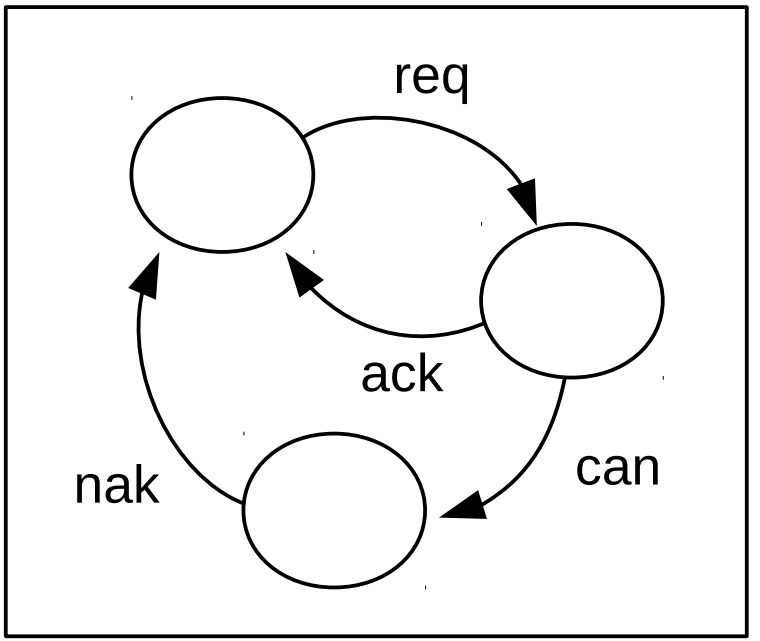
イベントによる同期型相互作用

- イベントによる同期型相互作用
  - 構成要素の遷移が同時に発生する（同期する）ことが相互作用であると考える
  - 遷移に付随するイベントが相互作用の意味を表す
  - 遷移が同期するかどうかはイベントによって定まる
  - 原理的には入出力の区別がなく構成要素間で対称
- 他の相互作用の表現
  - イベントによる同期型相互作用をプリミティブとして，他の相互作用を表現することができる
    - 選択
    - 同期型メッセージ通信

振る舞いとは何か

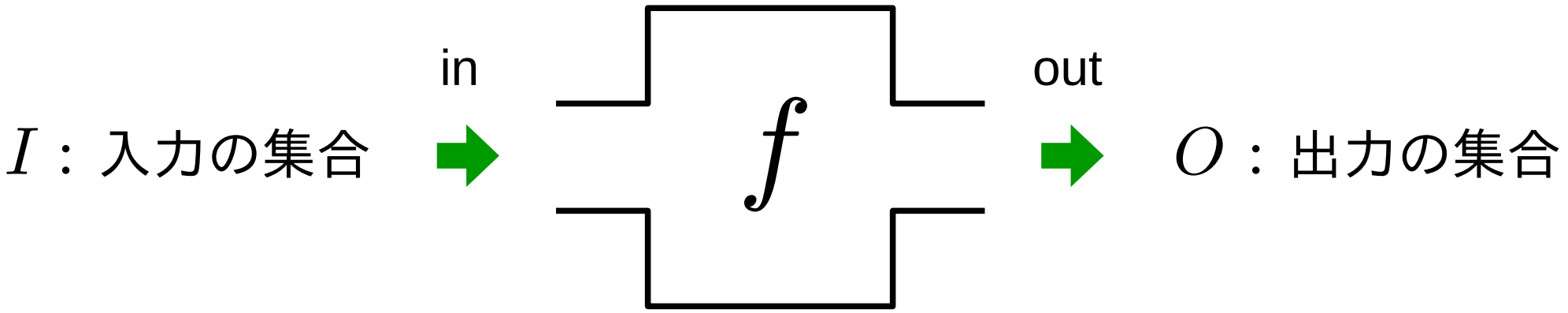
- リアクティブシステムの成果として期待されるもの
- 仕様で規定される
- 正当性検証の対象になる





比較：計算システムの成果

- 与えられた入力に対する出力：入出力の関係



計算システムの仕様：　$S \subseteq I \times O$

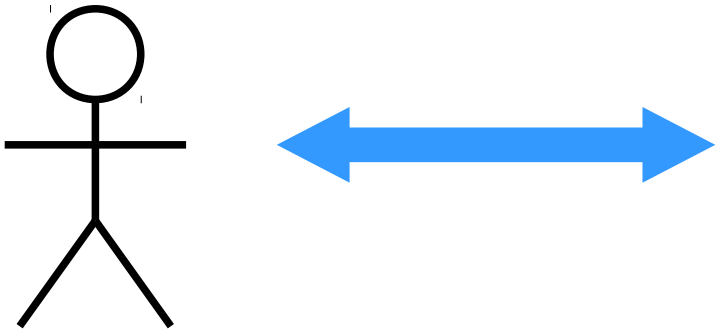
$$S = \{(i_1, o_1), (i_2, o_2), (i_3, o_3), \ldots\}$$

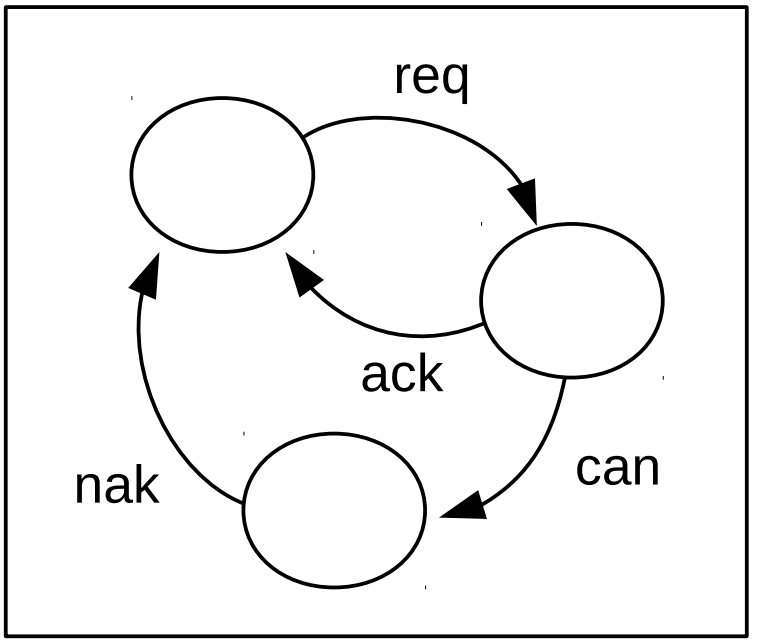
振る舞いとは何か（近似解）

※ この定義は後で修正される

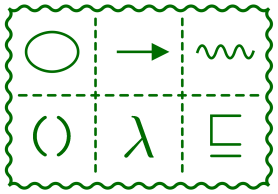
- システムが起動時から発生するイベントの列の集合
  - システムが起動時から発生するイベント列をトレースという
  - リアクティブシステムの振る舞いとは，システムがどのようなトレースを発生させるか，ということである

req ack req ack req can nak .....



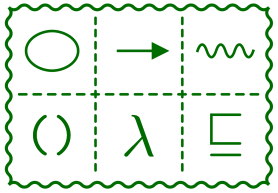


より正確には振る舞いの表現または意味（表示）という



プロセス

- リアクティブシステムの振る舞いをプロセスという
- 派生的な使用として，振る舞いの表現，特に状態遷移モデルのこともプロセスと呼ぶことがある
- さらに，振る舞いを行う主体である構成要素のこともプロセスと呼ぶことがある



並行システムの正当性

比較：計算システムの正当性

I：入力の集合　O：出力の集合

計算システムの仕様：　$S \subseteq I \times O$

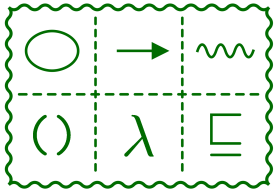
計算システムの実装：　$R \subseteq I \times O$

1. 仕様で規定されている入力をすべて受け付けること

$$\mathrm{dom}\, S \subseteq \mathrm{dom}\, R$$

2. 各入力に対する実装の出力は，仕様で規定されている出力の中に含まれていること

$$\forall i,\, o.\; i \in \mathrm{dom}\, S \wedge (i, o) \in R \Rightarrow (i, o) \in S$$



計算システムの正当性

$$S \sqsubseteq R \triangleq \mathrm{dom}\, S \subseteq \mathrm{dom}\, R \wedge \mathrm{dom}\, S \lhd R \subseteq S$$

1. 仕様で規定されている入力をすべて受け付けること

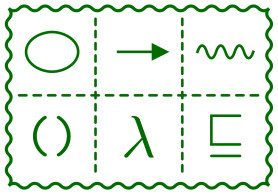
$$\mathrm{dom}\, S \subseteq \mathrm{dom}\, R$$

2. 各入力に対する実装の出力は，仕様で規定されている出力の中に含まれていること

$$\mathrm{dom}\, S \lhd R \subseteq S$$

定義域制限演算子

$$A \lhd R \triangleq \{(x, y) \mid x \in A \wedge (x, y) \in R\}$$



並行システムの正当性

$$S \sqsubseteq P \triangleq R \subseteq S$$

※ この定義は後で修正される

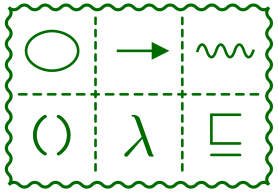
システムの仕様（トレースの集合）: S

システムの実装 P が生成しうるすべてのトレースの集合: R

計算システムの正当性条件 2 に近い
条件 1 に対応する部分がない．この点については後で修正する

トレースの集合を規定する2つの方法

- 状態遷移モデル　この講義では状態遷移モデルで仕様を記述する
- 時相論理

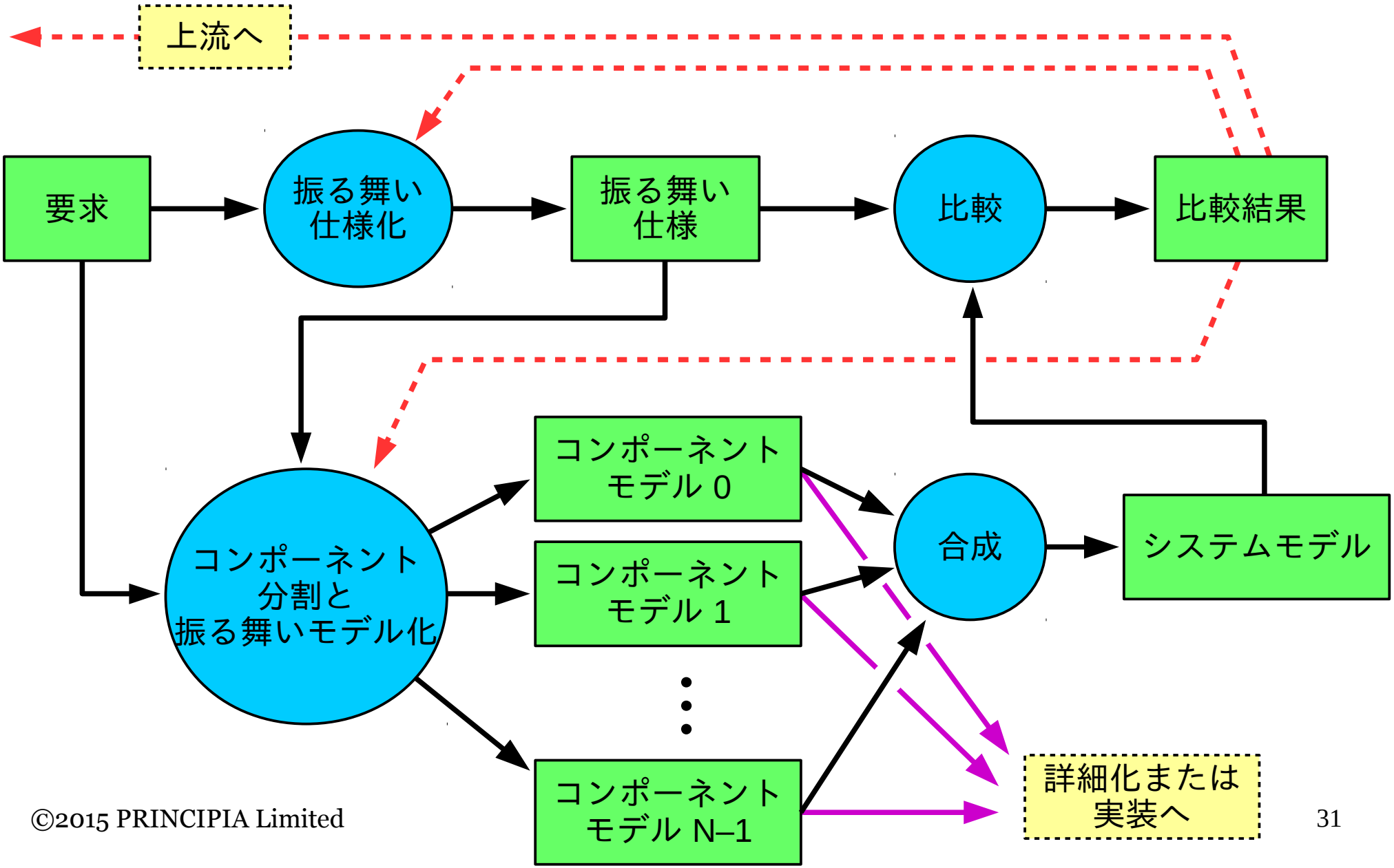


並行システム：3つの問と答え

- 相互作用とは何か
  - 構成要素の遷移が同時に発生すること
    （イベントによる同期型相互作用）
- 並行システムの振る舞いとは何か
  - システムが発生しうるすべてのトレースの集合※
- 並行システムの正当性とは何か
  - システムが発生しうるトレースがすべて仕様に含まれていること※

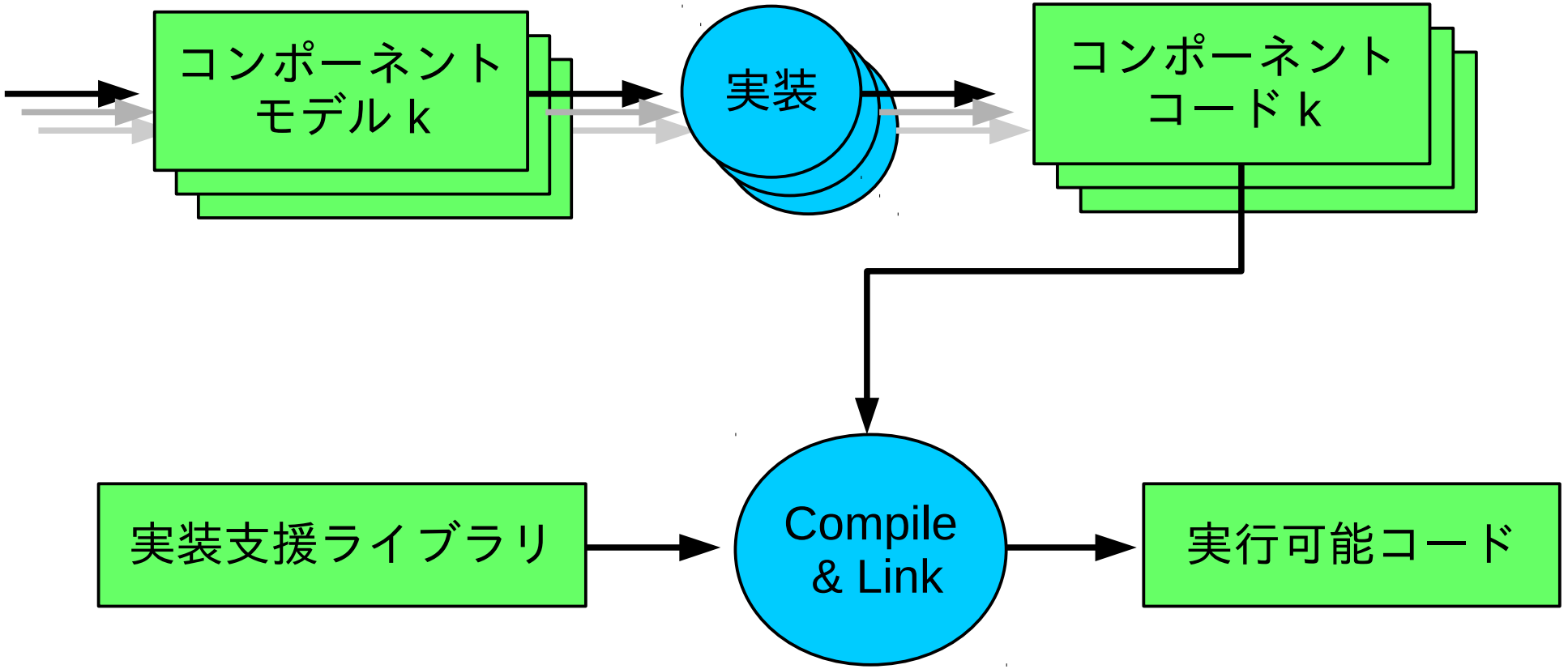
※ この定義は後で修正される

システムの設計（振る舞い側面）

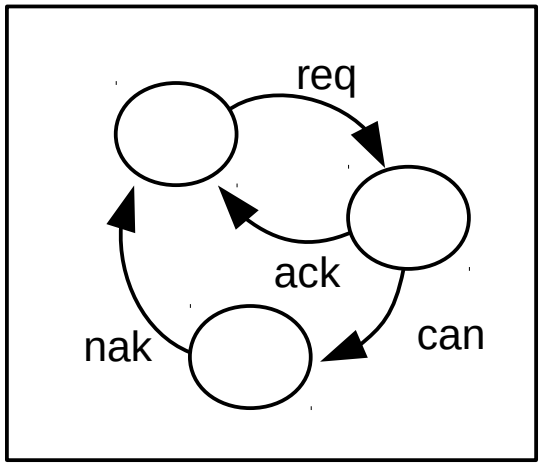


モデルから実装へ

上流から



並行システムの正当性検証



仕様

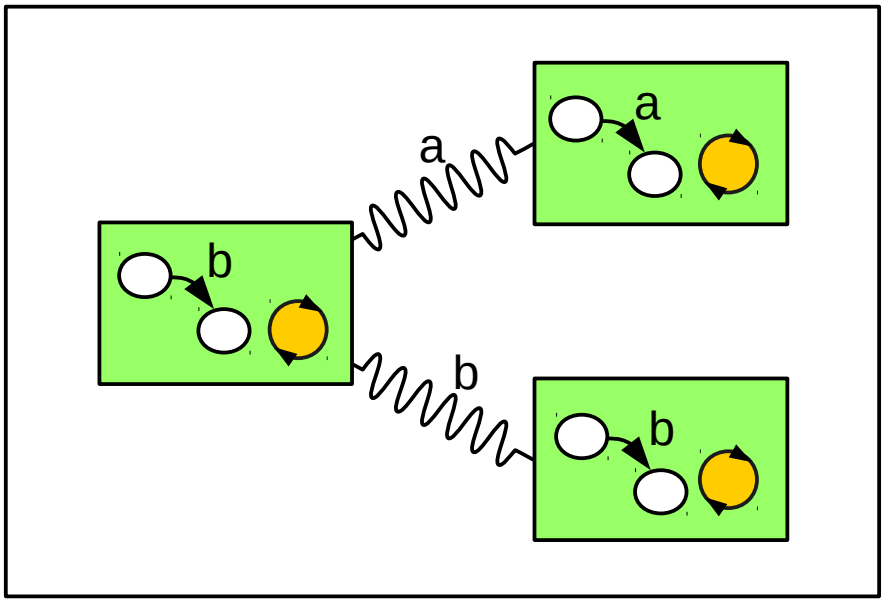
比較

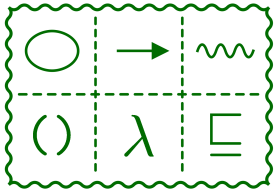
システムモデル

合成

並行システムの理論 CSP で
合成と比較が定義されている

コンポーネント
モデルの集合



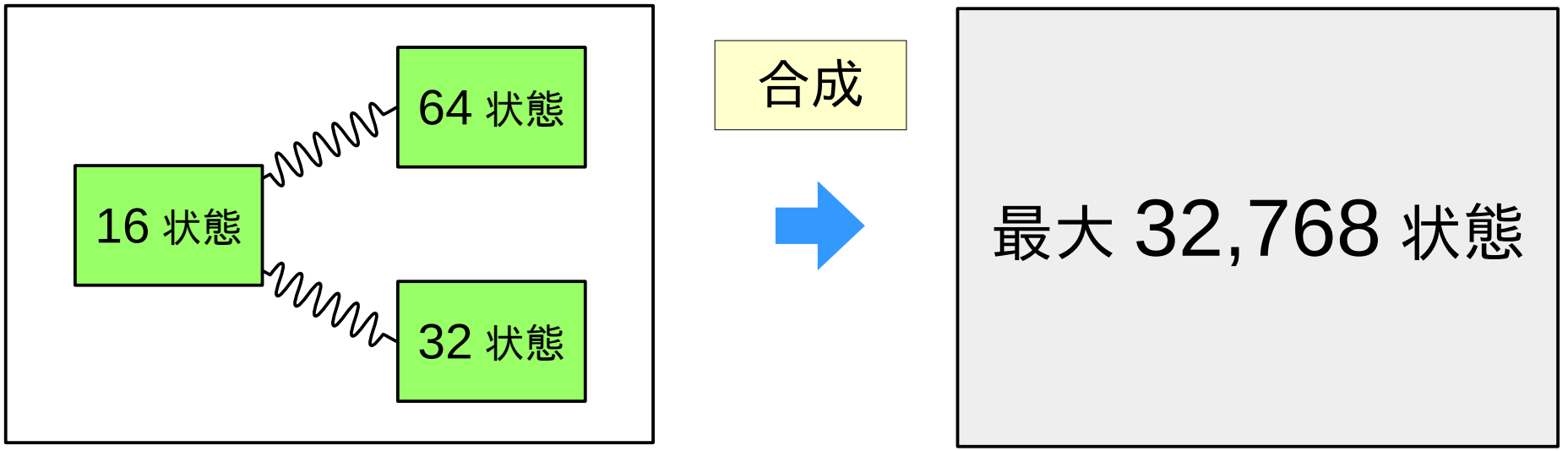


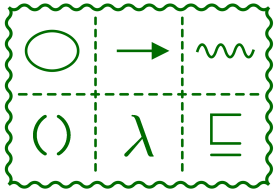
並行システム開発の課題

- 状態数の組み合わせ爆発
- 非決定性
- デッドロック
- ライブロック（発散）

状態数の組み合わせ爆発

- 合成したシステムの状態数は，相互作用による制約がなければ各コンポーネントの状態数の積になるので，コンポーネント数に対して指数関数的に増加する



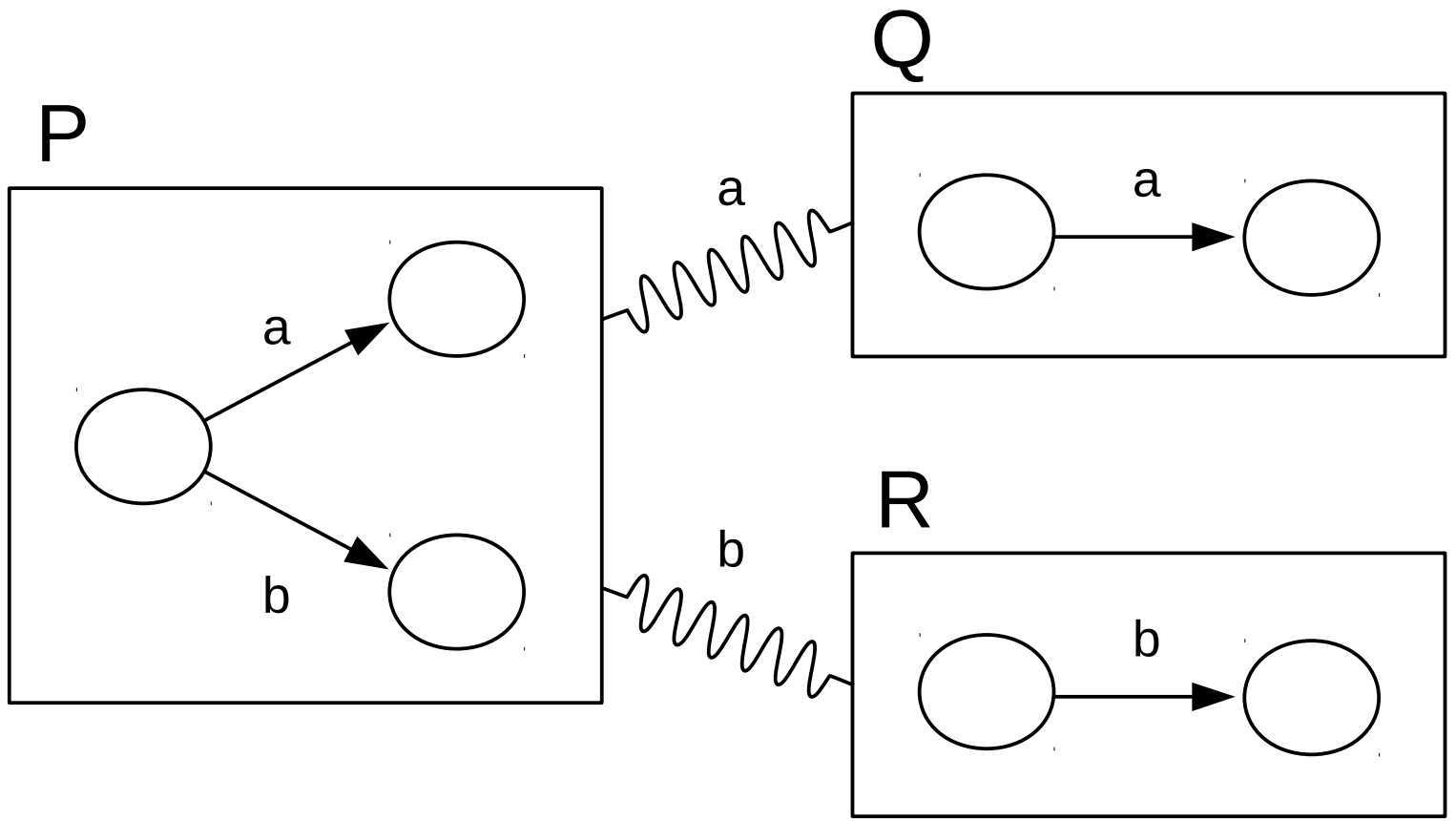


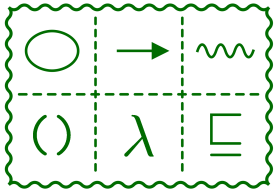
状態数の組み合わせ爆発

- 課題
  - 状態数が多いため網羅的にテストすることが難しい
- 対策
  - ツールと計算機のパワーを使って検査する
  - 抽象化を行い状態数を減らす

非決定性

- 同じ状況下で同じイベントを提示しても，そのたびに発生するイベントが異なる場合がある



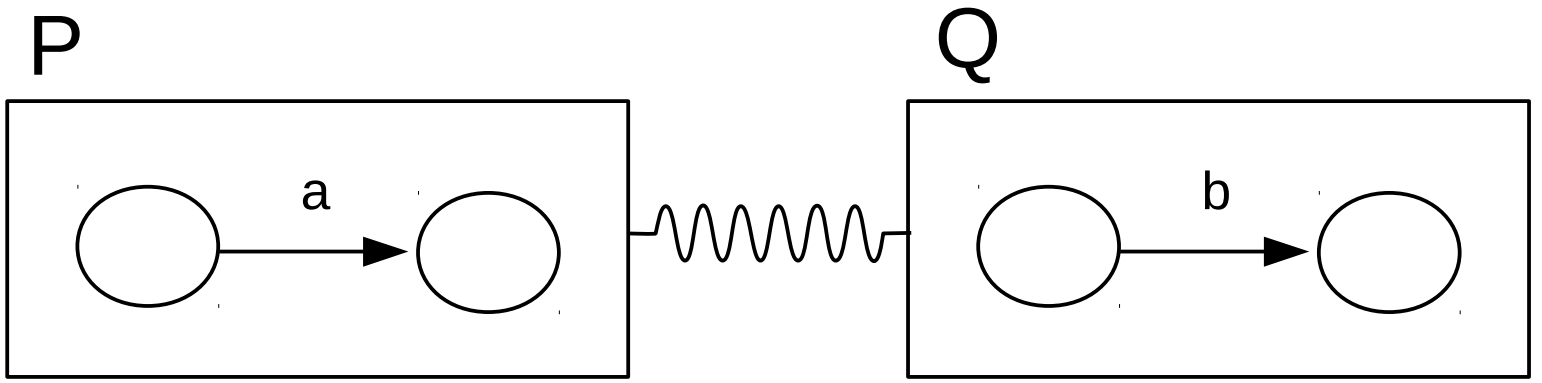


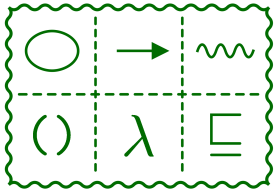
非決定性

- 課題
  - テスト条件を整えても結果が異なることがあるので，テストだけでは十分な検証ができない
- 対策
  - 理論に基づき，非決定性がある場合でも結果が保証されるような検証を行う

デッドロック

- 各コンポーネントは可能な遷移を持っているにもかかわらず，同期できるイベントがないためにシステム全体としては動作できない状態



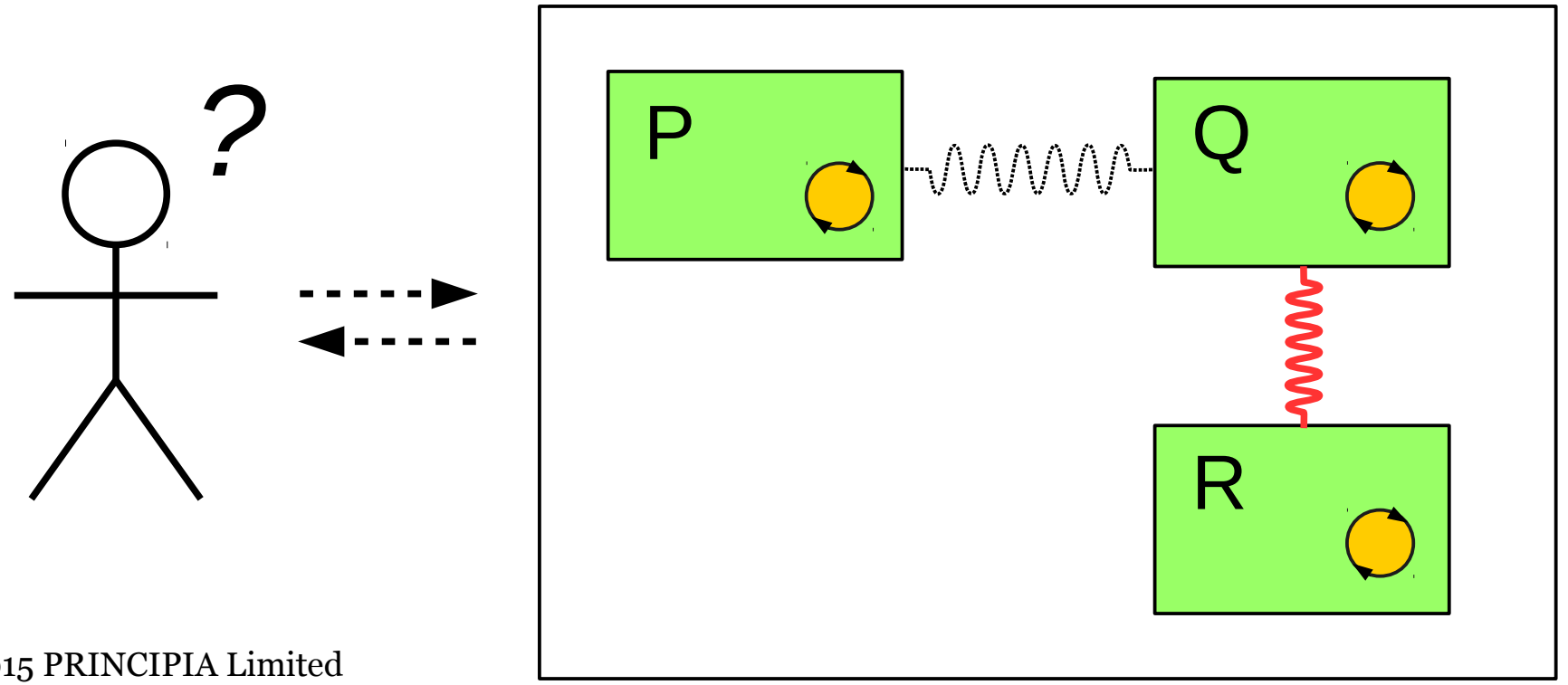


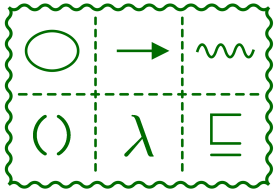
デッドロック

- 課題
  - デッドロックは相互作用の結果として発生するため，個々のコンポーネントを調べただけでは発見しにくい
- 対策
  - ツールを使ってシステムのモデルを作成し，デッドロックが存在しないことを確認する

ライブロック（発散）

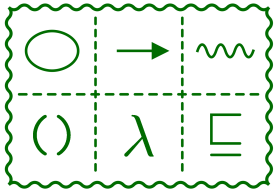
- 一部のコンポーネントが進捗のない相互作用を繰り返していることにより，システムが機能できなくなる状態





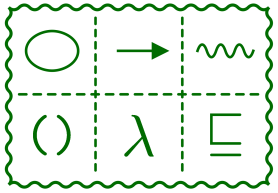
ライブロック（発散）

- 課題
  - ライブロックは相互作用の結果として発生するため，個々のコンポーネントを調べただけでは発見しにくい
- 対策
  - ツールを使ってシステムのモデルを作成し，ライブロックが存在しないことを確認する



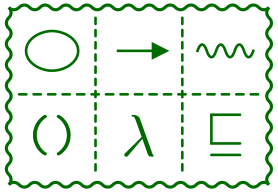
並行システムまとめ

- 並行システム
  - 複数の構成要素
  - 並行
  - 相互作用
- イベントによる同期型相互作用
- プロセス
- トレース
  - システムが起動してから発生するイベント列
- 並行システムの正当性
- 並行システムの開発プロセス
- 並行システム開発における課題
  - 状態爆発
  - 非決定性
  - デッドロック
  - ライブロック（発散）



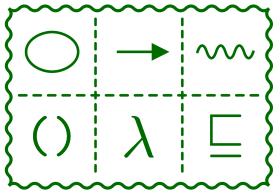
並行システムのまとめ２

- 並行システムは計算とコミュニケーションの2つの柱からなる
- 並行システムの正当性は外部から観測可能なコミュニケーションにもとづいて判断する
  - この点が計算を主とし，計算結果で正当性を判断するシステムと異なる
  - 状態や計算結果は観測の直接的な対象にはならず，コミュニケーションを通じて観測される

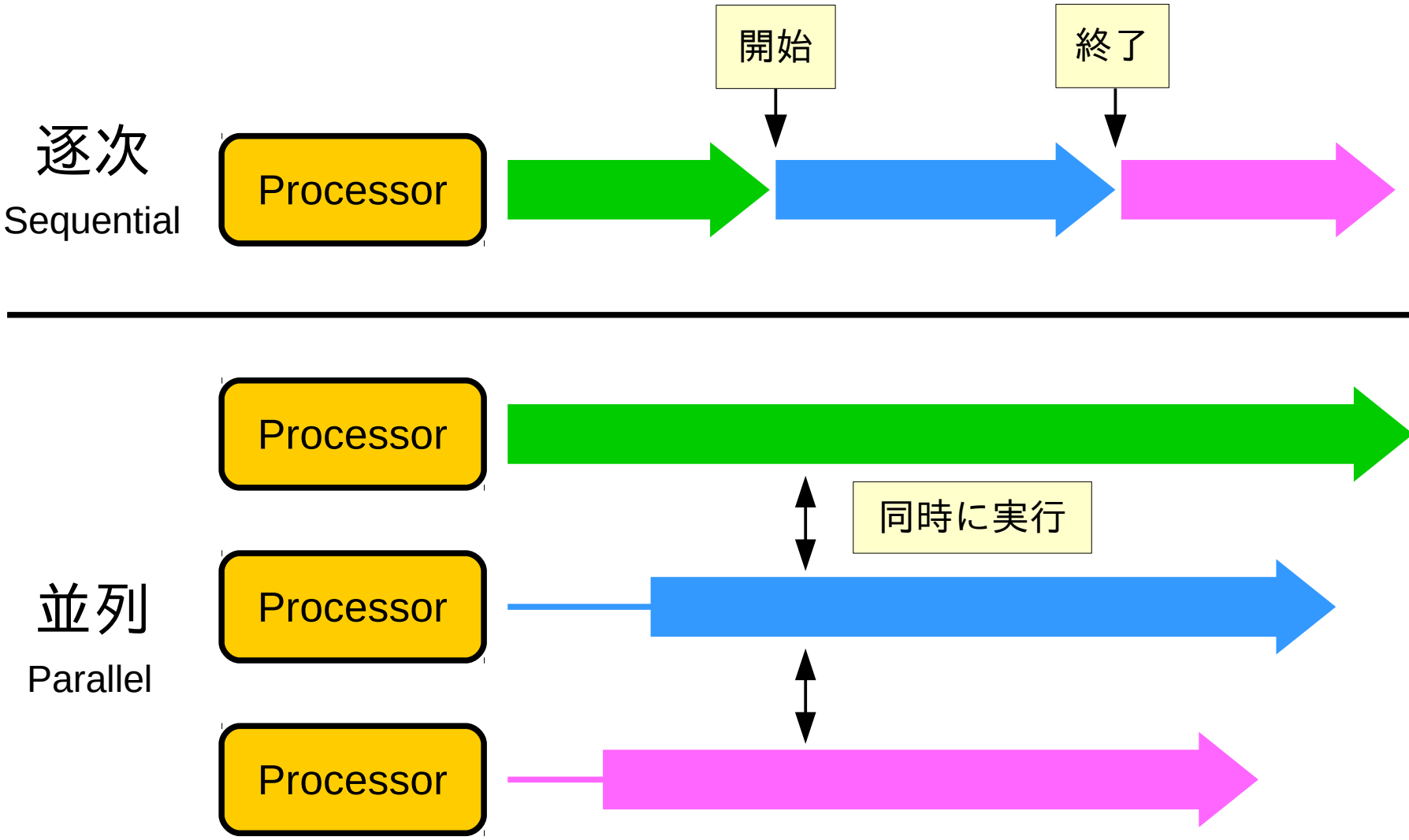


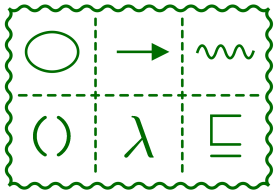
Appendix

- 逐次・並列・並行



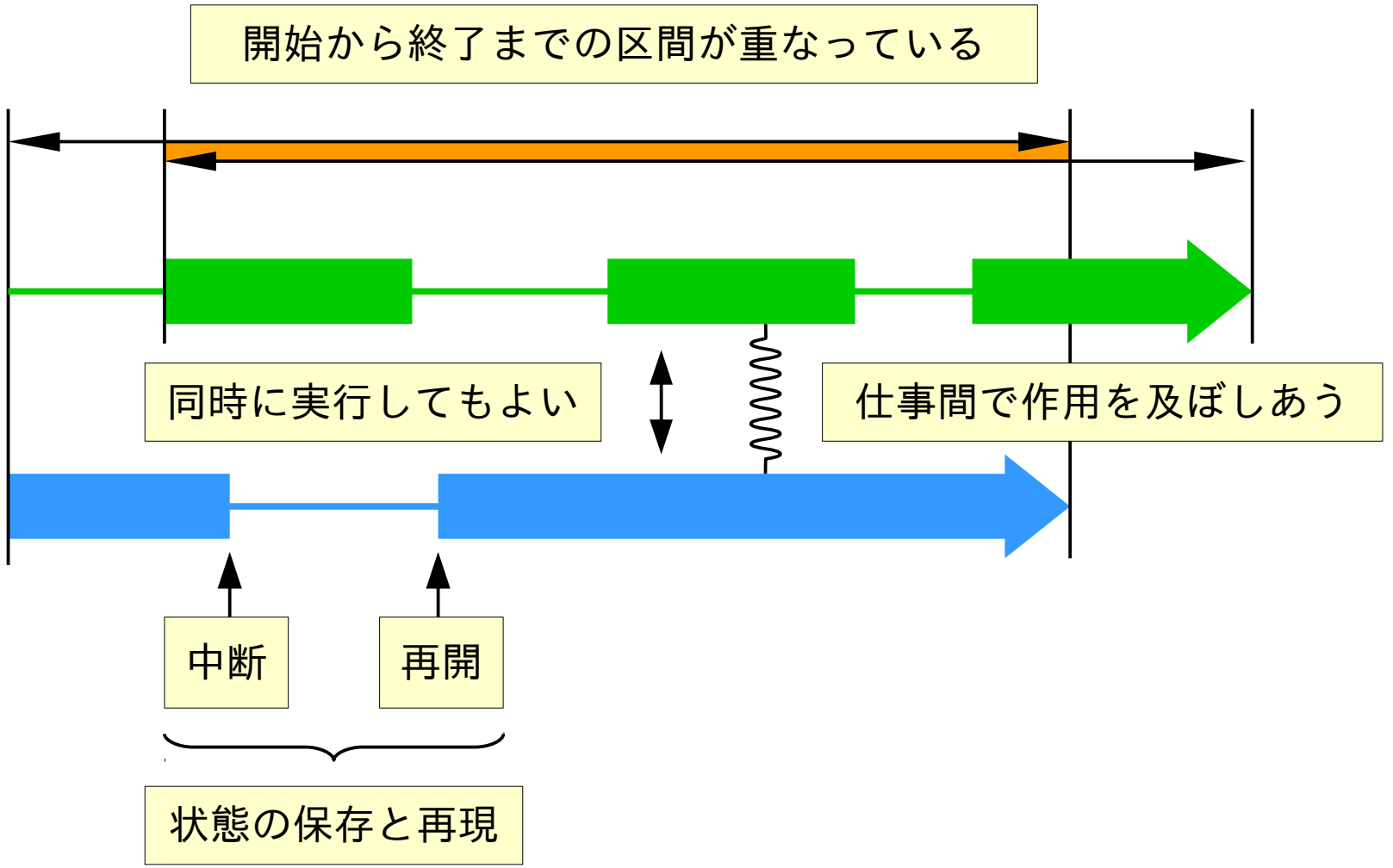
逐次と並列

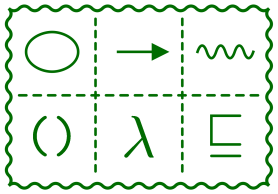




並行

並行
Concurrent





疑似並列（時分割）