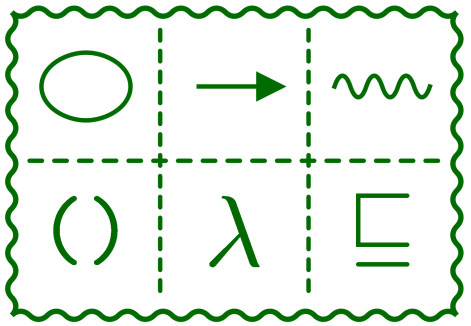
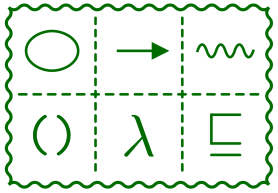


並行システムの検証と実装

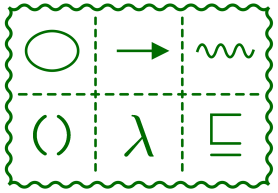
第0章 並行システム

PRINCIPIA Limited
初谷 久史



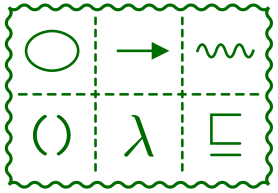
この講義で学習すること

- 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



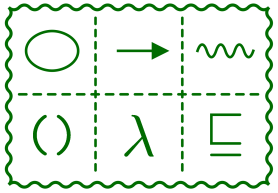
この講義で学習すること

- サブゴール
  - 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
  - 並行システムの振る舞いをモデル化し CSP 理論に基づいて検証するツールの使い方を学ぶ
  - 並行システムの実装方法について学ぶ



この講義で学習すること

- 並行システム開発を支援する3つの武器
  - 理論
    - Communicating Sequential Processes (CSP)
  - ツール
    - 構造と振る舞いのモデリング
    - 検査と結果の分析
  - 実装支援
    - （プログラミング言語）
    - ライブラリ



講義計画

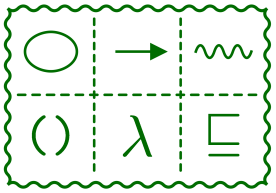
1 並行システム概説

2
3
4
5
6 プロセスのモデル化

7
8
9
10 詳細化検査

11
12 CSP理論

13
14
15 並行システムの実装

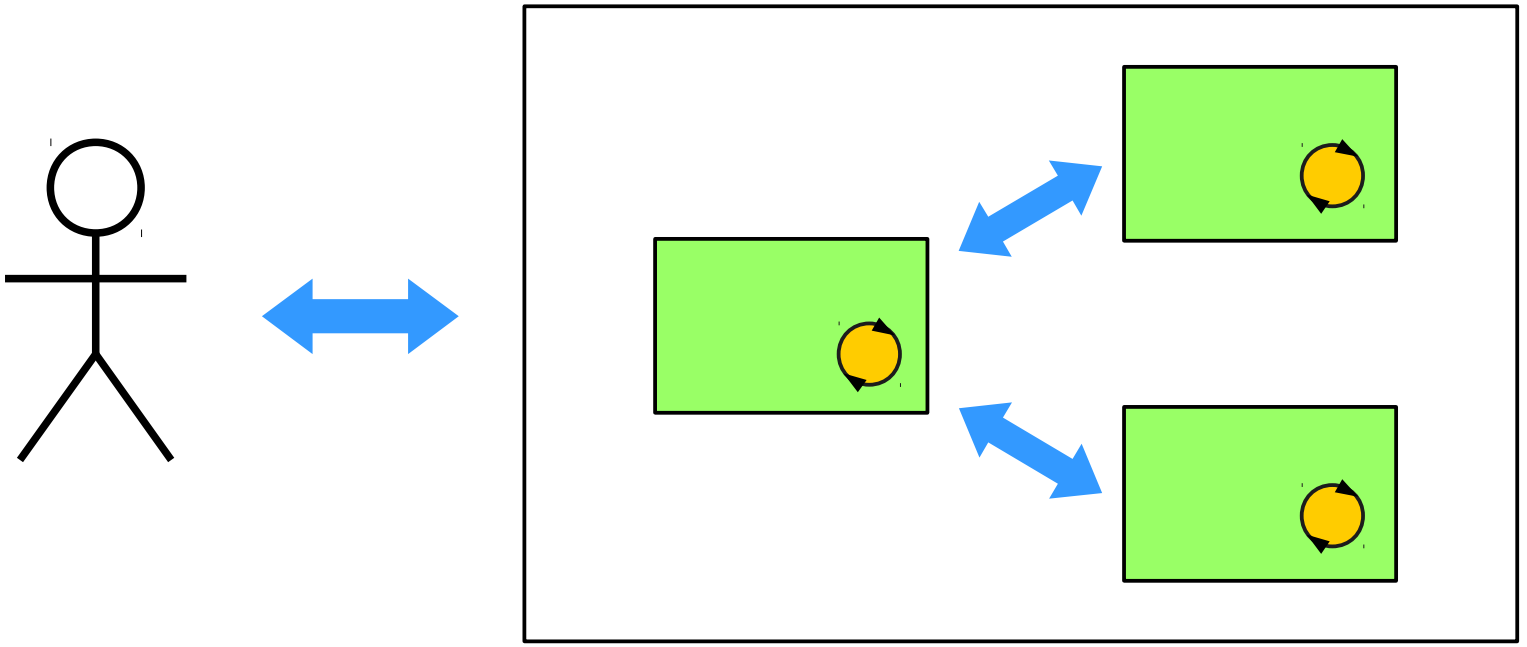


並行システム概要

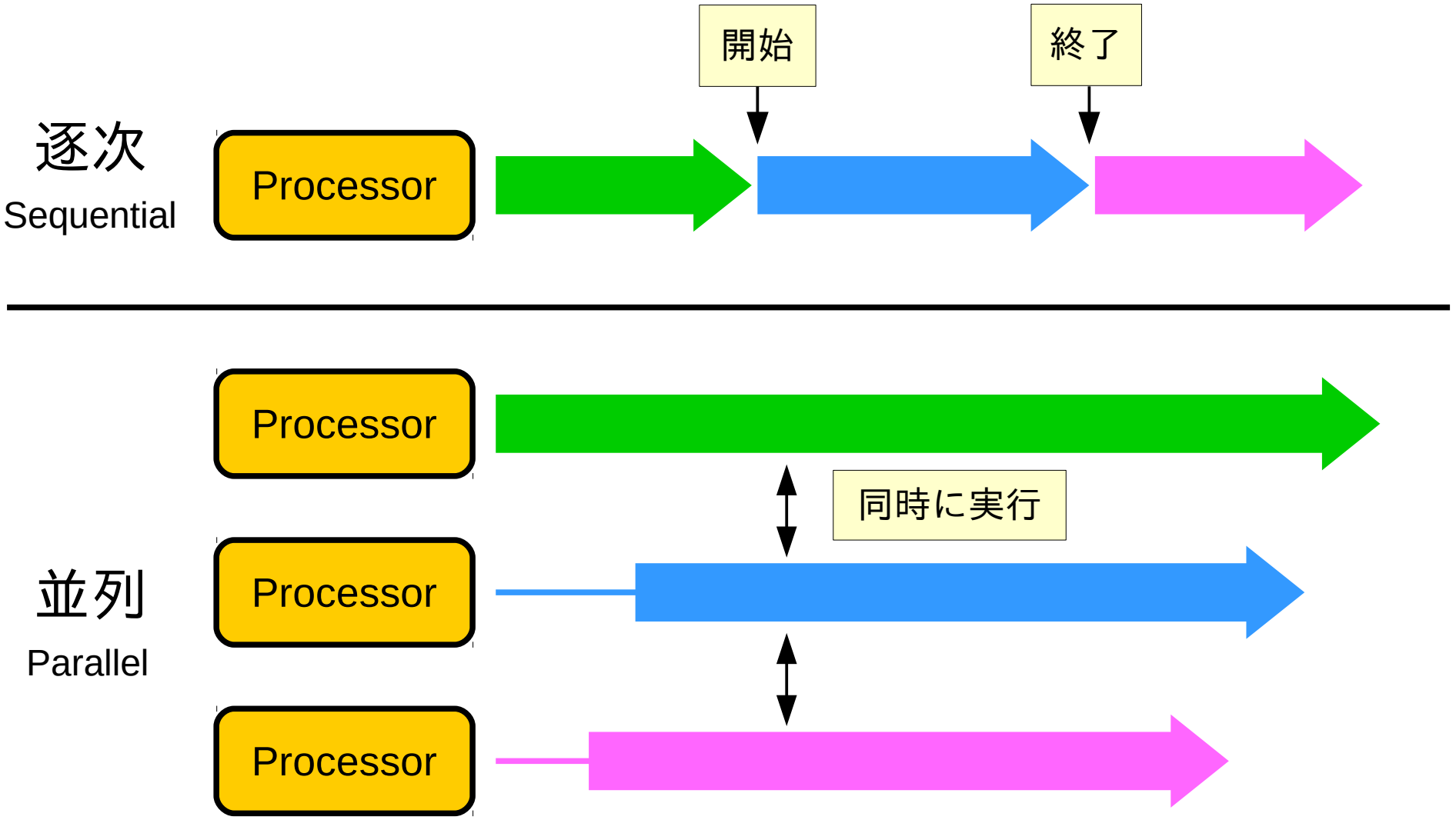
- 並行システム
- プロセス，相互作用
- 並行システムの設計
- プロセスの合成
- 正当性
- 並行システム開発における課題

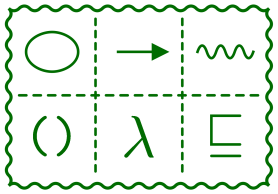
並行システムとは

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



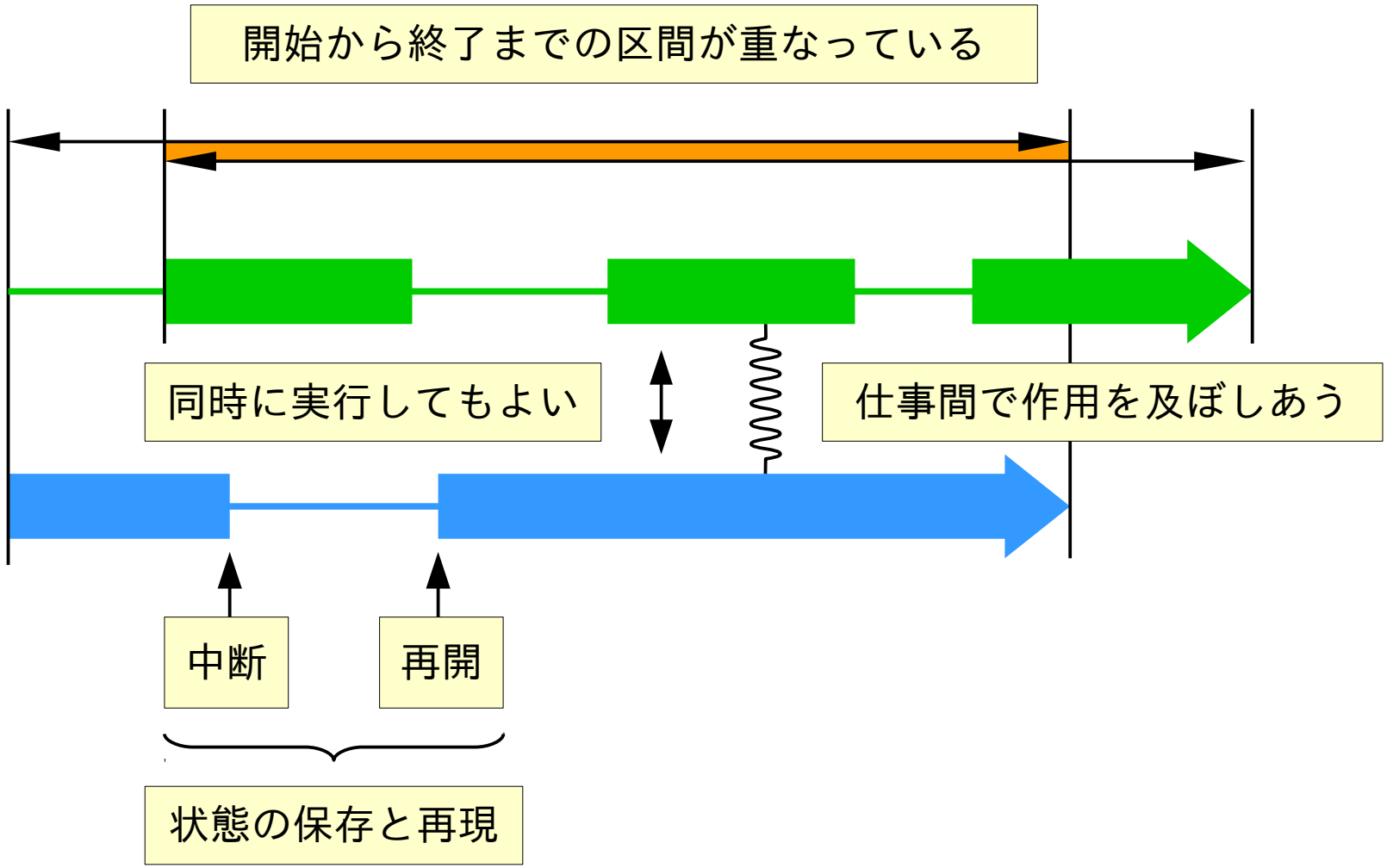
逐次と並列



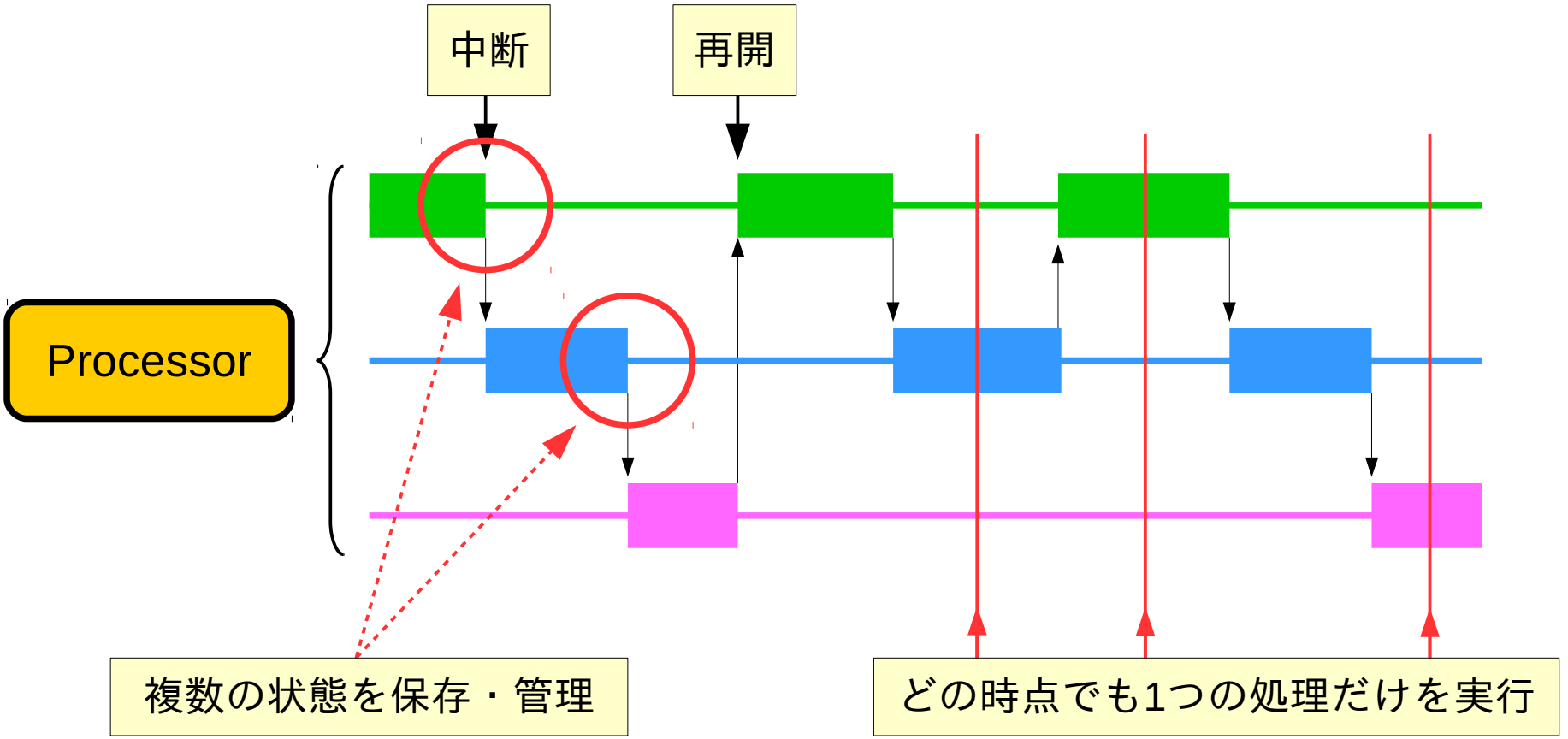


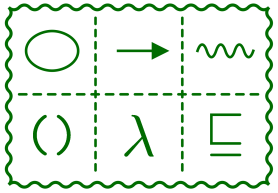
並行

並行
Concurrent



疑似並列（時分割）



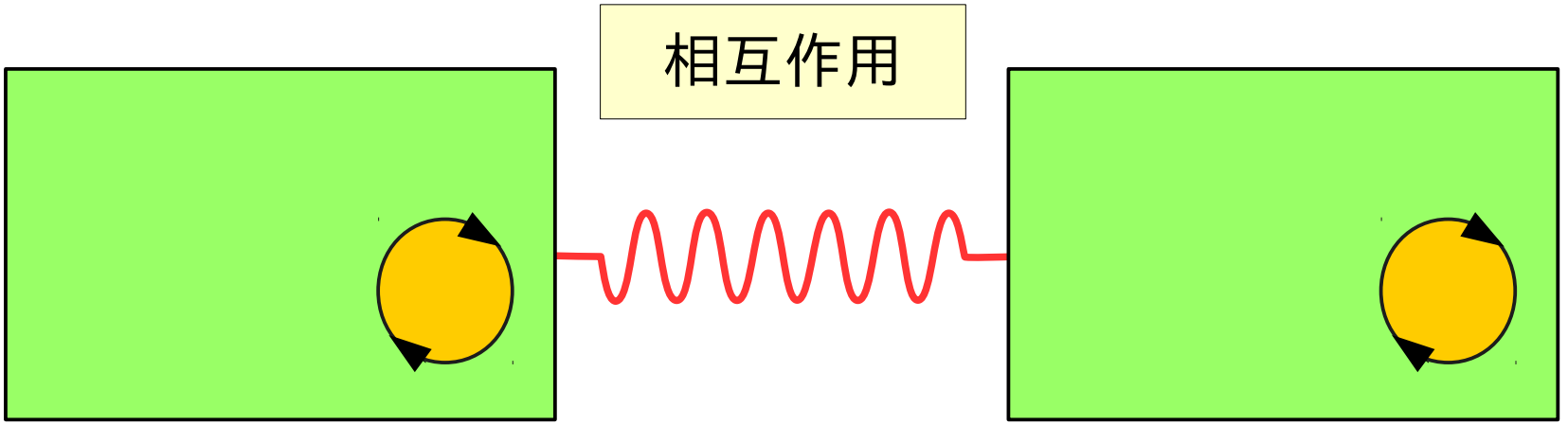


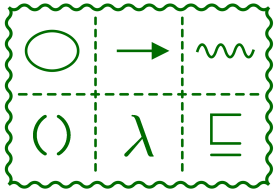
なぜ並行システムを扱うのか

- 重要な応用分野
  - ネットワーク
  - 組込みシステム
- 高速化
- 応答性
- システムの構築技法

逐次システムと並行システムの違い

- 複数の構成要素からなる
- 各構成要素が並行に動作する
- 構成要素がお互いに作用を及ぼしあう

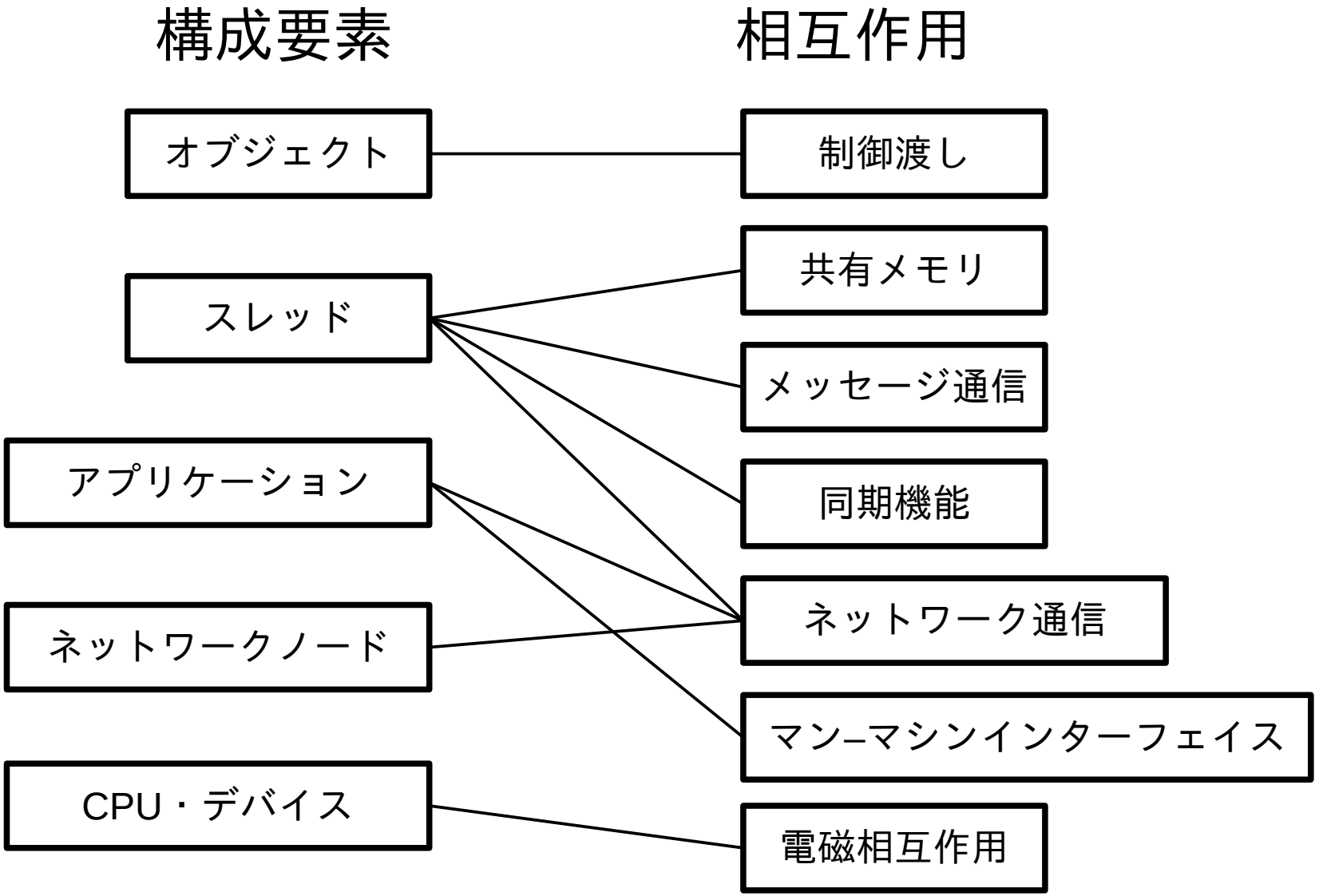


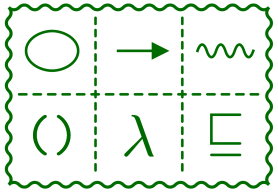


主な相互作用

- 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

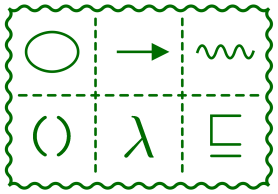
システムの構成要素と相互作用





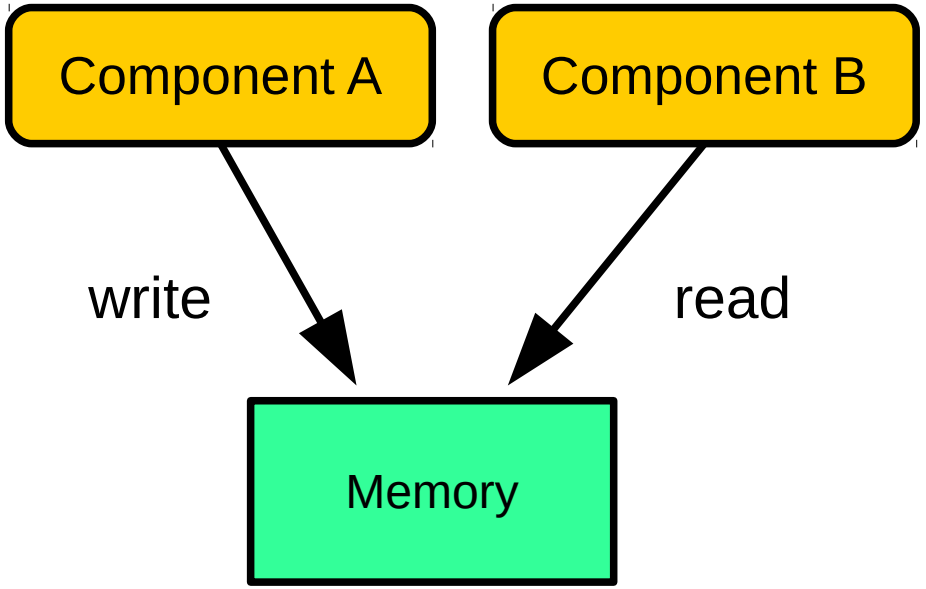
理論的相互作用

- 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

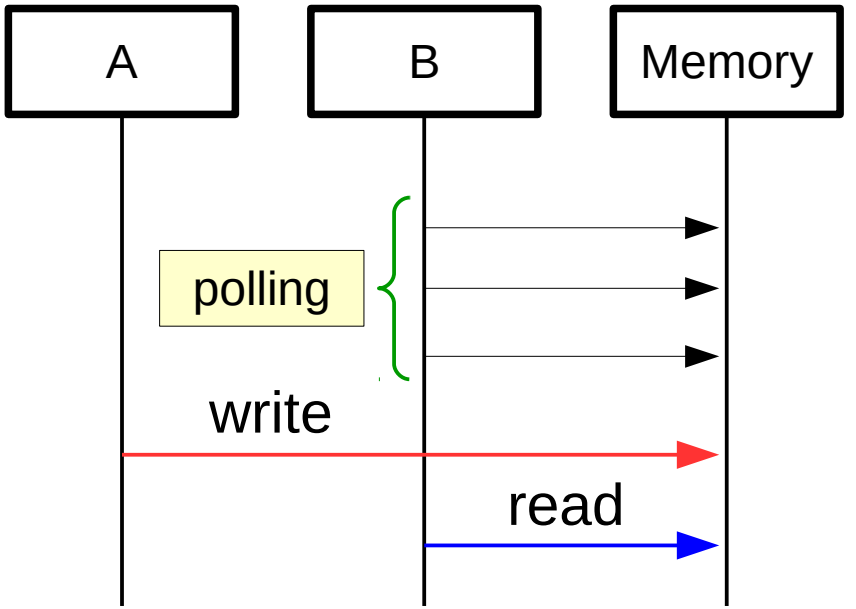


何をプリミティブと考えるか

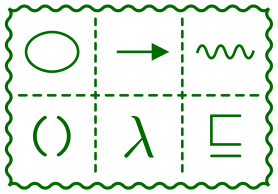
例：共有メモリ



順序をつけて排他的に処理されるものとする（メモリモデルの定義）

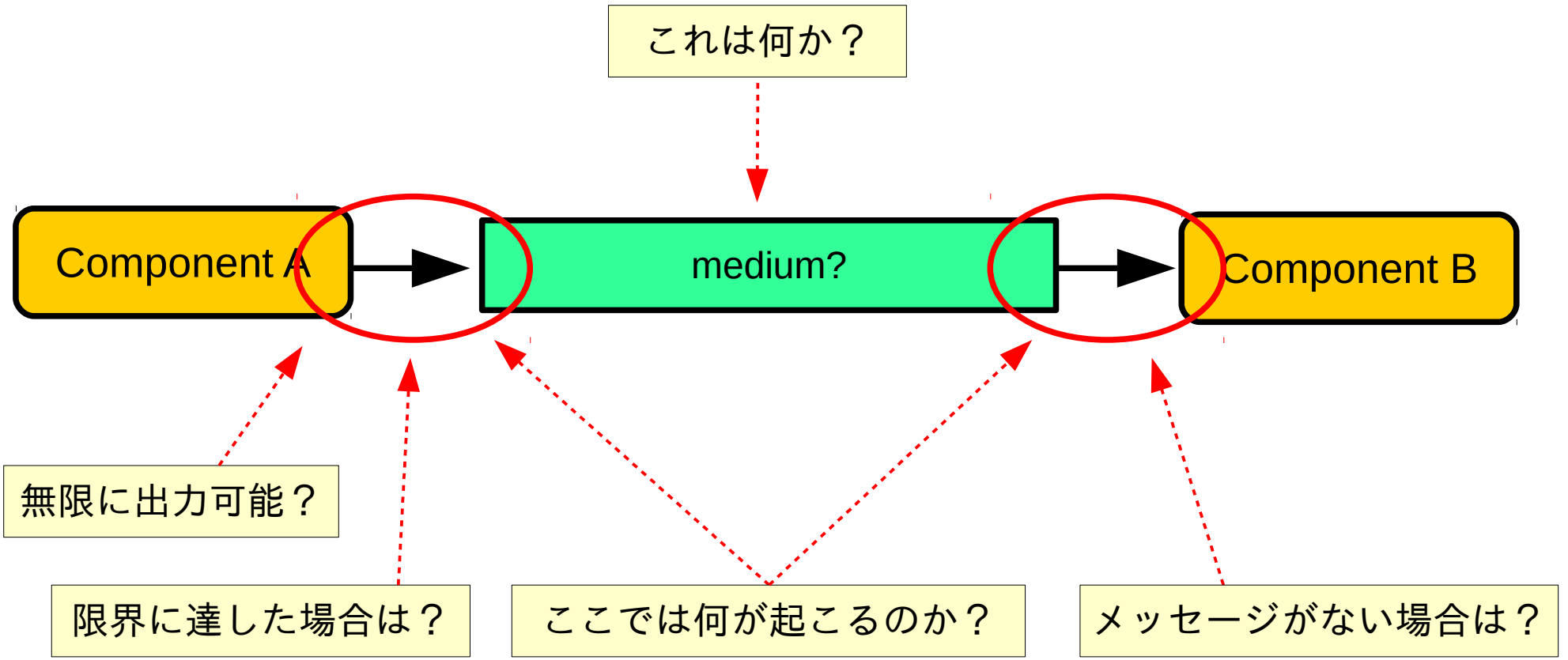


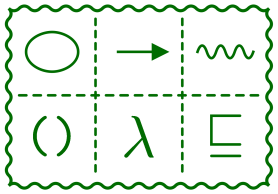
polling や busy-wait を避けるには同期機能の援用が必要



何をプリミティブと考えるか

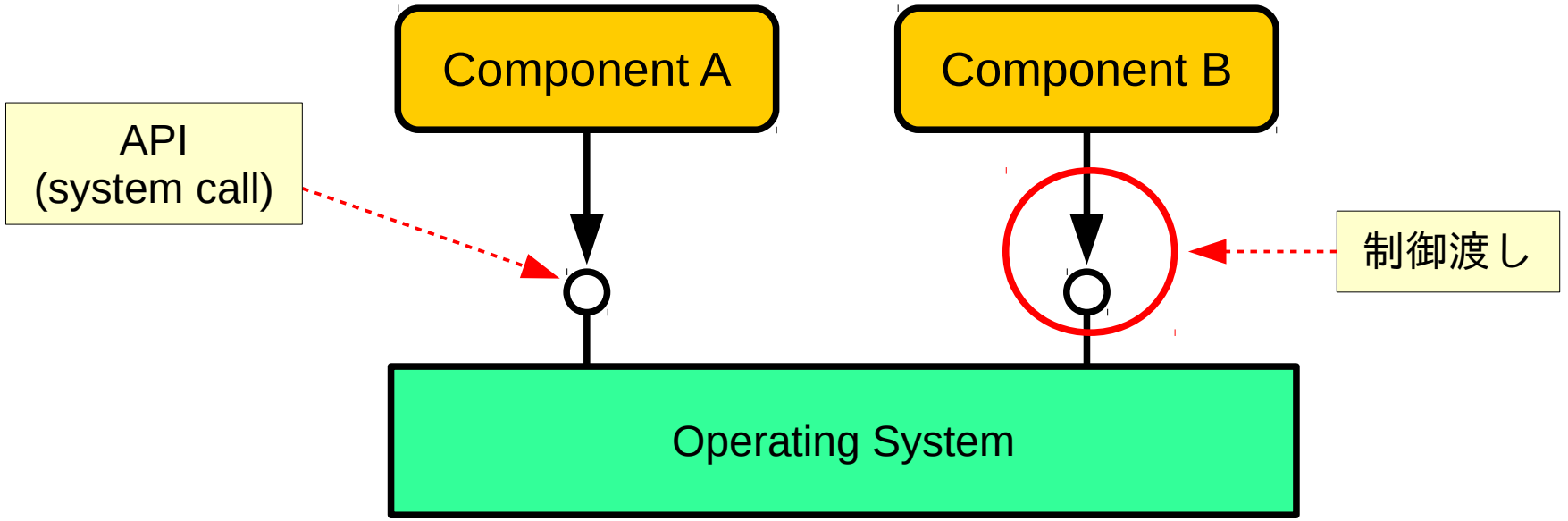
例：メッセージ通信



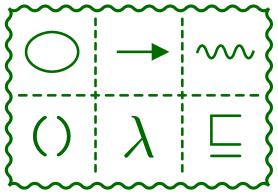


何をプリミティブと考えるか

例：同期機能

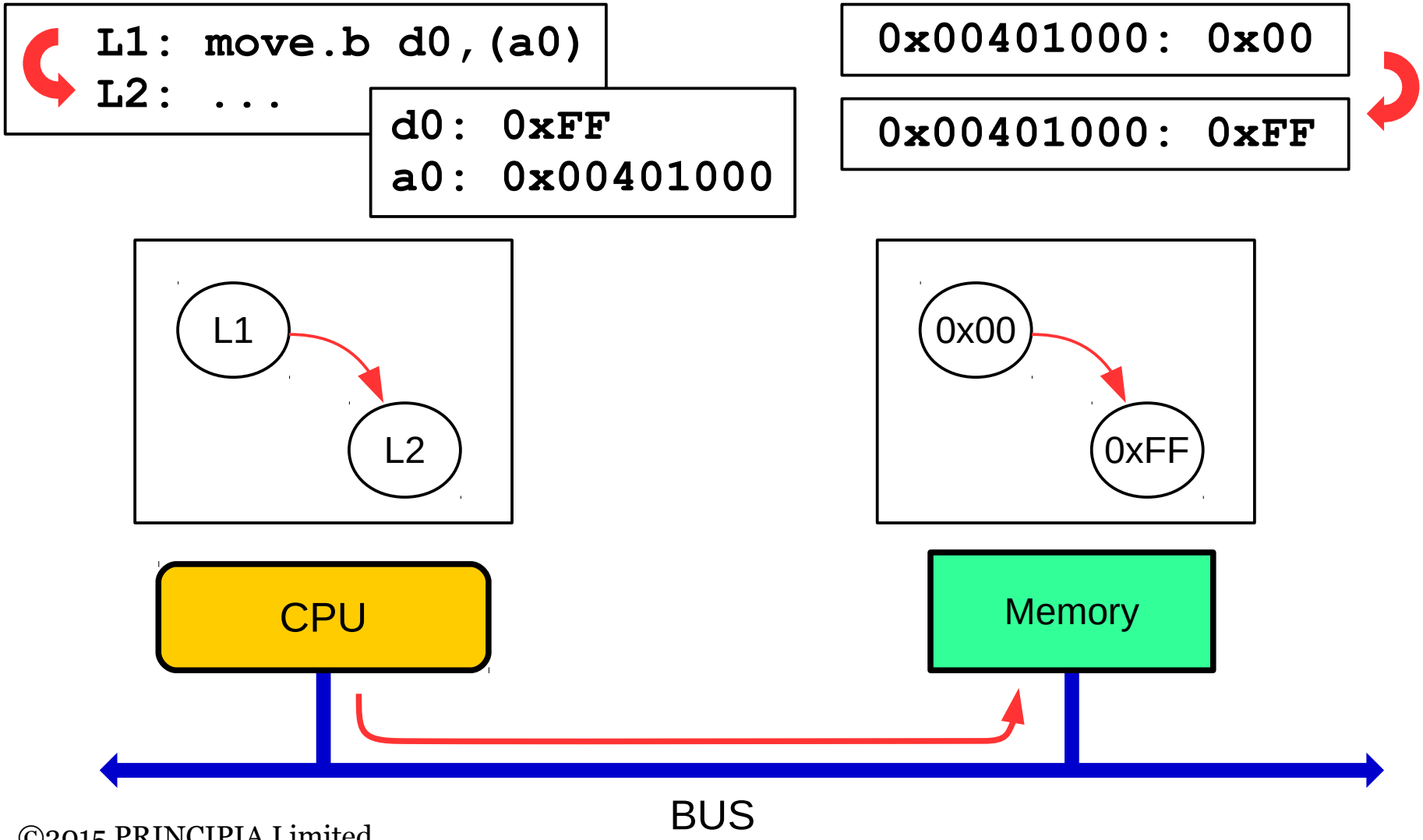


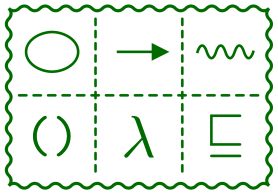
何が起るのか
例：ブロックするとはどういうことか？
どのようにモデル化するか



何をプリミティブと考えるか

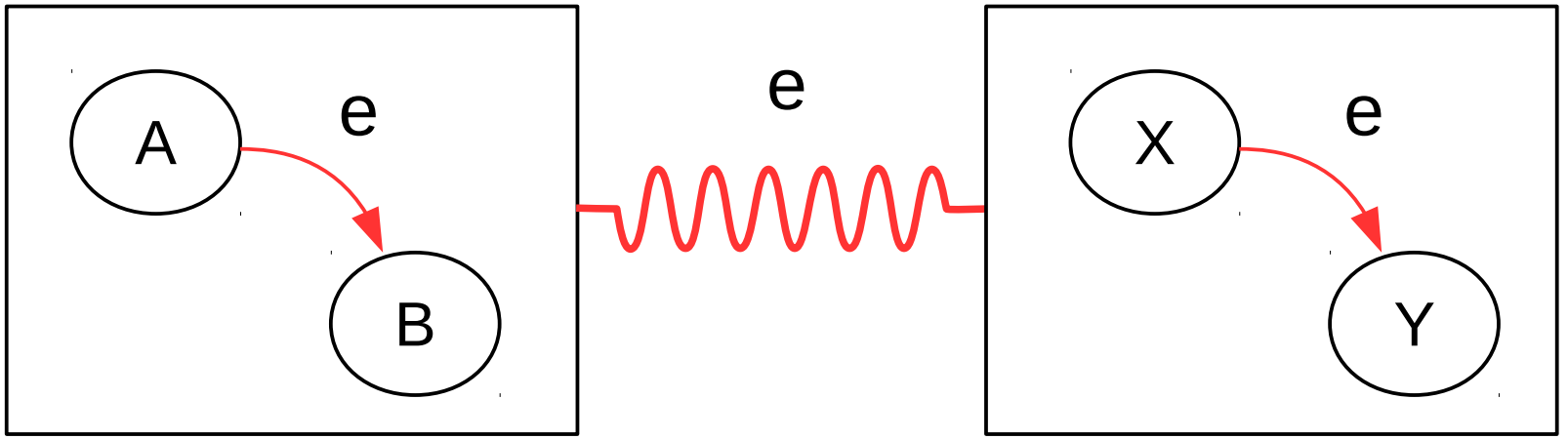
例: CPU–メモリ間相互作用





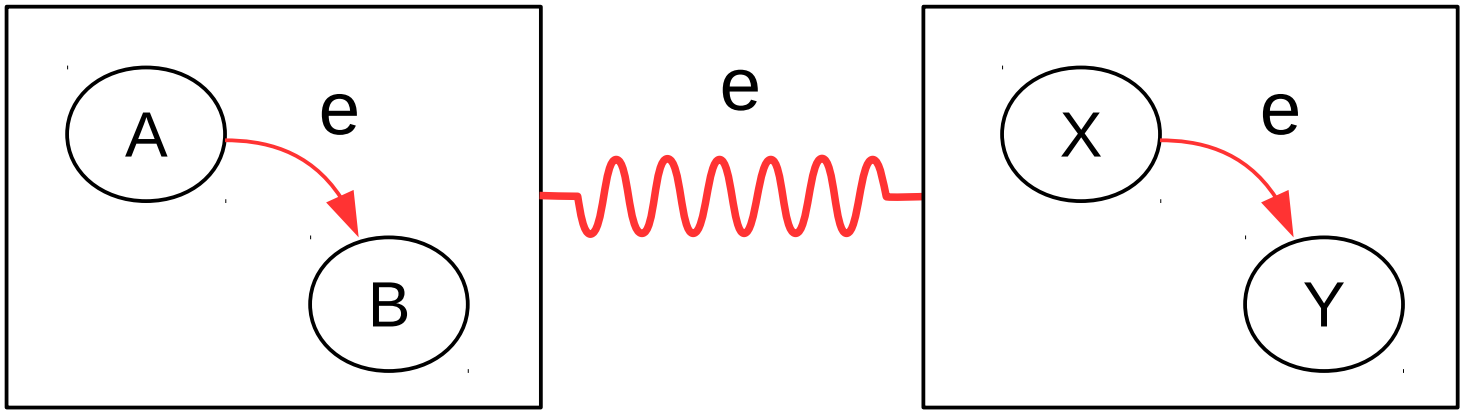
イベントによる同期型の相互作用

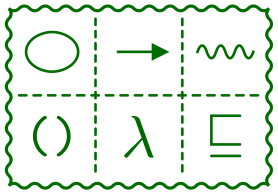
- 構成要素間で起こる相互作用の最小単位をイベントという
- イベントは名前を持ち，相互作用の種類を表す
  - 例：書き込み write，要求 req，応答 ack
  - 異なる遷移に同名のイベントをつけることができる
    - イベントは遷移の名前ではないということ



イベントによる同期型の相互作用

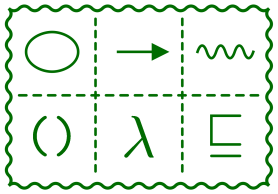
- 現状態からイベント e の付いた遷移があるとき，構成要素は「イベント e を提示している」という
- 2つの構成要素それぞれがイベント e を提示していて，対応する各遷移が同時に行われるとき，
  「イベント e が発生する」または
  「イベント e で同期する」という



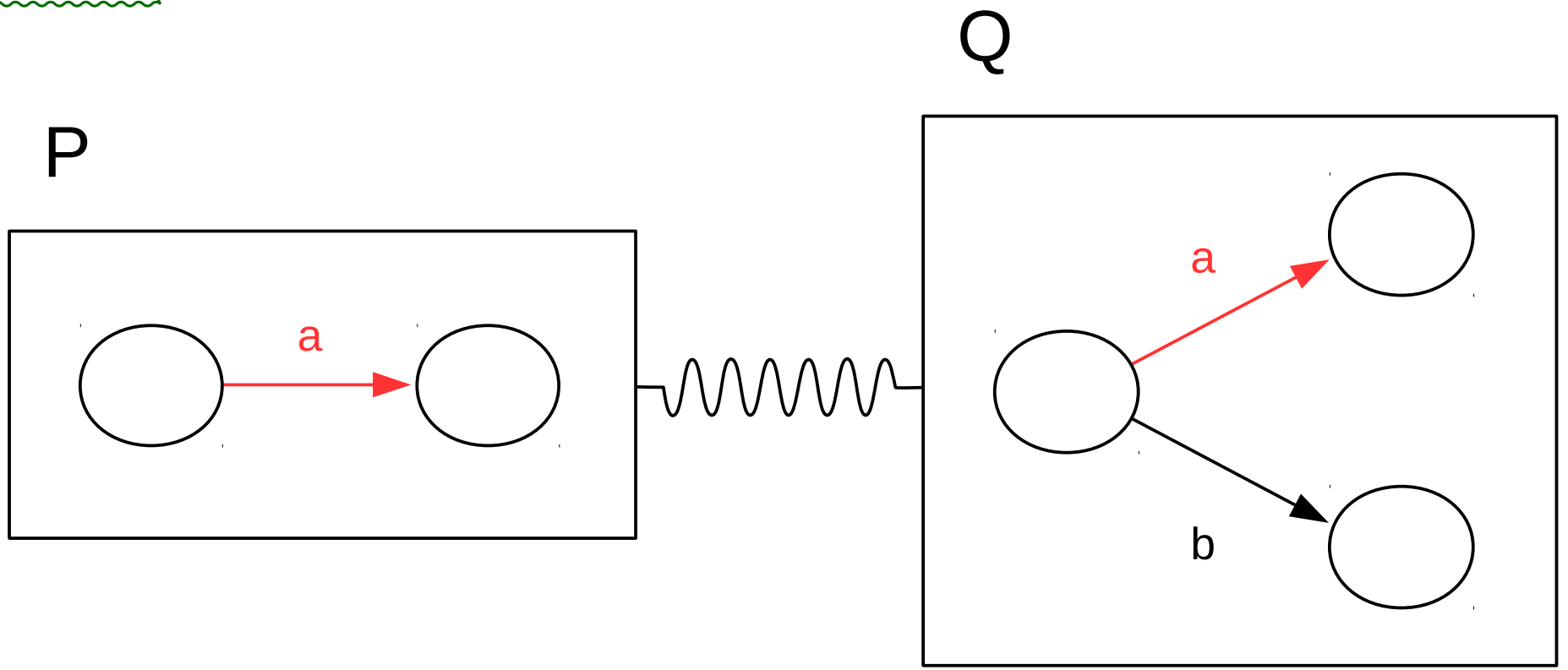


他の相互作用を表現する

- 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

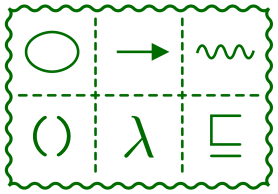


選択



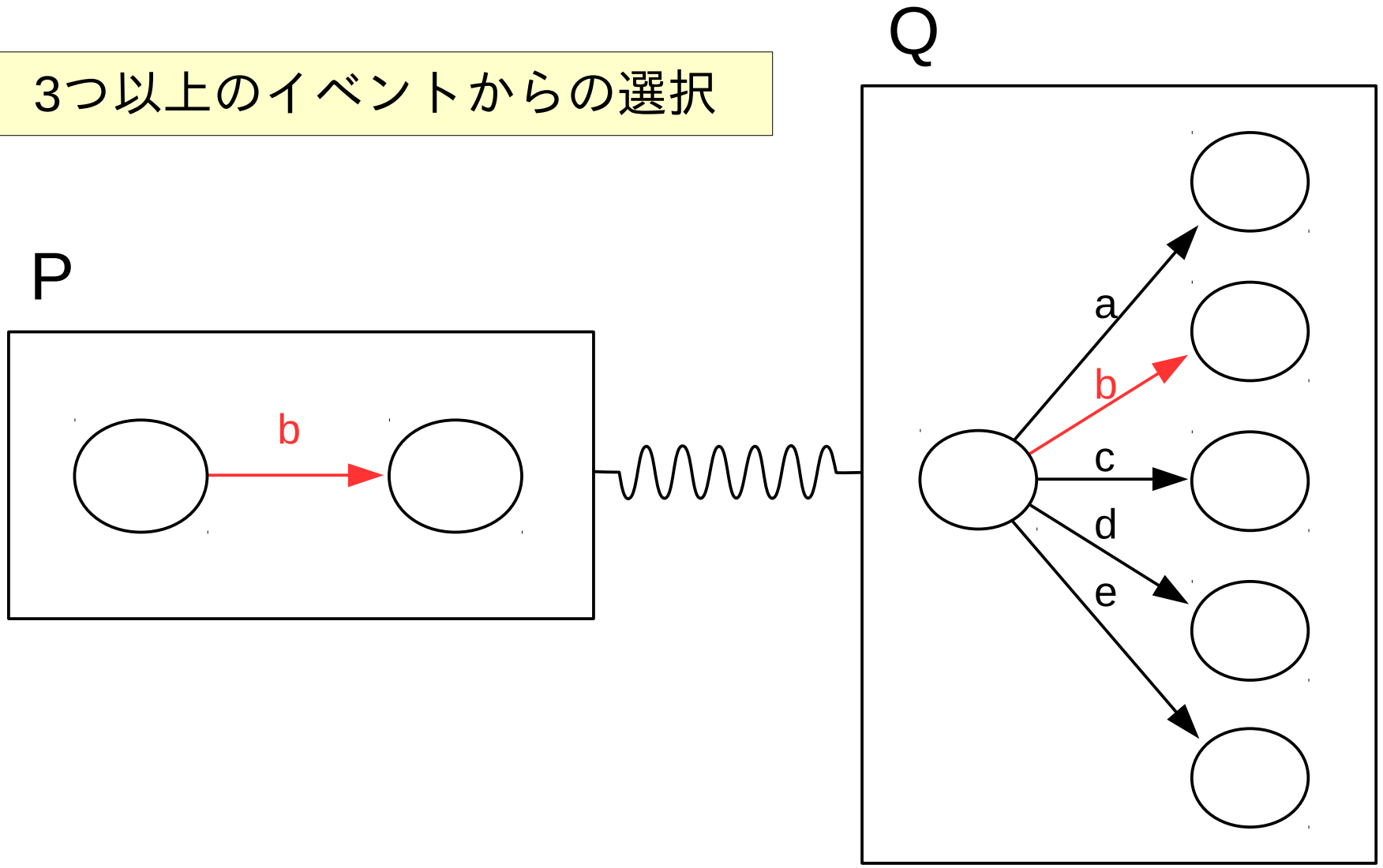
2つのイベント a, b を提示する

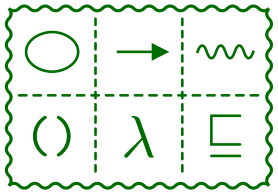
どちらが発生したかによって
相手が提示したイベントがわかる



選択

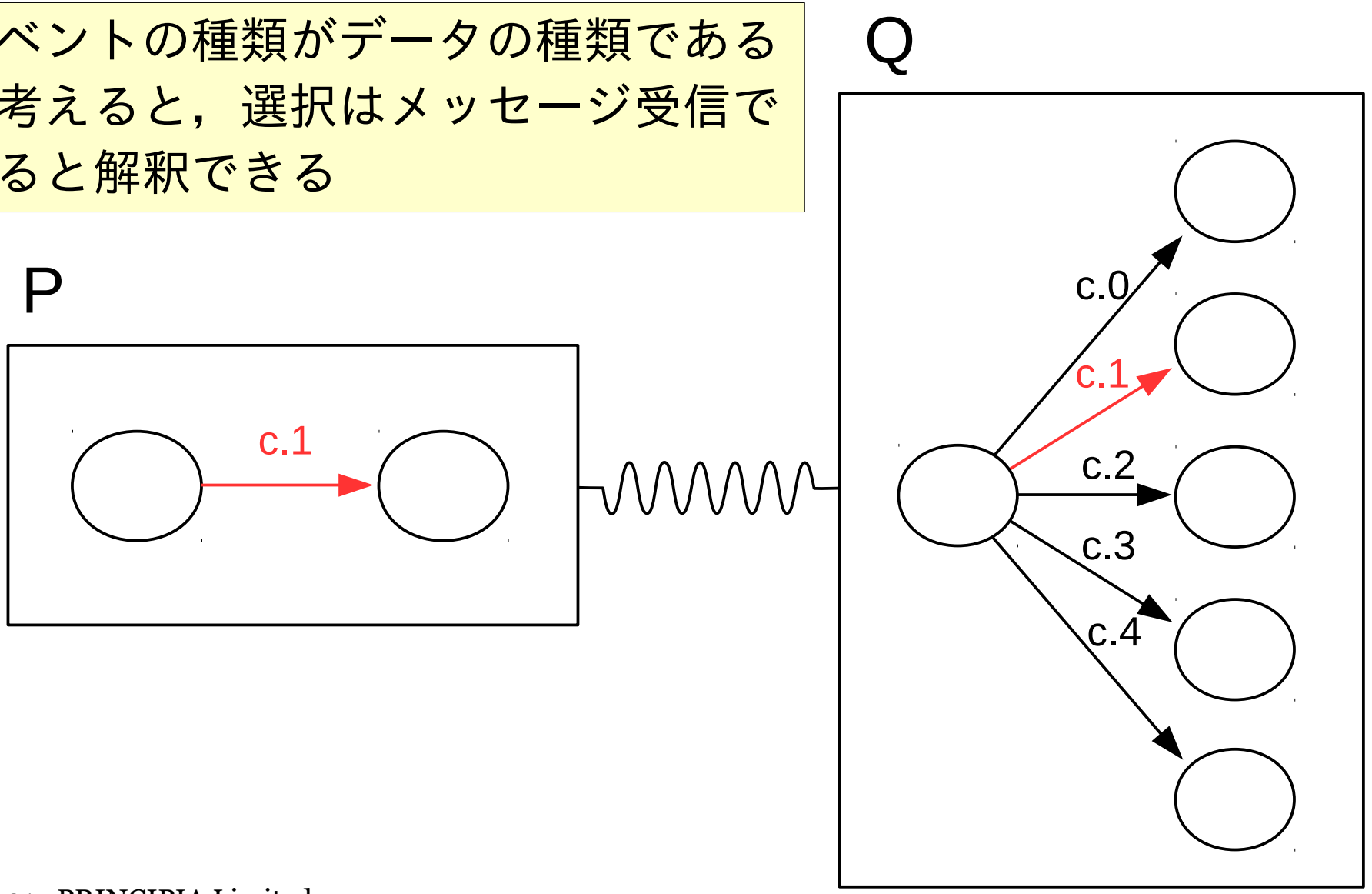
3つ以上のイベントからの選択

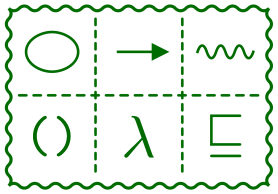




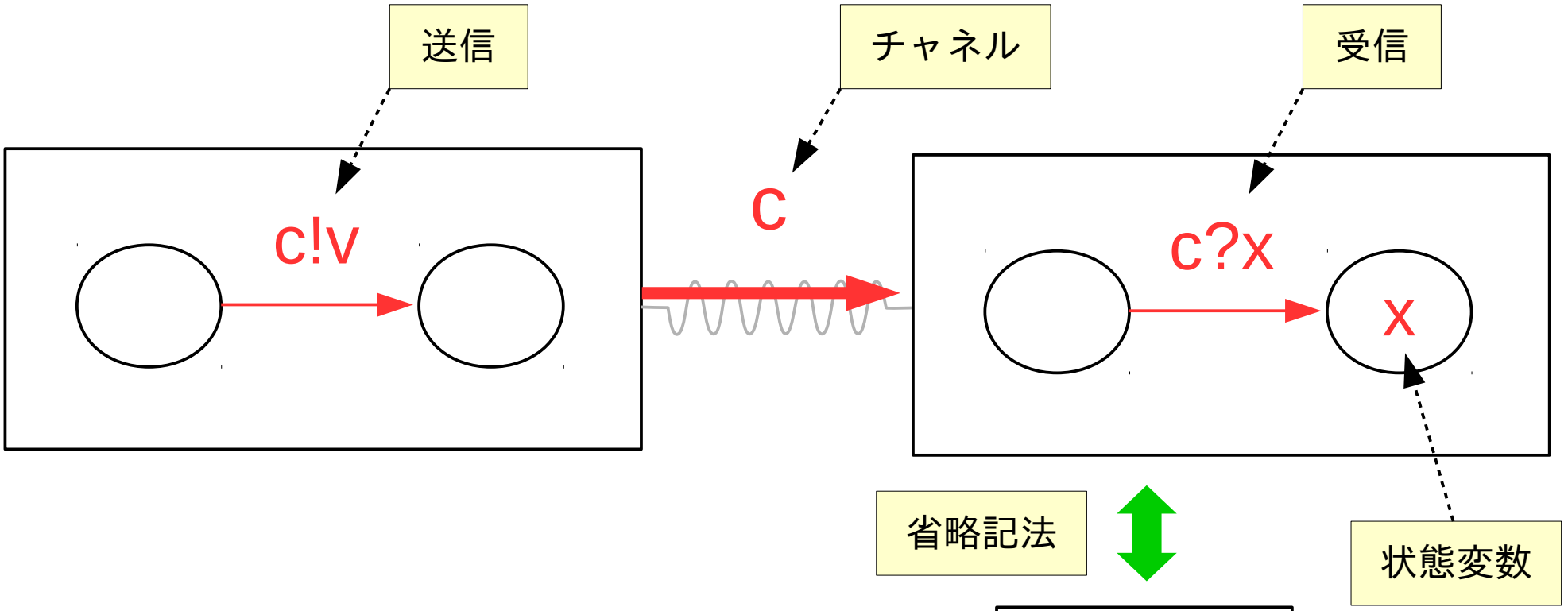
同期メッセージ通信

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

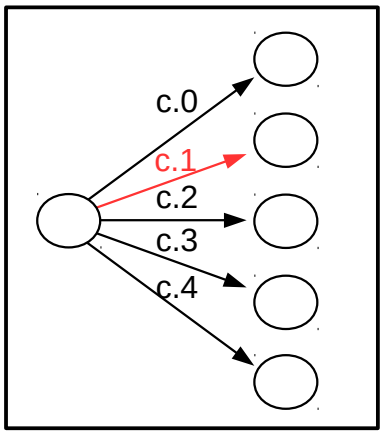




同期メッセージ通信とチャネル

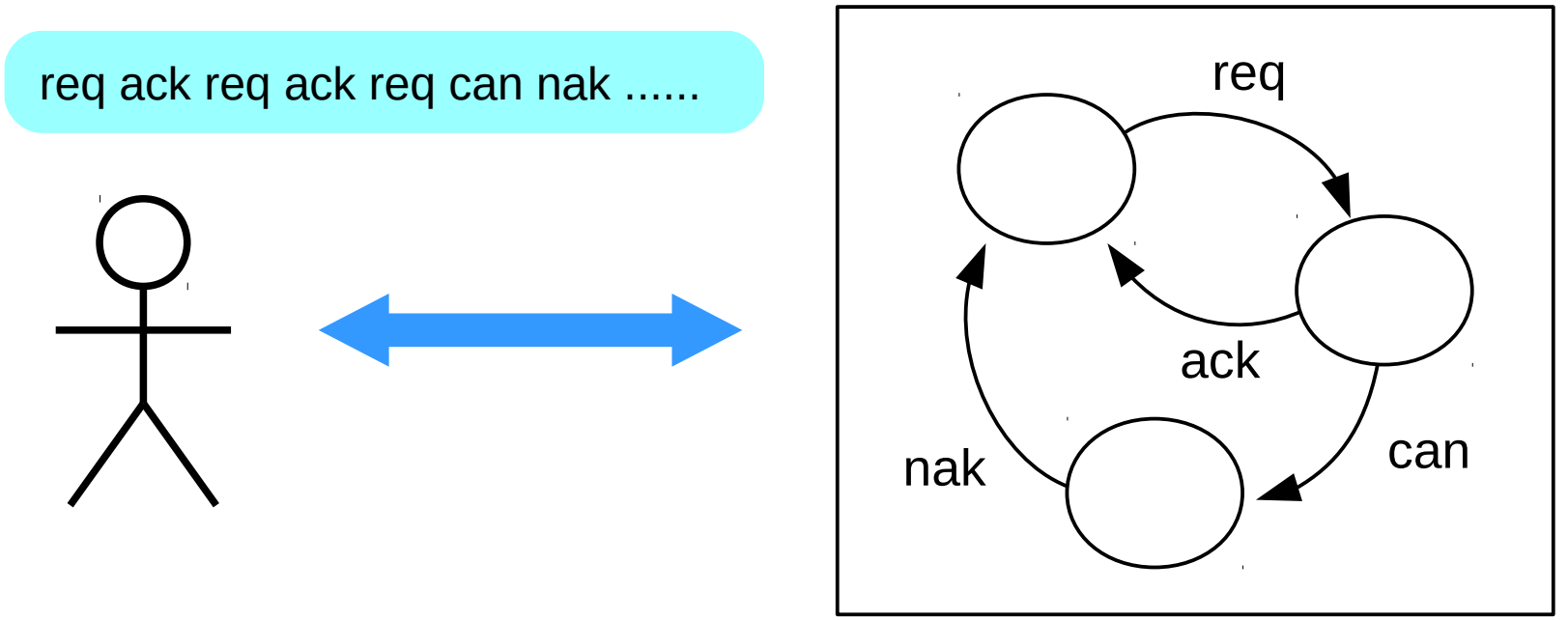


チャネル c を通じてメッセージ送受信を行っていると解釈することができる.

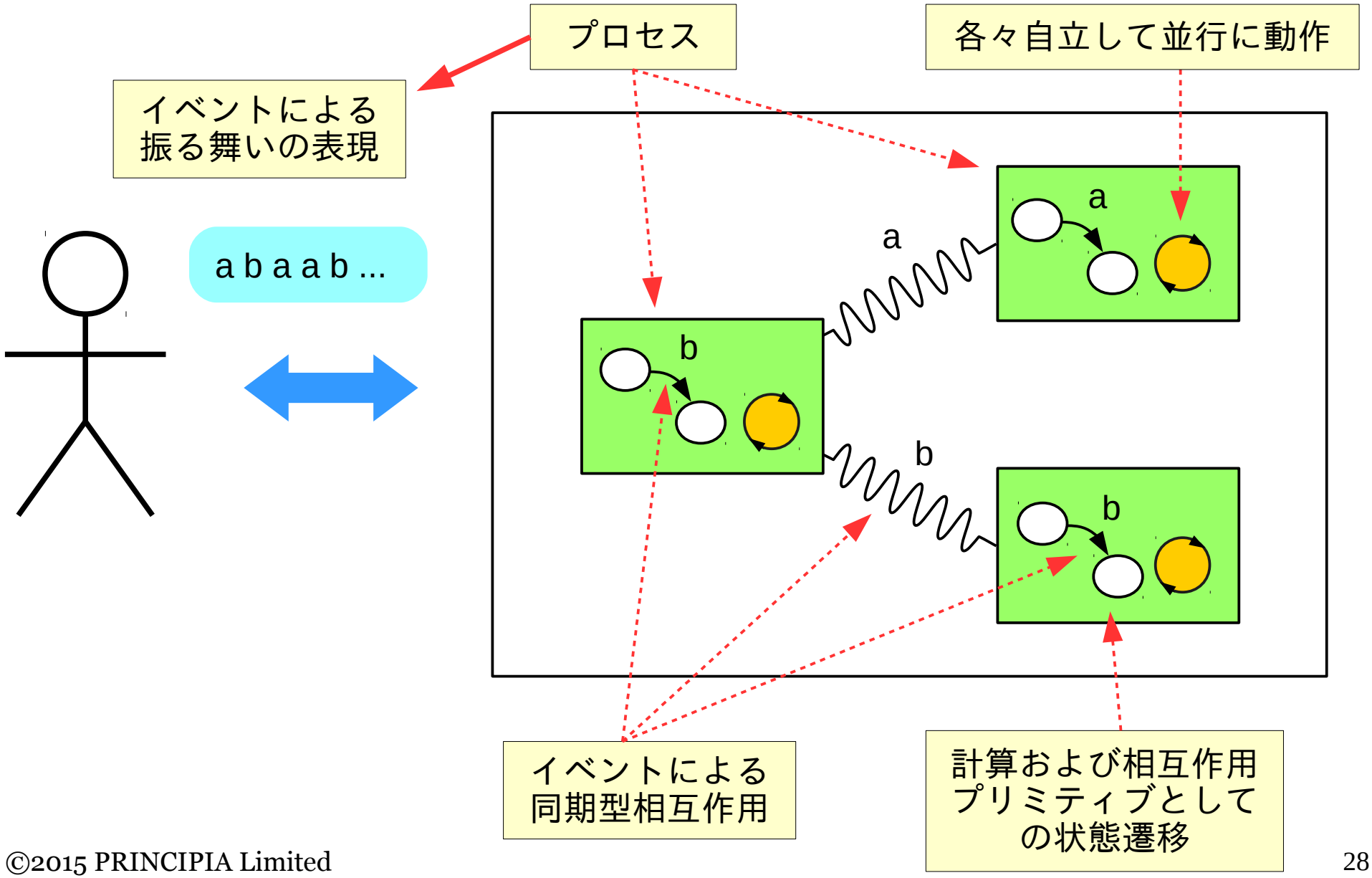


プロセス

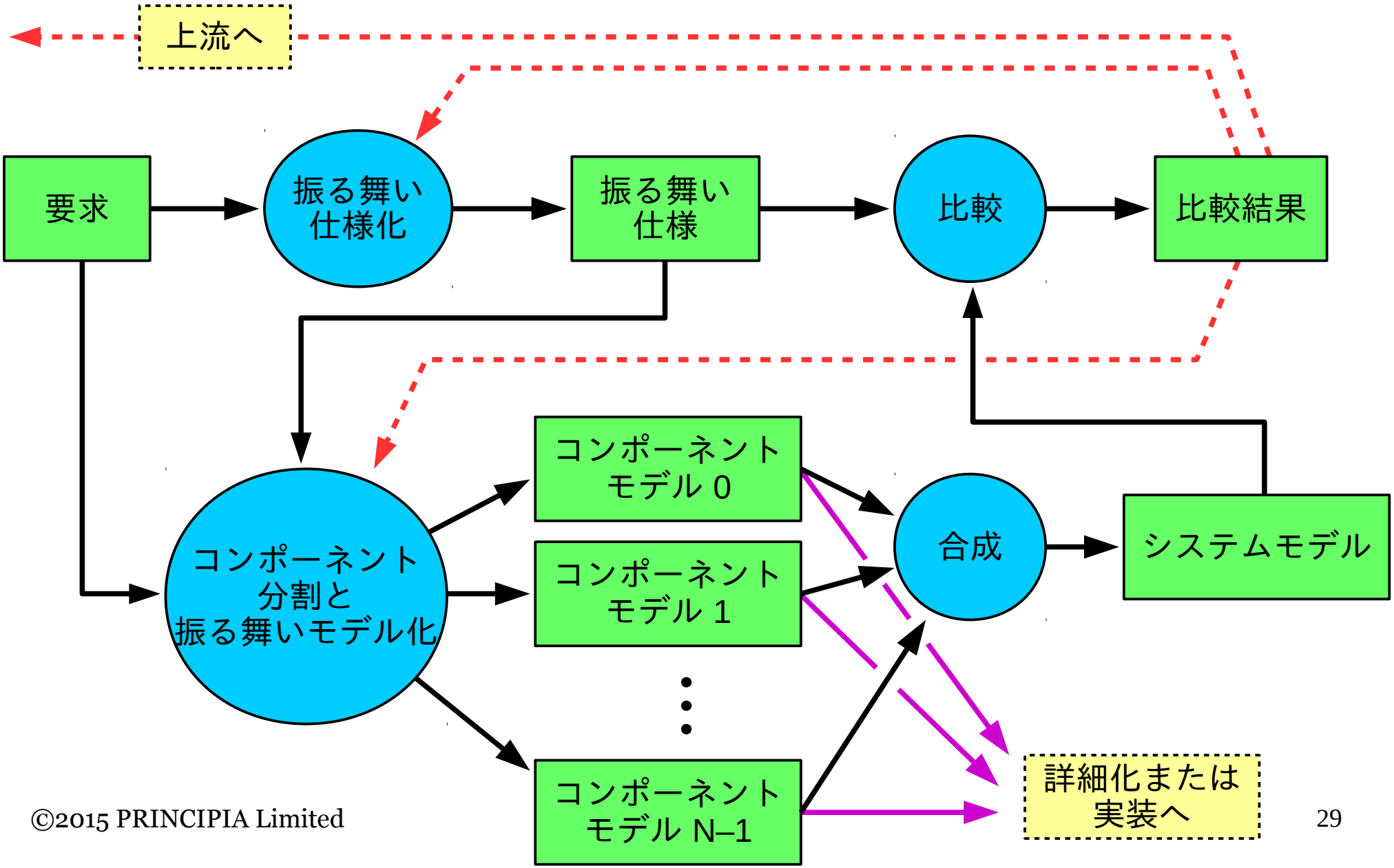
- イベントを使うと，システムや構成要素が他者と行う相互作用全体を，発生するイベントの系列として観測したり表現したりできるようになる．これは構成要素の振る舞いを，具体的な実現に依らず抽象的に表したものと考えることができる．これをプロセスと呼ぶ.
- 抽象的に表された構成要素のモデルもプロセスと呼ぶ.



並行システムのモデル

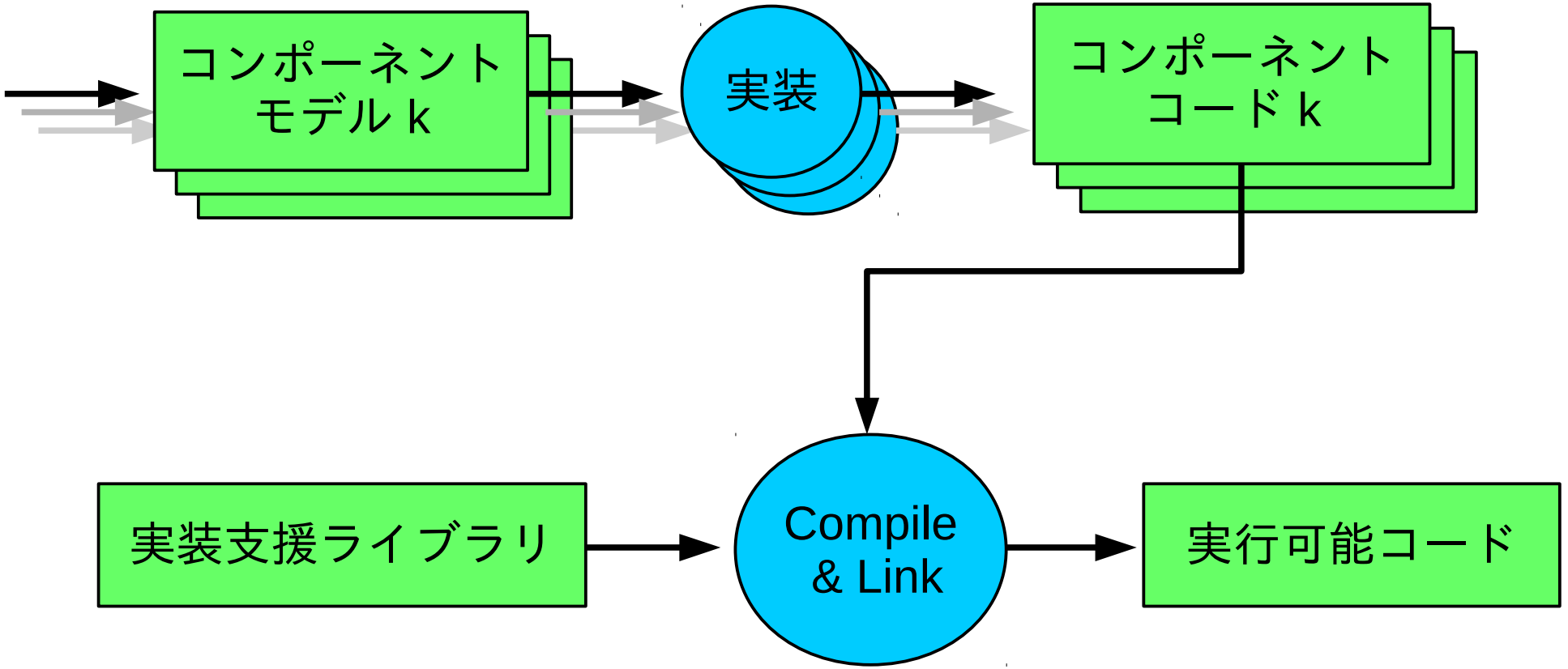


システムの設計（振る舞い側面）

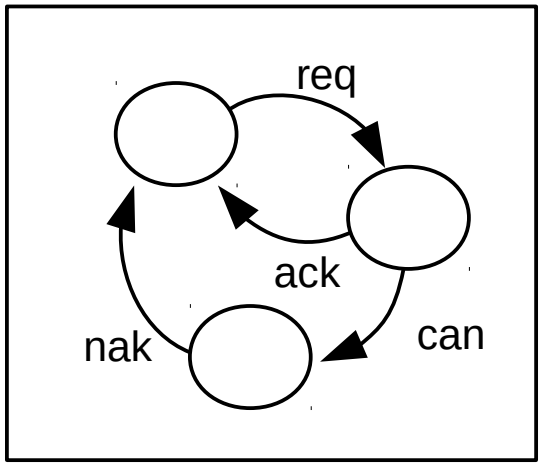


モデルから実装へ

上流から



並行システムの正当性検証



仕様

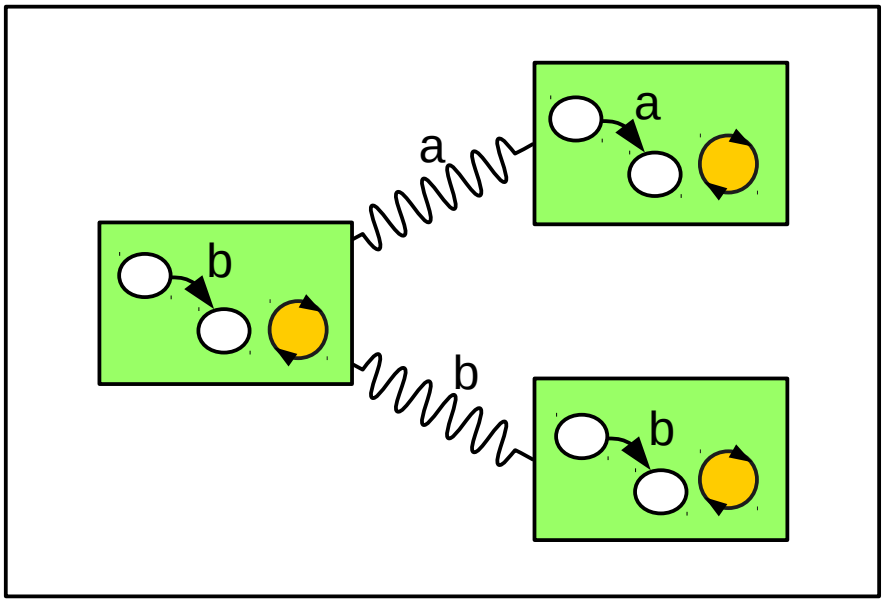
比較

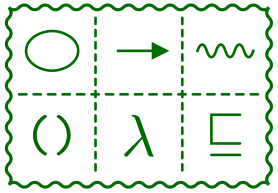
システムモデル

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



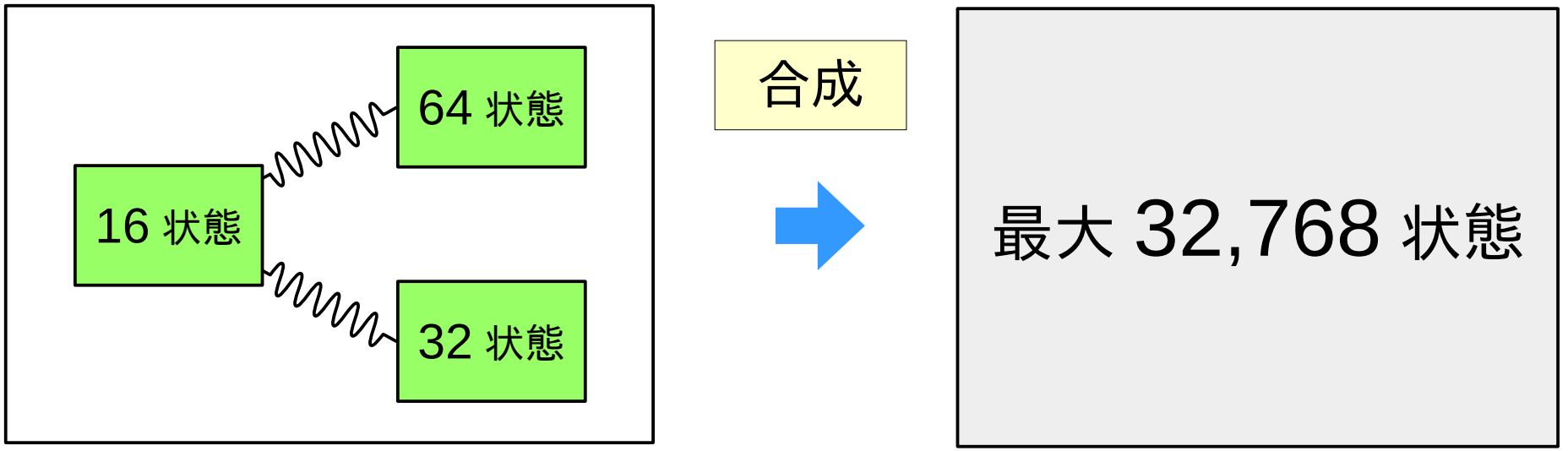


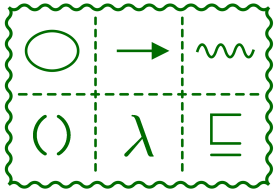
並行システム開発の課題

- 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



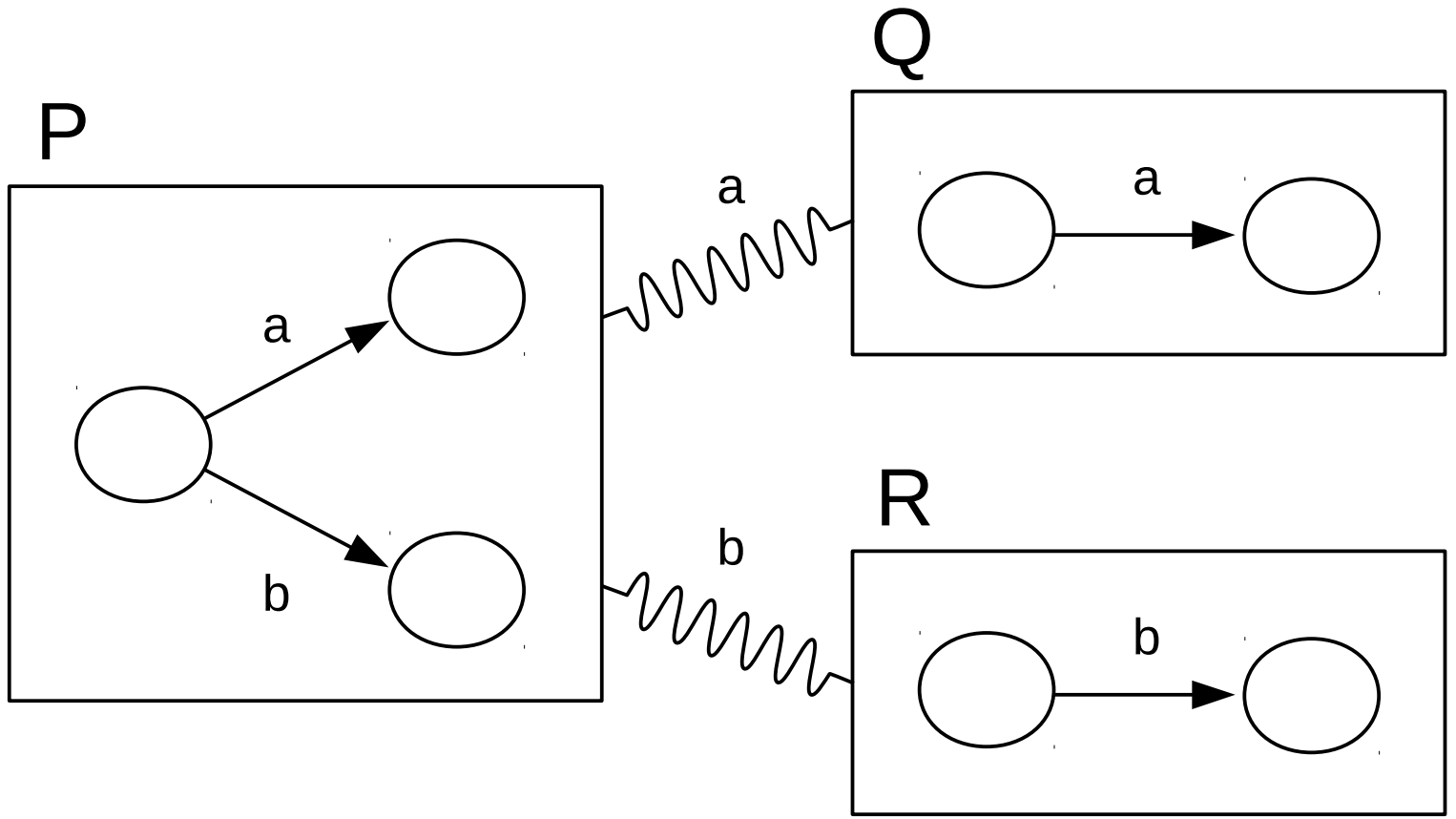


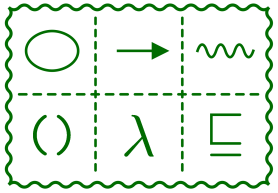
状態数の組み合わせ爆発

- 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



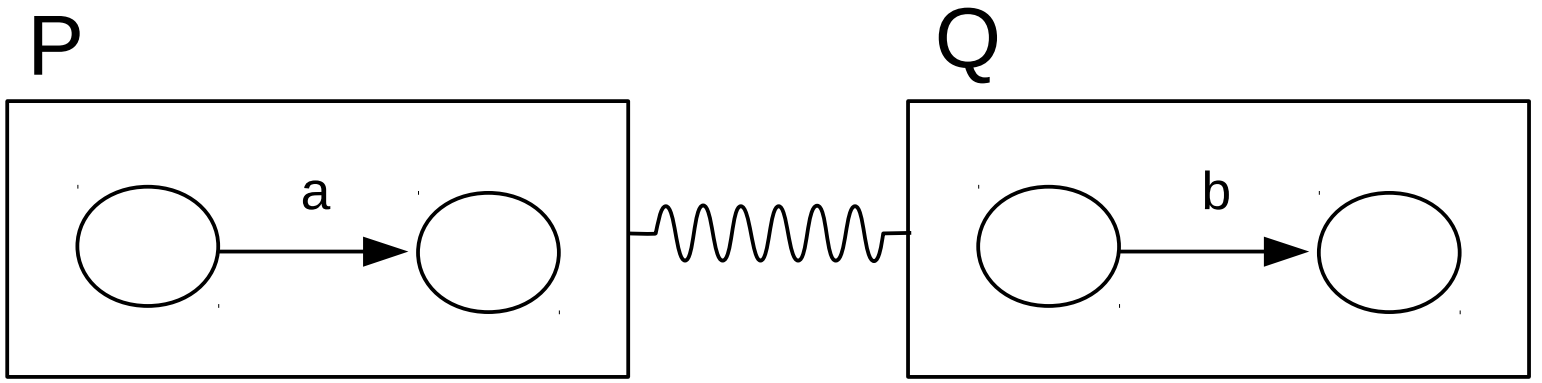


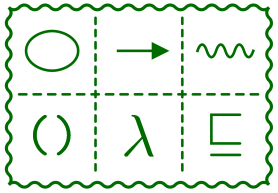
非決定性

- 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



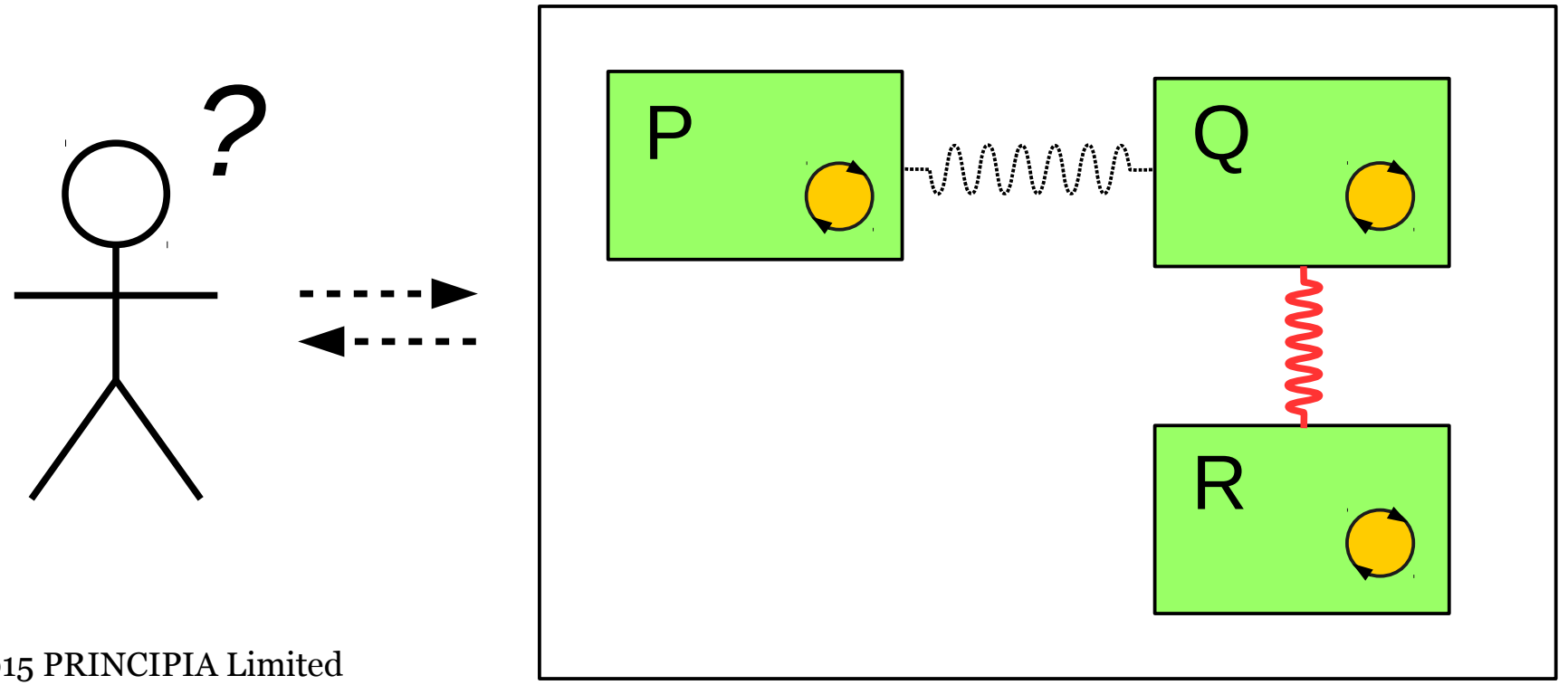


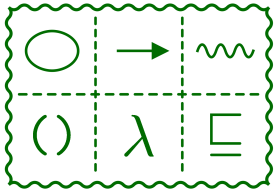
デッドロック

- 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）

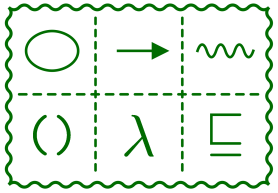
- 一部のコンポーネントが進捗のない相互作用を繰り返していることにより，システムが機能できなくなる状態





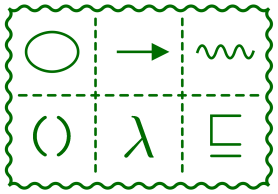
ライブロック（発散）

- 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
- イベントによる同期型相互作用
- プロセス: イベントによる振る舞いの表現
- 並行システムの設計
  - コンポーネント分割と振る舞いのモデル化
  - プロセスの合成
  - 比較による正当性検証
- 並行システム開発における課題



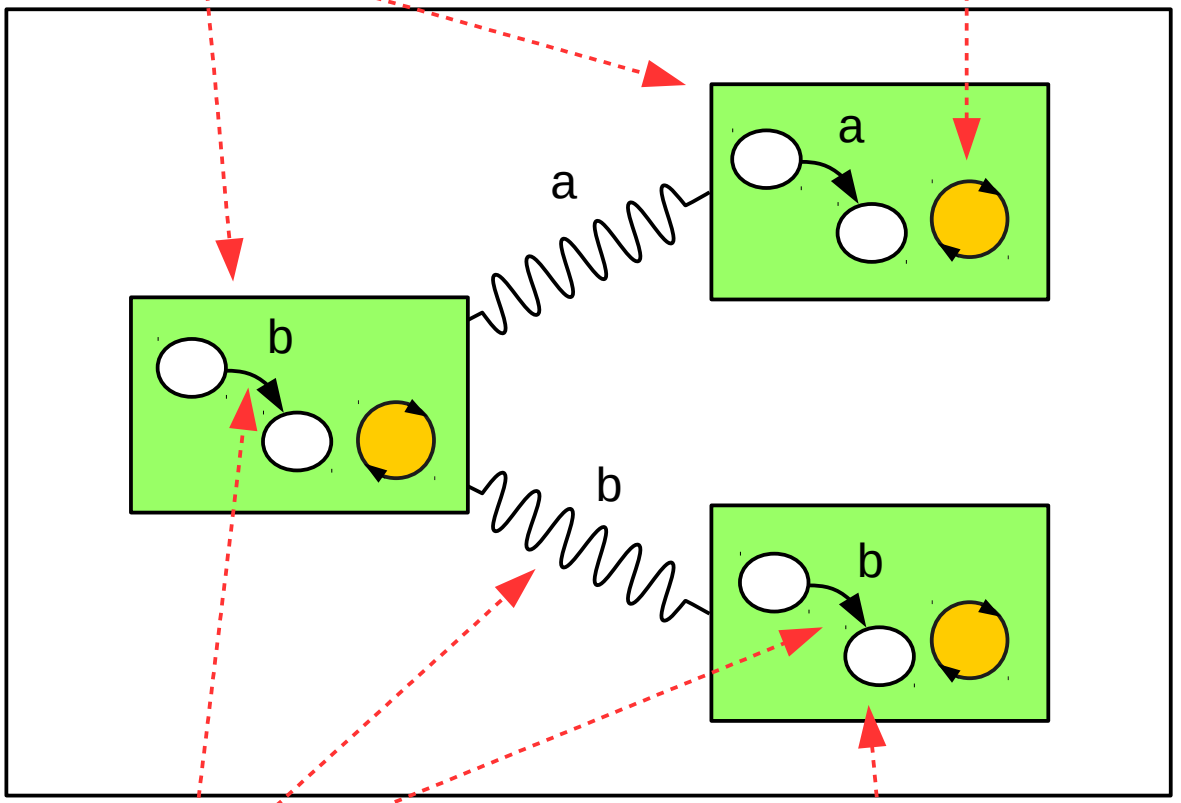
並行システムのモデル

プロセス

各々自立して並行に動作

イベントによる
振る舞いの表現

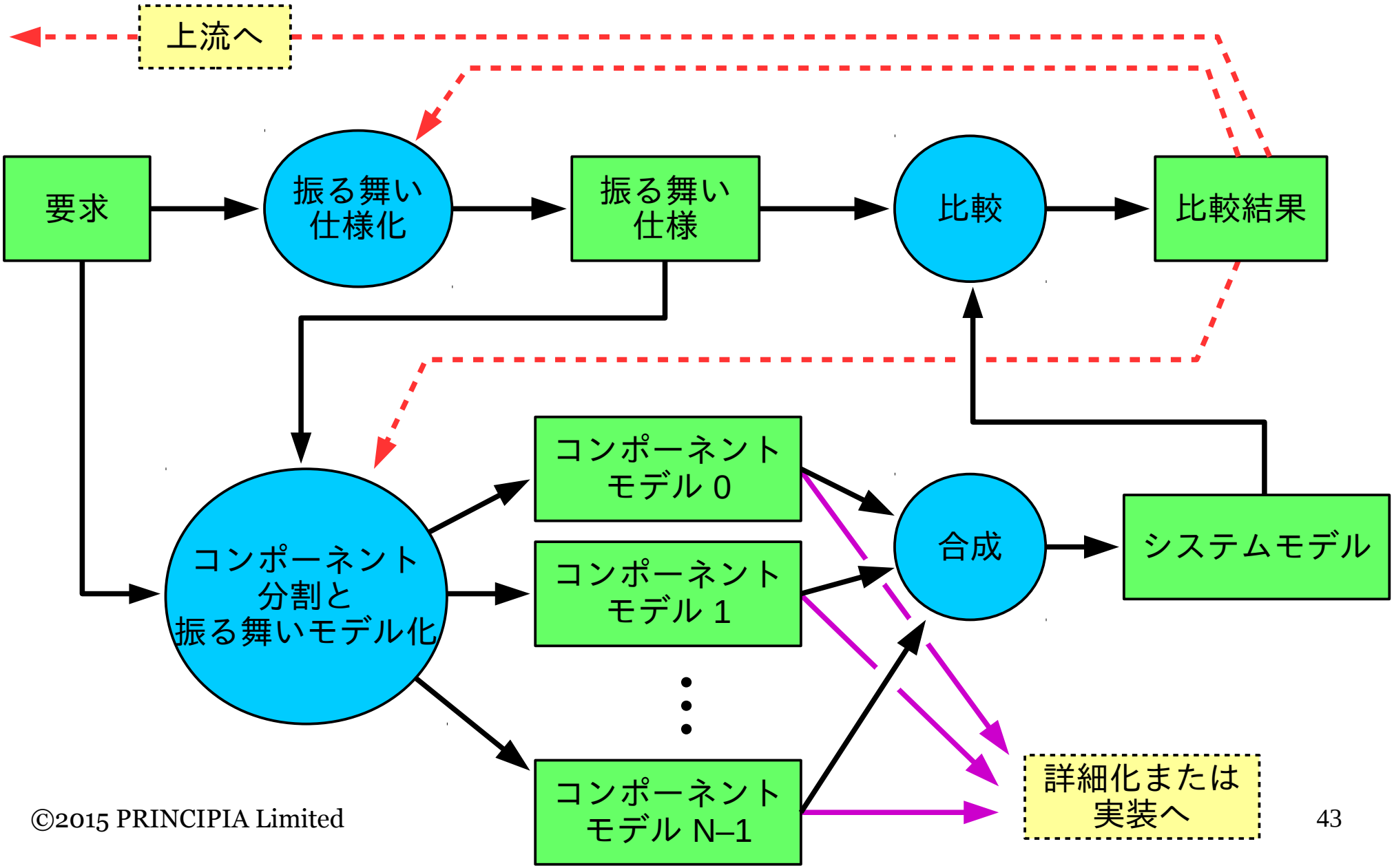
a b a a b ...



イベントによる
同期型相互作用

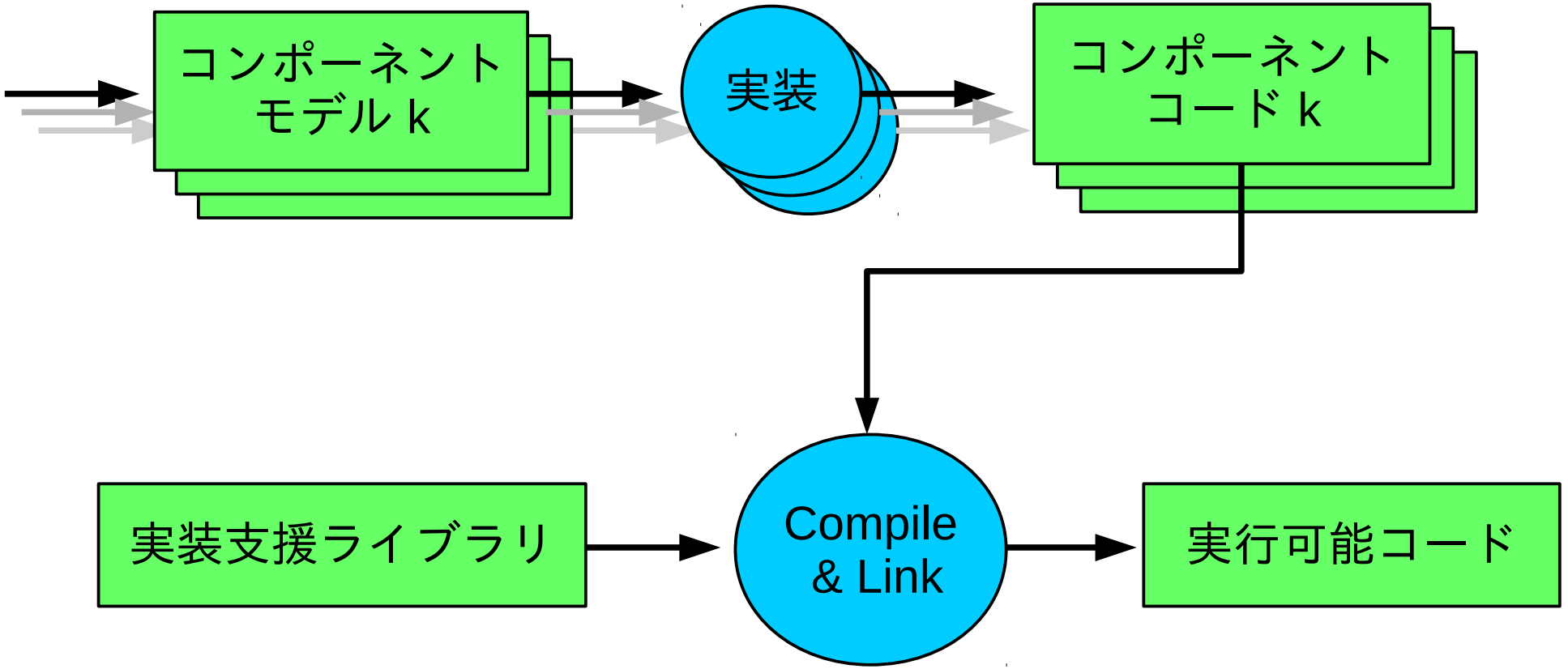
計算および相互作用
プリミティブとして
の状態遷移

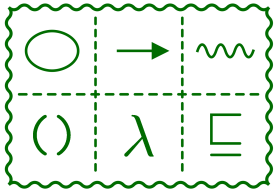
システムの設計（振る舞い側面）



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）