

テストと検証シリーズご紹介

担当講師:

- 石川 冬樹 (NII)
- 宇佐美 雅紀 (Amazon Jpn)
- 桑野 文洋 (日本工業大学)
- 久連石 圭 (東芝)
- 田辺 良則 (鶴見大学)
- 長谷川哲夫 (東芝)
- 初谷 久史 (PRINCIPIA)
- 早水 公二 (フォーマルテック)
- 藤本 洋 (イーソル)
- 前田 直人 (NEC)
- 松崎 和賢 (中央大学)
- 吉岡 信和 (早稲田大学)



テストと検証シリーズ

ソフトウェアの品質確保に関する手法

- テスティング
 - テスティング（基礎）
- 静的解析
 - プログラム解析
- モデル検査
 - モデル検査入門I, II
 - 設計モデル検証
 - モデル検査事例演習
 - 並行システムの設計と検証
 - 性能モデル検証



テストティング (基礎)

- テストツール
- テスト駆動開発
- ブラックボックステスト
 - 直交表
- ホワイトボックステスト
 - パスカバレッジ
- テスト自動化
- テスト計画
- テストの管理と評価・改善

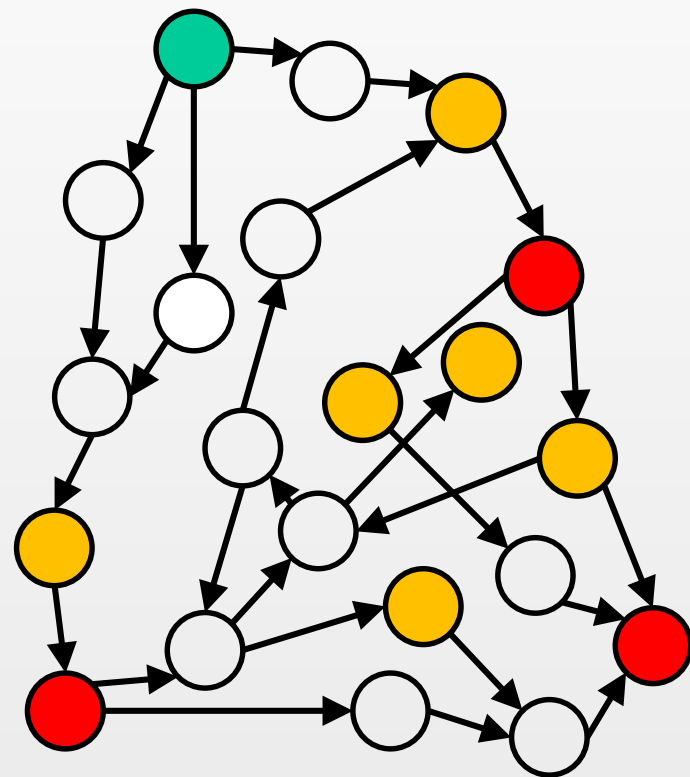


プログラム解析

- オムニバス形式で, いろいろな静的解析の手法・ツールを紹介する.
 - 有界モデル検査 / CBMC
 - データフロー解析 / SOBA
 - 記号実行 / Triton
 - 契約による設計 / JML

モデル検査技術

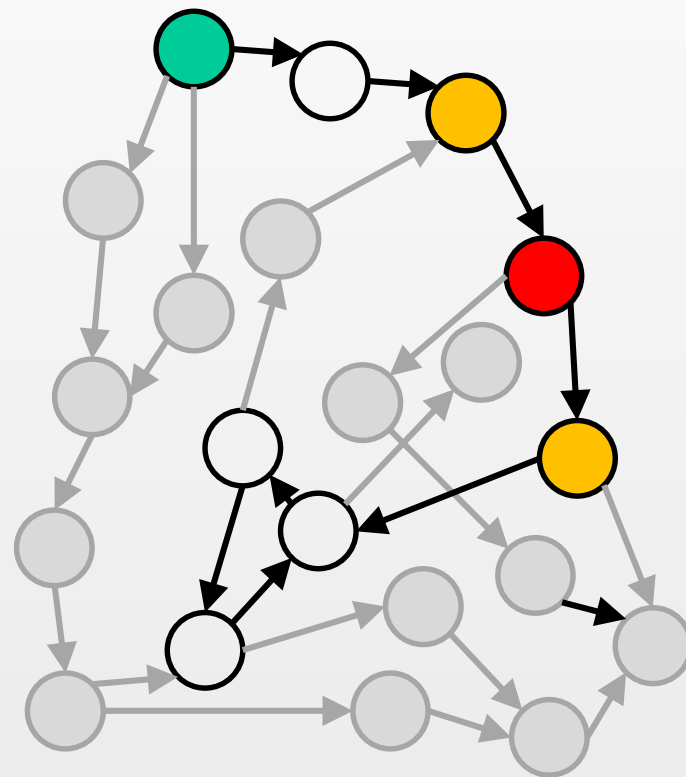
- システムが取り得るあらゆる状態を含む状態遷移系を構築する.
- 状態遷移系を効率よく探索して, 不具合を起こす経路の有無を判断.



緑: スタート, 黄色: オープン, 赤: クローズ
オープンしたら必ずクローズされるか?

モデル検査技術

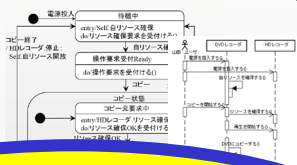
- システムが取り得るあらゆる状態を含む状態遷移系を構築する.
- 状態遷移系を効率よく探索して, 不具合を起こす経路の有無を判断.



緑: スタート, 黄色: オープン, 赤: クローズ
オープンしたら必ずクローズされるか?

モデル検査のアプローチ

仕様書



仕様書だけ?
ソースコードは?

成り立つべき性質

オープン

性質って
どう書くの?

他の開発工程
との関係は?

または

モデル検査が
有効な領域は?

状態遷移系 (数学的モデル)

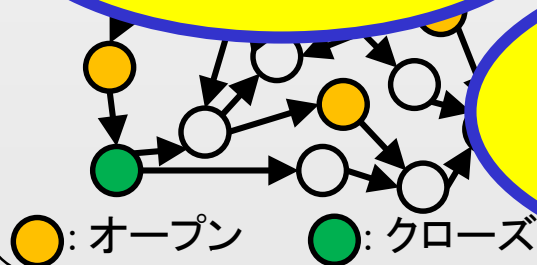
遷移系の
作り方は?

時間制約も
扱える?

モデル

モデル検査器の
使い方は?

検査結果を説
明してわかって
もらえるかな?





性質って
どう書くの？

遷移系の
作り方は？

モデル検査器
の使い方は？

モデル検査入門Ⅰ, Ⅱ

■ モデル検査の基礎理論

■ 検証する性質の表現 = 時相論理

- Linear Temporal Logic

- Computational Tree Logic

■ 状態遷移系

■ 代表的なモデル検査器に習熟

- SMV - 状態遷移系を直接記述

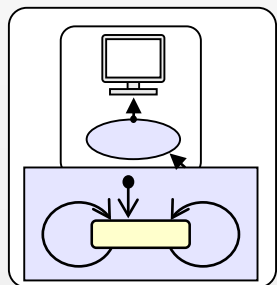
- SPIN - プロセスをプログラミング言語風に記述

■ ロボットを利用した実習

モデル検査が有効な領域は？

並行システムの設計検証

逐次システムや仕様



モデル化

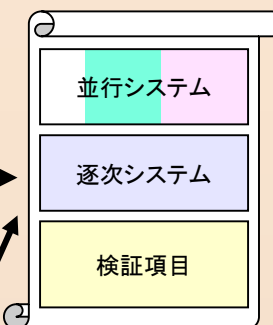
プロセス代数CSP
によるモデル化

CSPモデル

```
SRseq(K) = SendRec(0,0,0,K)
SendRec(n,m,i,K)
= i < K & disp!send.n
  → SendRec((n+1)%N,m,i+1,K) □
  i > 0 & disp!receive.m
  → SendRec(n,(m+1)%N,i-1,K)
```

検証

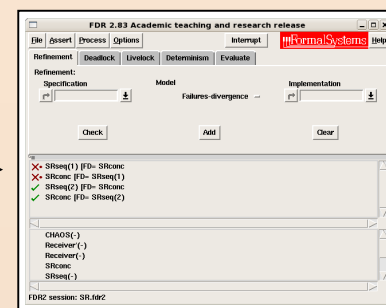
FDRスクリプト



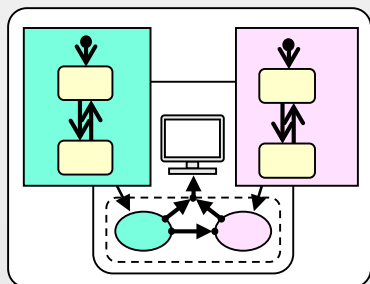
実行

モデル検査器
FDRによる検証

FDR



並行システムや仕様



モデル化

CSPモデル

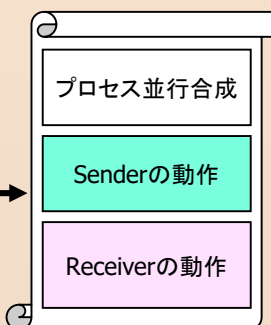
```
SRconc
= (Sender(0) [] { | chan | } []
  Receiver(0)) \ { | chan | }

Sender(n)
= disp!send.n → Sender'(n)
Sender'(n)
= chan!n → Sender((n+1)%N)

Receiver(m)
= chan?m → Receiver'(m)
Receiver'(m)
= disp!receive.m → Receiver(m)
```

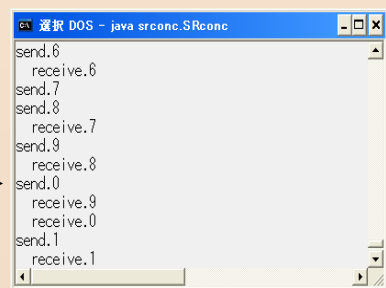
実装

JCSPプログラム



実行

Java



Javaライブラリ
JCSPによる実装

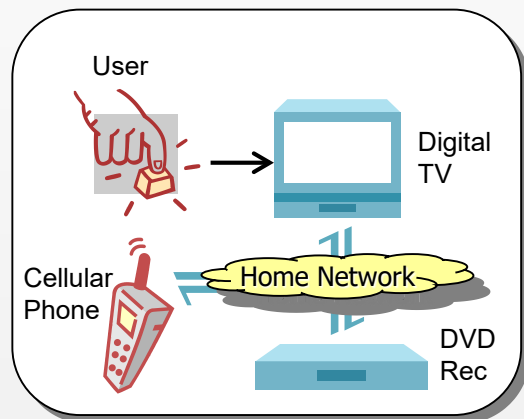
他の開発工程との関係は?

設計モデル検証

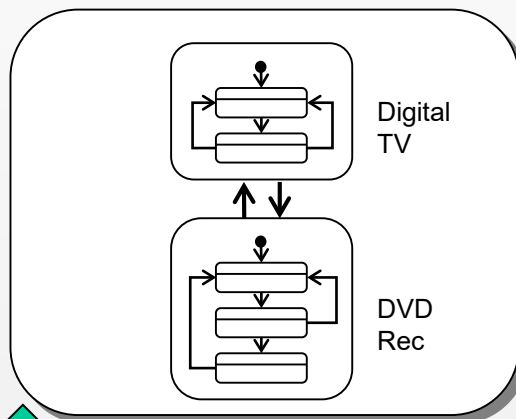
要求仕様

設計モデル

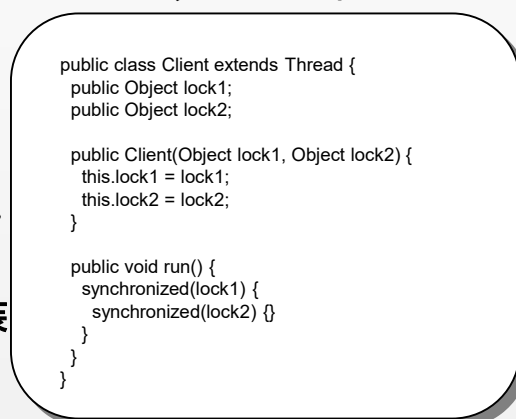
ソースコード



設計



実装

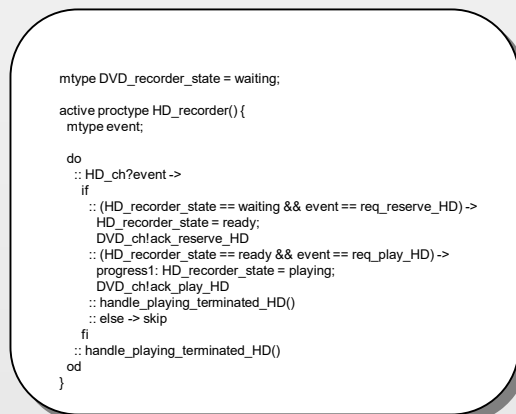


検証要求分析
検証モデル設計

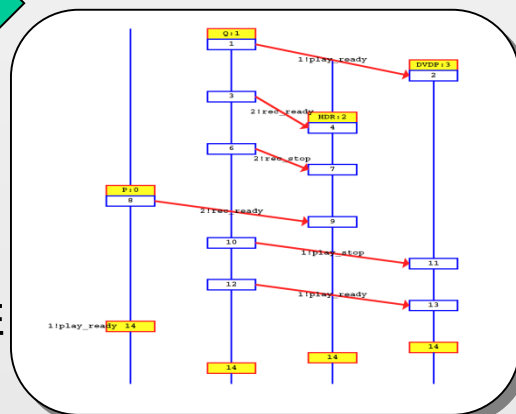
設計誤り発見

経験者の
頭の中

実装



検証



検証モデル

モデル検査記述

検証結果／反例



時間制約も
扱える?

性能モデル検証

- 組み込みソフトウェアの**時間制約**に関する設計検証
- 講座で取り扱う課題
 - いろいろな時間要素の検証モデルへの反映方法と検証結果分析の難しさ
 - 設計プロセスと検証プロセスのシームレスな連携
- 利用するツール
 - 時間制約を扱えるモデル検査ツール**UPPAAL**
 - スウェーデンのUppsala大学とデンマークのAalborg大学により開発



モデル検査事例演習

【目的】企業におけるモデル検査推進者の育成

- モデル検査の実務(組織)を想定し, その開始から終了までの全プロセスを体験する演習

【習得できる知識】

- 組織におけるモデル検査の**適用プロセス**

1. 開発者からのヒアリング
2. ソフトウェアのモデル化
3. モデル検査
4. 結果報告
5. 最終報告書

- モデル検査**報告書の作成**方法

- モデル検査の導入を推進するための**ノウハウ**

【検証対象】実例を基にしたソースコード (C言語)、もしくは仕様書

検査結果を説明してわかってもらえるかな?

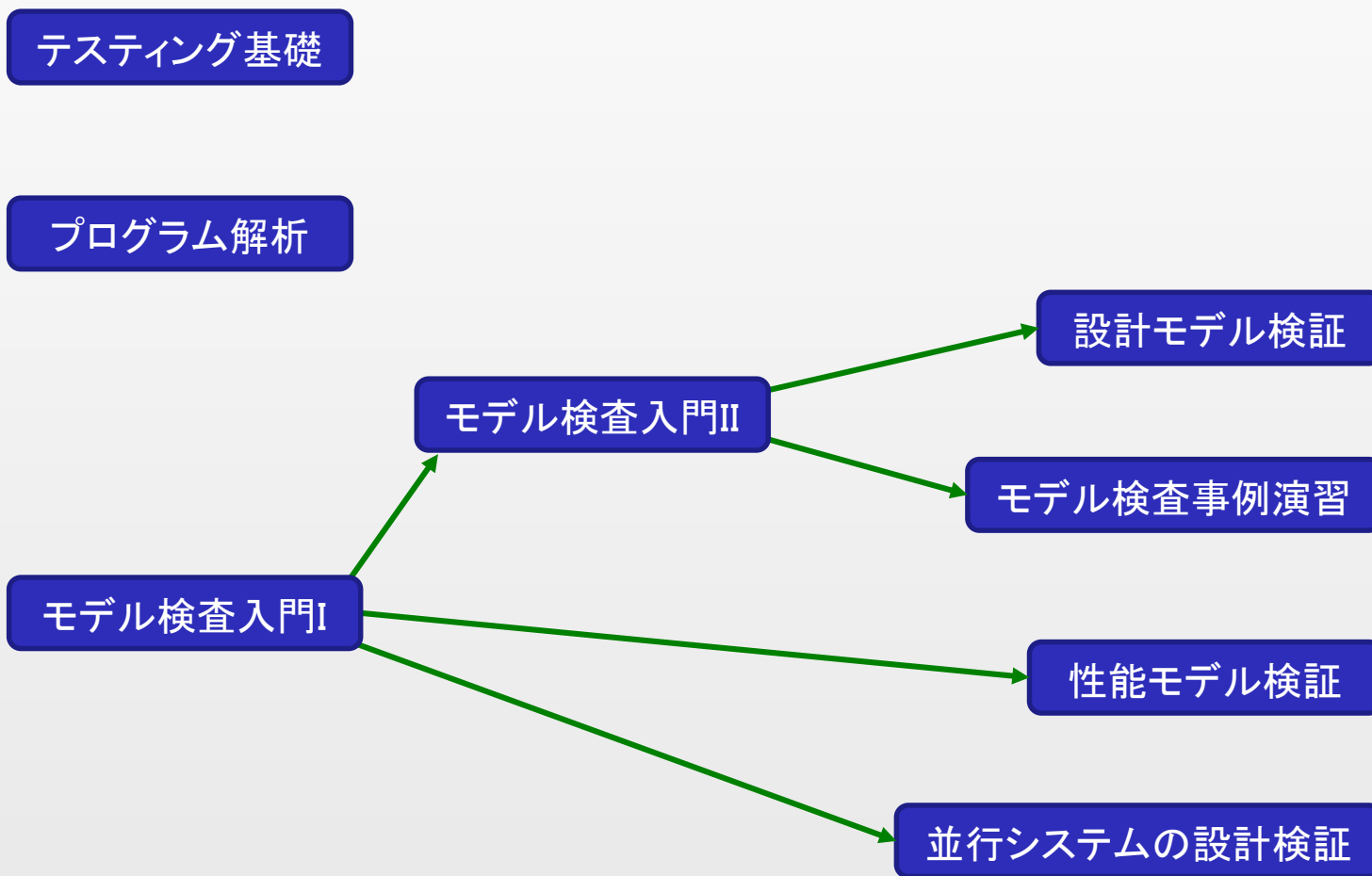


モデル検査技術を学ぶ意義

- 並行実行に起因する，再現が難しい不具合を追求する手段を獲得できる
- モデリング技術・抽象化技術の適用を学べる



科目間依存関係





目的別科目選択例 -1-

ソフトウェア工学の基礎科目を幅広く受講したい

- テスティング基礎
- モデル検査入門I

興味があれば, 下のどちらかまたは両方を追加

- モデル検査入門II + 設計モデル検証
- プログラム解析



目的別科目選択例 -2-

ソフトウェアの堅牢な構築に興味がある.

まずは3科目受講

- テスティング基礎
 - モデル検査入門I
 - モデル検査入門II + 設計モデル検証
- つづいて興味に応じて選択
- プログラム解析
 - モデル検査事例演習
 - 性能モデル検証
 - 並行システムの設計検証

[補足] モデル検査系科目群のまとめ

