性能モデル検証

講義予定

- 1,2回：導入、UPPAALの基本、ツール操作
- 3,4回:UPPAALモデルの意味と挙動/検証式
- 5,6,7回：
  - 開発プロセスと設計ドキュメント
  - UML ステート図とUPPAAL時間オートマトンの関係
  - Rhapsodyによる設計シミュレーション
  - 実機によるテスト
- 8,9回:
  - 検証プロセス
  - ギア制御例題
- 10～12回：
  - オーディオプロトコル例題
- 13～15回:
  - 個人例題

講義予定(第一回分)

- 導入
  - 設計モデル検査(基礎編)と性能モデル検査
- 性能モデル検証
  - モデル検査の基本
  - UPPAALとは
  - 時間オートマトン

性能モデル検証導入

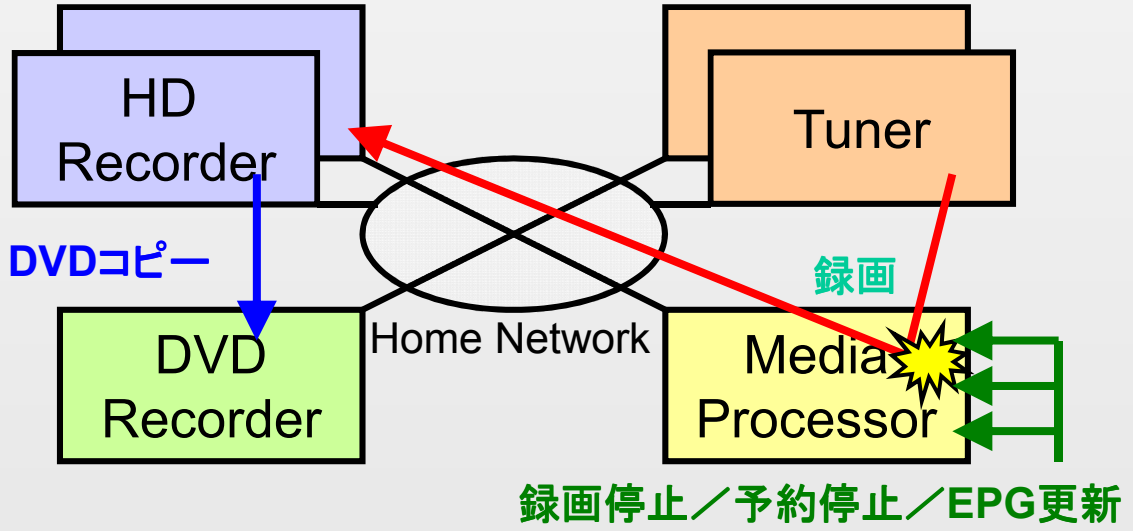
なぜ性能モデル検証を学ぶのか?

- 設計モデル検証(基礎編)で何を学んだか?
- 性能モデル検証では、新たに何を学ぶのか?
- なぜその内容を学ぶ必要があるのか?

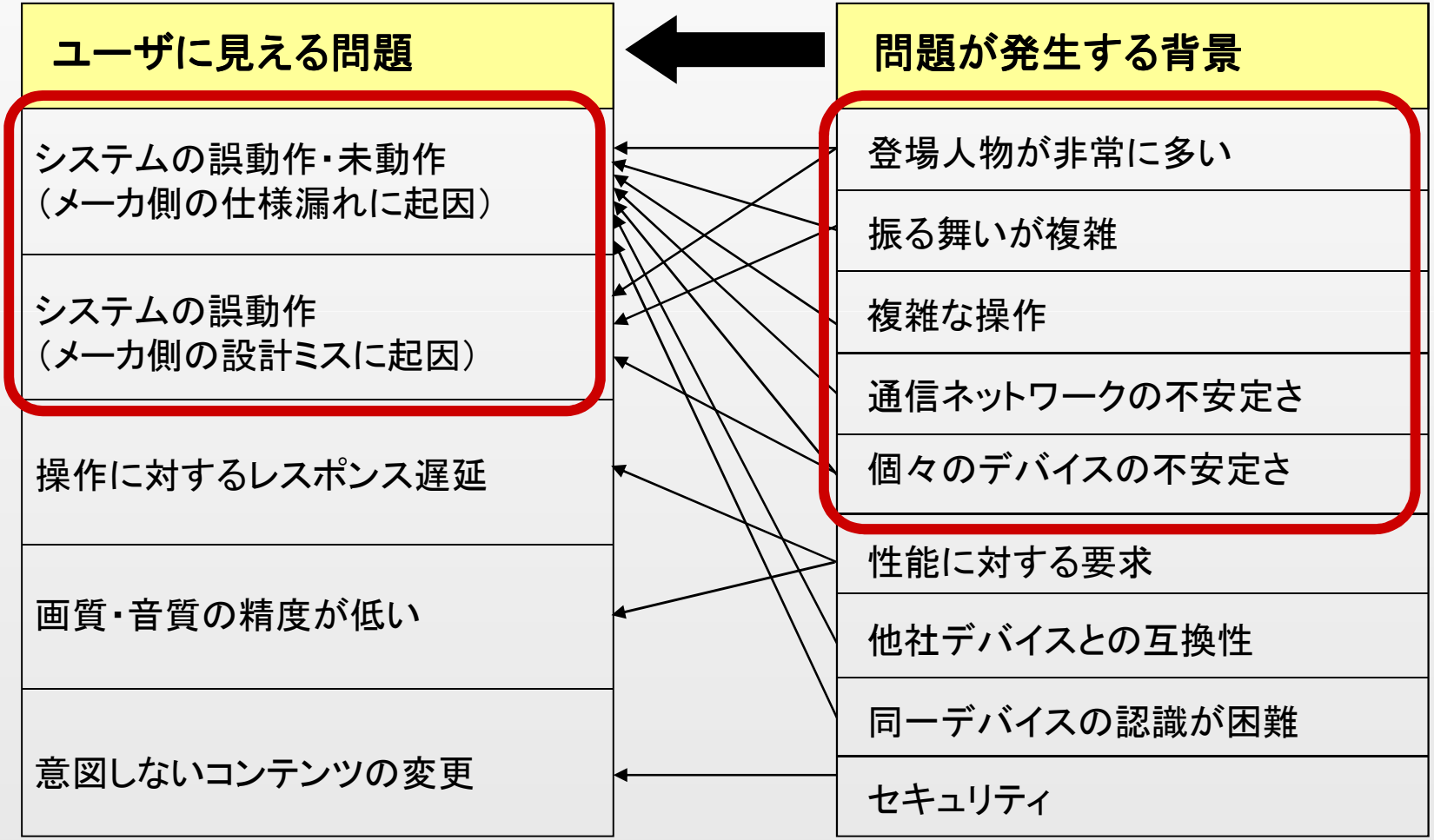
設計モデル検証(基礎編)で何を学んだか?
ネットワーク家電の課題

将来像
- 機能ブロックが分散
- 同種の機能ブロックが複数存在
- ホームネットワークを介して相互に接続

課題
- 機能検証
- 安全性、耐故障性の検証
- タイミング、性能の検証



設計モデル検証(基礎編)で何を学んだか?
基礎編でターゲットとした問題

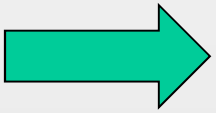


設計モデル検証(基礎編)で何を学んだか?

- 解決方法:
  - システムの動作をモデル化
  - 動作の集合が、論理式などで書かれた性質を満たすかどうかをツールが網羅的に検査
  - 満たさない場合は反例となる動作トレースを出力

性能モデル検証では、新たに何を学ぶのか?

- 基礎編で学ばなかったこと
  - 検証できる性質が、動作ロジックに関するもののみ
  - 時間制約などの性能面に関する性質の検証は学んでいない



性能編ではこれを学ぶ

なぜ性能面に関する性質の検証を学ぶのか?

- ネット家電などの組み込みシステムでは、性能面の性質が重要
- 性能面の性質の検査:従来はテストで解決
  - テストの難しさ→

性能面に関するテスト・デバッグの問題点

- 性能面に関し、別個に検査すべきテストケースは無数
  - どのパラメータがどのようにテスト結果に影響するかの場合分けが多岐にわたる
- テストケース通りに動作させるのが困難
  - 設定した時間パラメータ通りに動作させるためには、ハードウェアも含めたチューニングが必要
  - 別個にパラメータを指定する必要がなし

設計モデル検証(性能編)で学ぶ解

- 性能面に関するパラメータも含めて、システムの動作をモデル化
  - イベントの発生を順番でなく時刻範囲でモデル化可能
- 性能面に関する性質を記述し、動作の集合が満たすかどうかを検査
  - どのパラメータがどのように動作に影響するかも含めて網羅的に検査可能

使用する技術とツール

- 技術：性能面に関する性質を表現可能なオートマトンのモデル検査
- オートマトンの主な種類
  - リアルタイム性(時間制約)：時間オートマトン
  - 信頼性：確率オートマトン
- 本教材では、時間オートマトンによる時間制約性質の検証に絞る
  - 優れたツールUPPAALの存在

性能モデル検証
1. UPPAALの基本

UPPAALとはどのようなツールか?

- モデル検査とは?
- モデル検査ツールUPPAAL
- 時間オートマトンとは?
- UPPAALの使い方
- 検証式

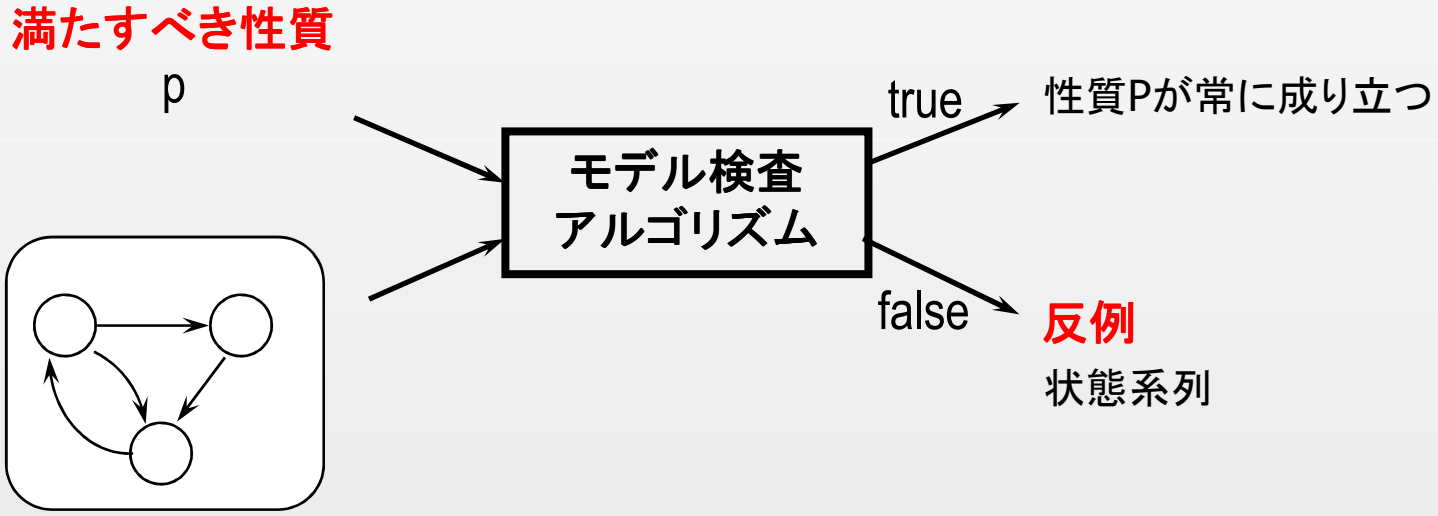
第2回

モデル検査とは?

- (基礎編の復習)

モデル検査

- 状態空間を網羅的に探索する検証手法



システムのモデル化

- 基本
  - 対象システムの振る舞いを有限状態モデルで表現する.
  - 状態…システムのスナップショット
  - 遷移…時間変化やイベントに伴う状態の変化
- 並行システム
  - 並行して複数の処理が実行されるシステムを1つの有限状態モデルで表現するのは難しい
  - 単機能処理を有限状態モデルとして表現し，合成する仕組みが備わっている
- メッセージ交換
  - 通信バッファを有限状態モデルとして表現する
  - 表現方法が備わっているモデルチェッカもある

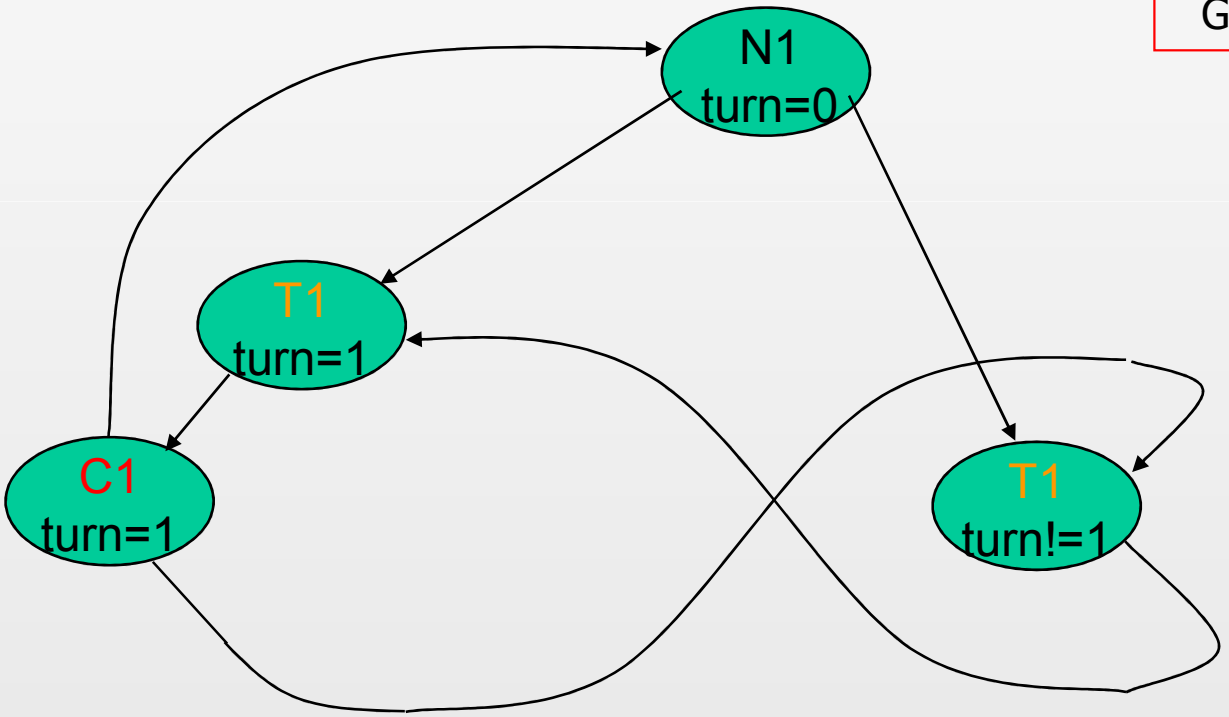
時相論理

- 状態の遷移や時間の経過の観点から、システムの性質を記述するための論理体系
  - LTL (Linear Time Temporal Logic)
    - 線形時間時相論理
  - CTL (Computation Tree Logic)
    - 分岐時間時相論理
- システムが満たすべき性質の記述に利用

検証例

- ネットワーク接続されたDVDレコーダーと HDDレコーダーが1つのチューナを共有する
- 排他制御の振る舞い全体を状態機械でモデル化 （DVDとHDDの振る舞いを合成した状態機械）

進行性
DVDが資源獲得を要求したら、いつか必ず利用できる
G (T1 -> F C1)　**True**

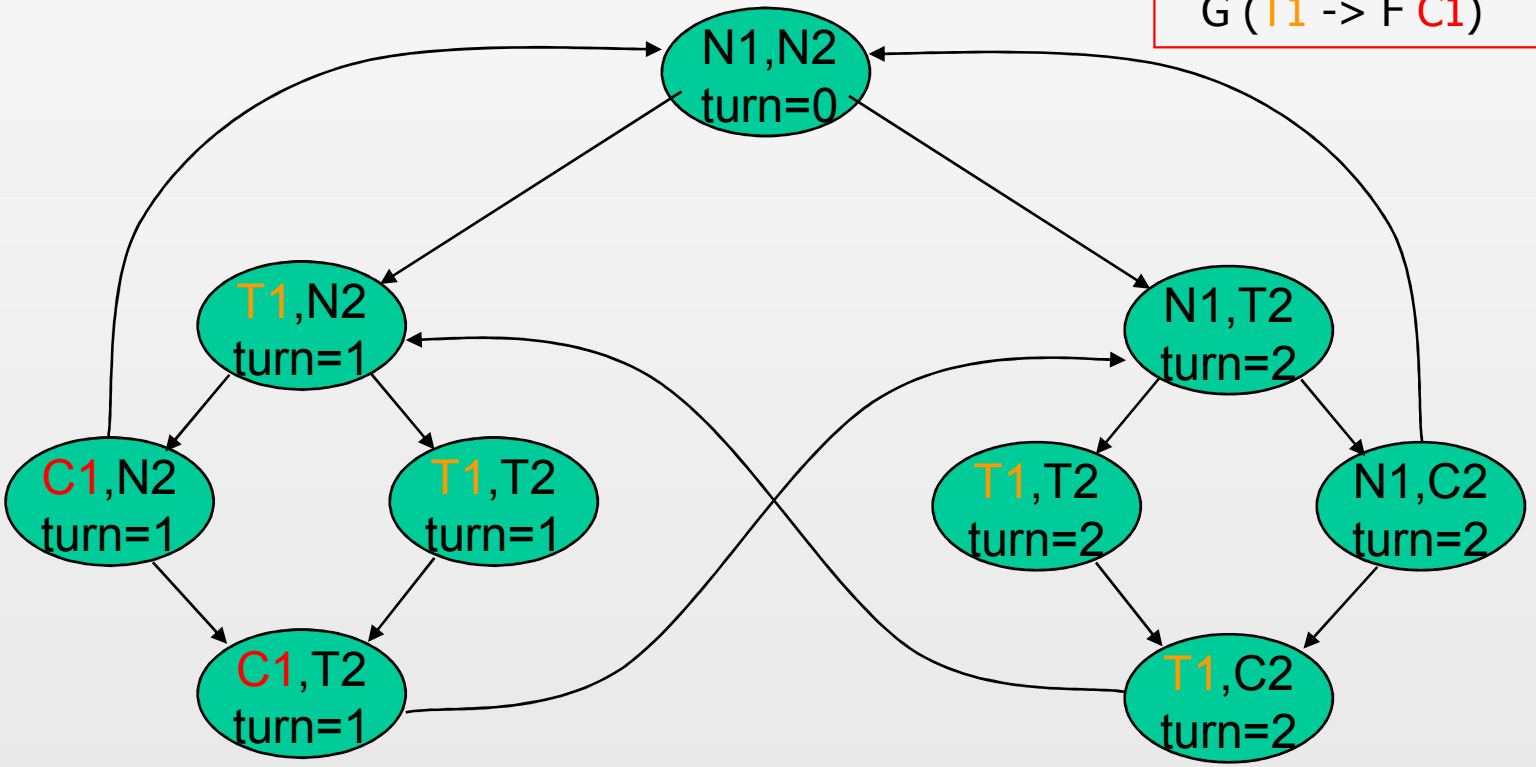


N = noncritical　T = trying,　C = critical,　1 : DVD

検証例

- ネットワーク接続されたDVDレコーダーと HDDレコーダーが1つのチューナを共有する
- 排他制御の振る舞い全体を状態機械でモデル化 （DVDとHDDの振る舞いを合成した状態機械）

進行性
DVDが資源獲得を要求したら、いつか必ず利用できる
G (T1 -> F C1)　**True**



N = noncritical　T = trying,　C = critical,　1：DVD,　2：HD

基礎編における検証の視点

現象	理由	検証のパターン
仕様に書かれている機能が実行でない、または異常処理から復旧できない。	資源を競合している	デッドロック
途中まで正常に機能実行するが、途中で停止する、または異常動作を行なう。	初期状態に戻れない 無限ループにおちいる	ライブロック
取り合う資源 CPU Media Processor チューナー DVD HDD 予約操作（録画予約時間） リモコン 番組情報(EPG) 競合が起こりうる操作 番組録画 vs DVDコピー リモコン予約操作 vs 遠隔予約操作 ダイレクト録画 vs 予約録画 HDDへのコピー vs 予約録画 編集作業 vs 予約録画	機能が満たすべき性質が侵害される（メモリオーバフロー等）	不変性質の侵害
	到達できない状態が存在する	到達可能性の侵害
ユーザの同じ操作に対してシステムの動作がその時々で異なる	1つの状態において、同じイベントに対して複数の動作が定義されている	非決定性

検証可能な性質

代表的な5つの性質

例えば、資源利用に対するマルチプロセス間の排他制御を考える。

- 到達可能性
  - 資源を利用していない待機状態へ到達できる
- 進行性
  - 資源を繰り返し(無限回)利用することができる
- 安全性
  - 資源競合は決して発生しない
  - 資源獲得のデッドロックが発生しない
- 応答性
  - 外部環境からの資源獲得要求イベントに呼応して、必ず確認イベントが発生する
- 不変性
  - 資源を利用するプロセスは常に1つである

モデルチェッカ毎に、検証できる性質も記法も異なる

モデル検査ツールUPPAAL

UPPAALとは

- Uppsala universityとAalborg universityが共同開発
  - UPPsala + AALborg = UPPAAL
- 通信を伴う並行処理、排他処理が得意
- 時間を扱うことが可能
- GUIベースでモデルを作成
- 開発ヒストリ
  - 1999/6　:初版
  - 1999/9 :UPPAAL2K (3.0.0に相当?)
  - 2006/4 :UPPAAL 3.6 beta3
  - 2006/5 :UPPAAL 4.0.0
  - 2009/9 :UPPAAL 4.0.10　ただしテキストは4.0.6ベース

代表的なモデル検査ツールの比較

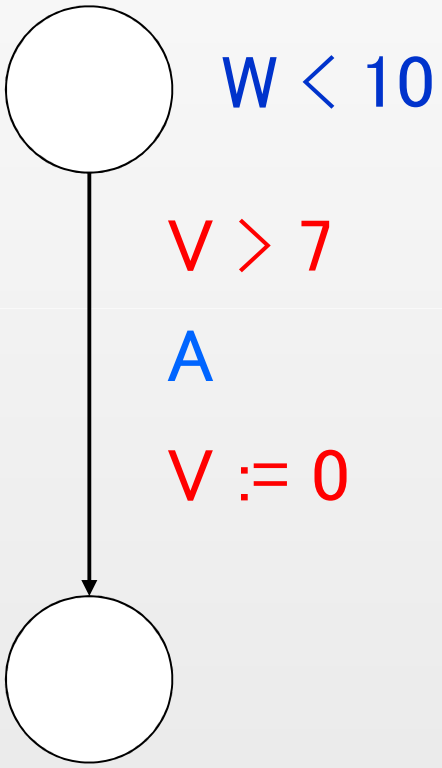
用途	ツール名	Real Time	GUI
回路検証	NuSMV		○
非同期並行プロセス	UPPAAL	○	○
	KRONOS	○	
	SPIN		
ソースコード検証	Java Path Finder		
	Static Driver Verifier		

UPPAALの記述

- システムモデル
  - プロセス毎に時間オートマトンでモデルを記述
    - UPPAAL独自の拡張あり
- 満たすべき性質
  - 時相論理式CTL(Computation Tree Logic)で検証式を記述
    - UPPAAL独自の制限、拡張あり

時間オートマトンとは?

オートマトンの基本

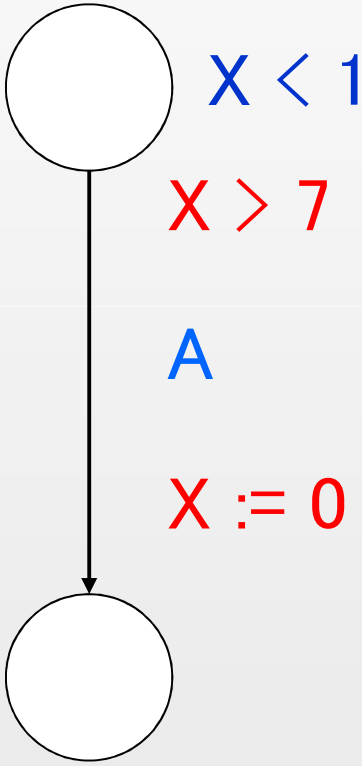


- 滞在可能条件
  - 条件が成立している間はロケーションに滞在可能
  - 変数値に対する制約で記述
- 遷移条件(ガード)
  - 条件が成立すると遷移可能
  - 変数値に対する制約で記述
- 同期通信アクション
  - 遷移の際に行われる他プロセスとのやりとり(後述)
- 変数値の更新
  - 遷移の際に行われる変数への代入

時間変数

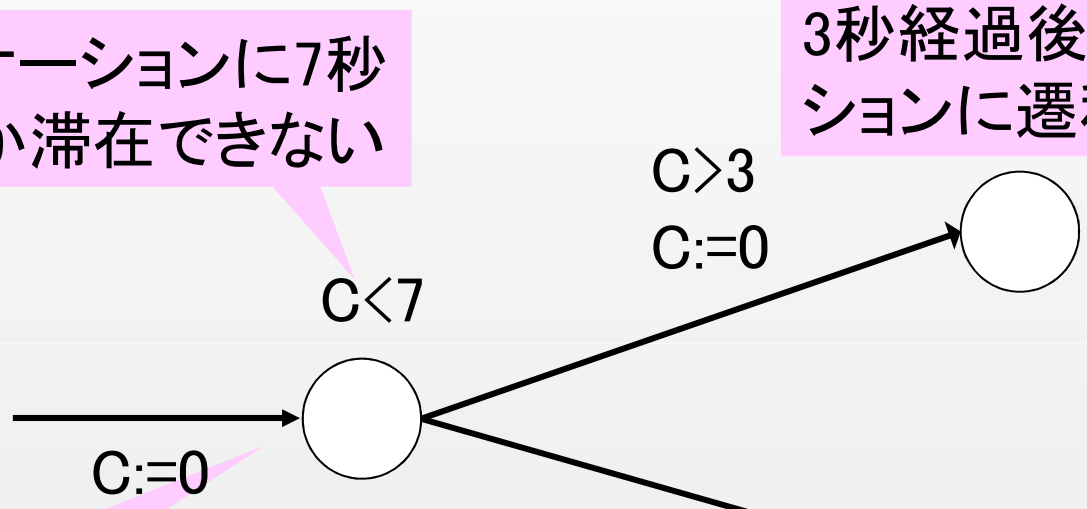
- 時間変数とは
  - 初期状態では0
  - 時間の経過とともに値が増加
  - 複数定義可能
    - すべての時間変数は一定の速度で値が増加
  - 値の代入操作により、初期化(0化)可能

時間オートマトンの基本



- 滞在可能条件
  - 時間変数に対する条件で、時間制約を記述
- 遷移条件(ガード)
  - 時間変数に対する条件で、時間制約を記述
- 同期通信アクション
  - 後述
- 変数値の更新
  - 時間変数の初期化
- ロケーションには(滞在可能条件を満たしている間は)任意の時間滞在できる
- 遷移は時間経過しない

時間オートマトンの挙動例



このロケーションに7秒未満しか滞在できない

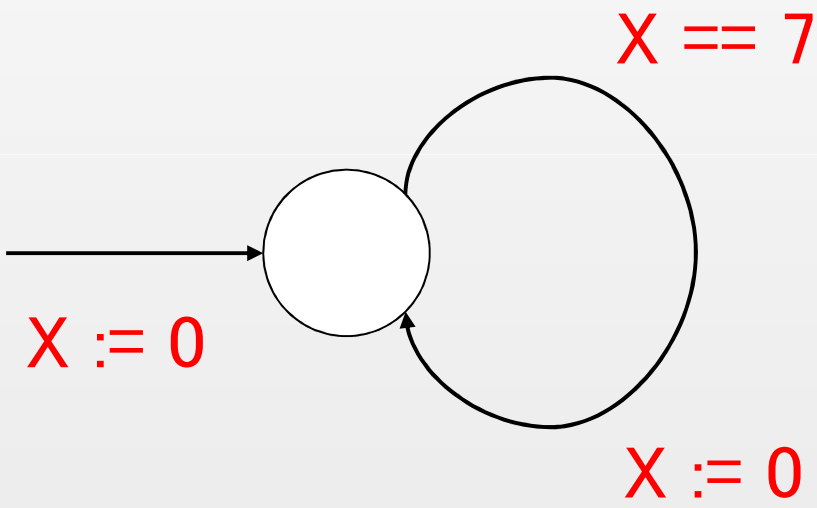
3秒経過後、このロケーションに遷移可能

このロケーションに遷移した時点でCは0に初期化

5秒経過後、このロケーションに遷移可能

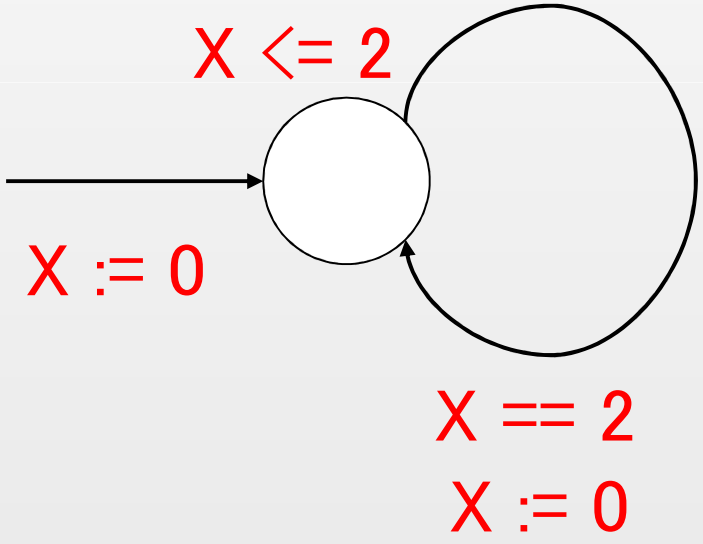
C:時間変数とし説明上単位を仮に秒とする

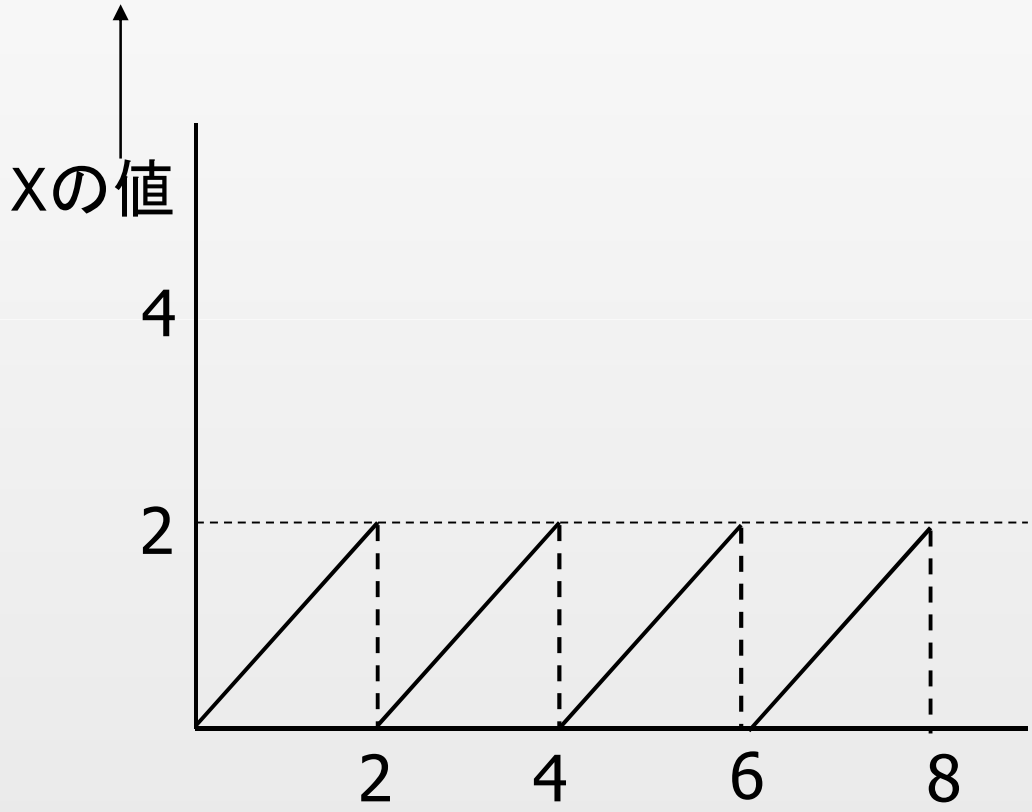
時間オートマトンの挙動パターン
周期タスク



時間オートマトンの挙動パターン(1)

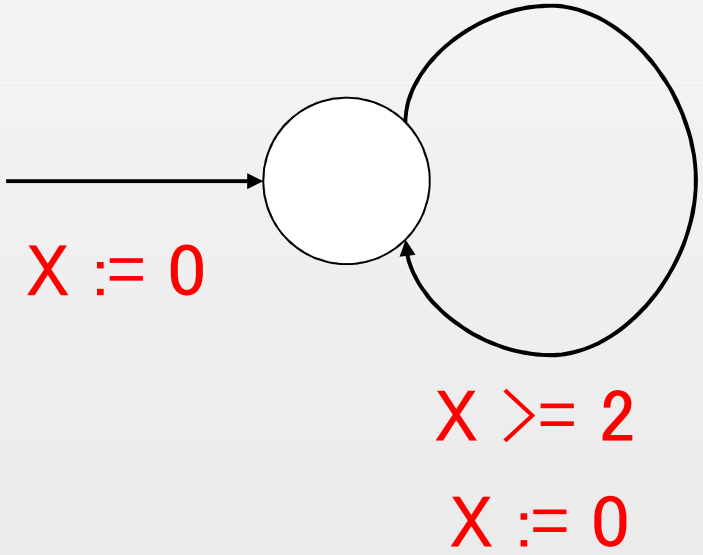
- 周期タスク

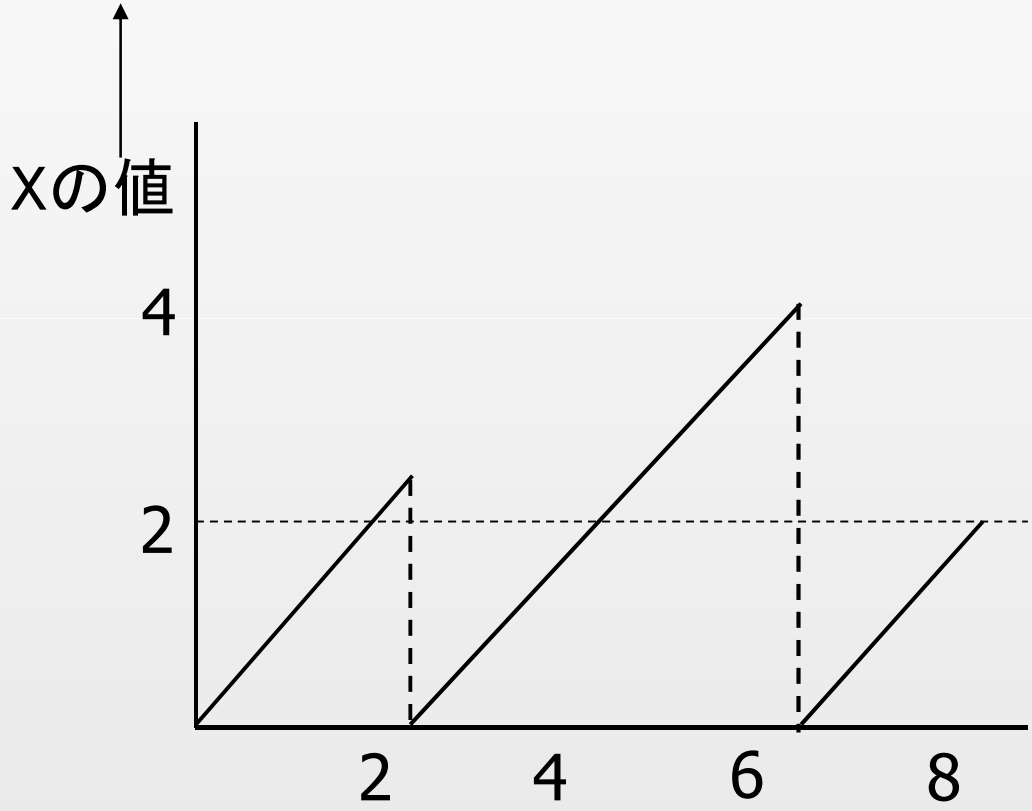




時間オートマトンの挙動パターン(2)

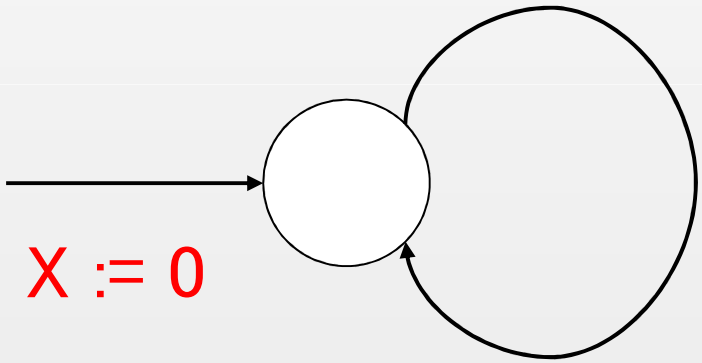
- 散発的(sporadic)タスク





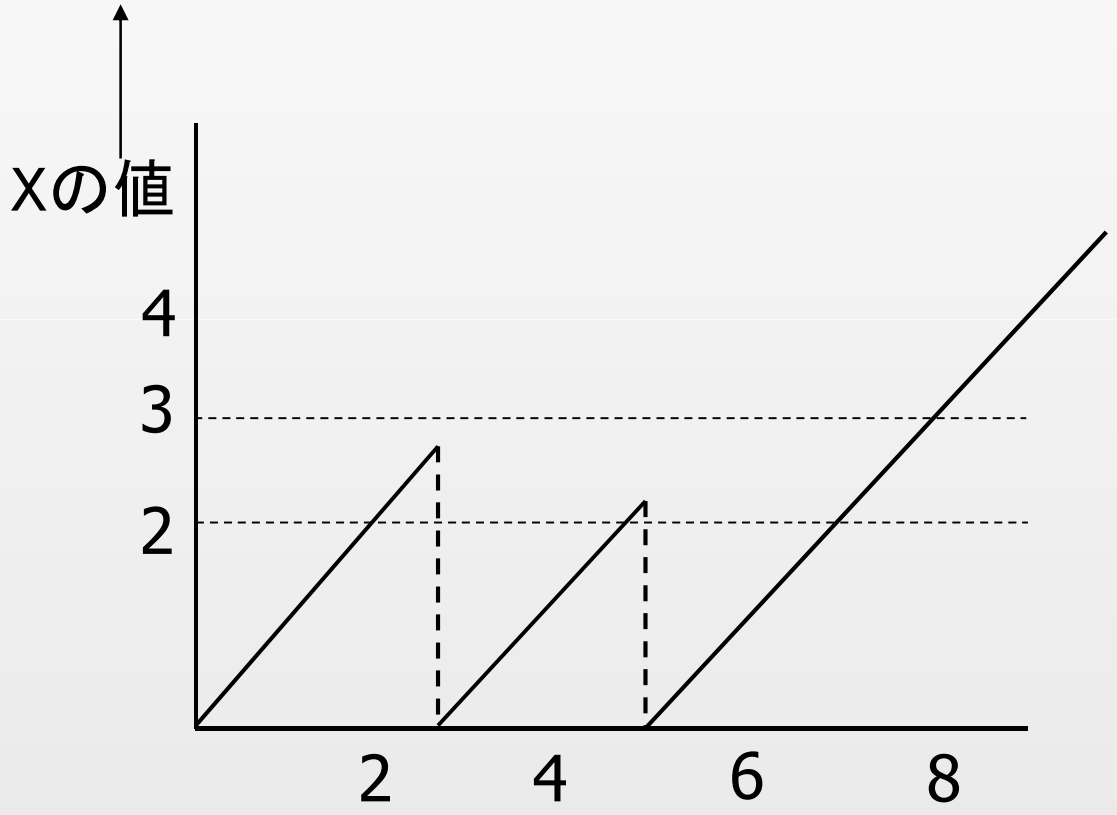
時間オートマトンの挙動パターン(3)

- 周期的タスク



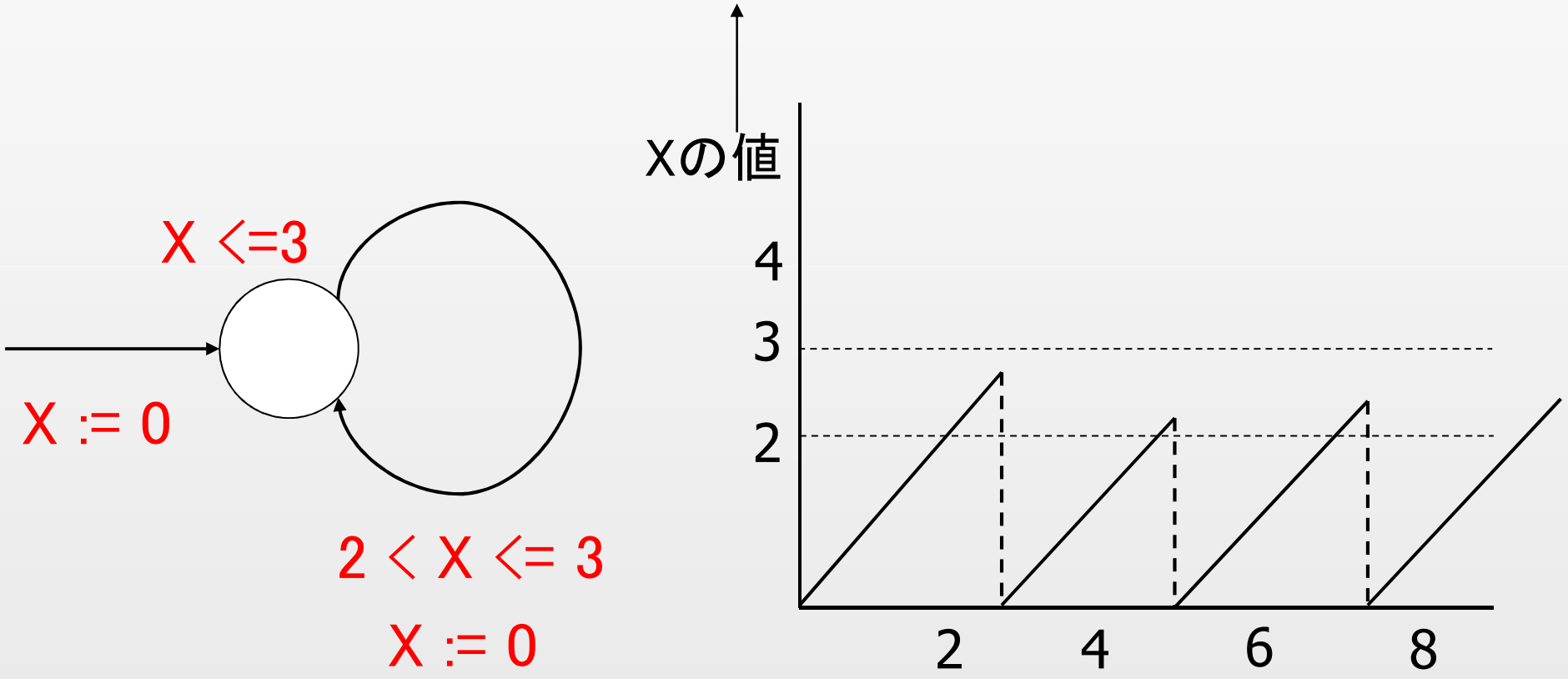
2 < X <= 3

X := 0



時間オートマトンの挙動パターン(4)

- 周期的タスク(ロケーションの滞在制約有り)



時間制約

x,yを時間変数とすると、制約は

$$\alpha ::= x \angle c \mid x-y \angle c \mid \neg\alpha \mid (\alpha \wedge \alpha)$$

where $c \in N$ and $\angle \in \{ <, \leqq \}$

で定義できる

時間オートマトンの操作的意味論

- 各時間変数にdを加えても、現在状態の時間制約を満たすとき、現在状態にとどまったまま時間d経過できる
  - delay transition
- ある遷移条件が満たされていれば、時間経過0でその遷移のアクションを実行し、リセット変数の値を0にして、次状態に遷移できる
  - action transition

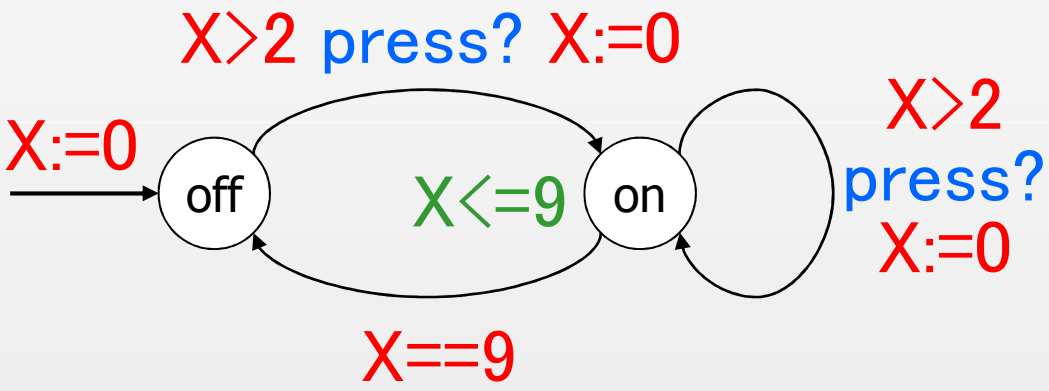
現在状態　時間変数値　　滞在可能条件

$$- \langle l, u \rangle \xrightarrow{d} \langle l, u + d \rangle \; if \; u \in I(l) \; and \; (u + d) \in I(l) \; for \; a \; non\text{-}negative \; real \; d \in \mathbb{R}_+$$

$$- \langle l, u \rangle \xrightarrow{a} \langle l', u' \rangle \; if \; l \xrightarrow{g,a,r} l', u \in g, u' = [r \mapsto 0]u \; and \; u' \in I(l')$$

ガード、アクション、リセット(する時間)変数

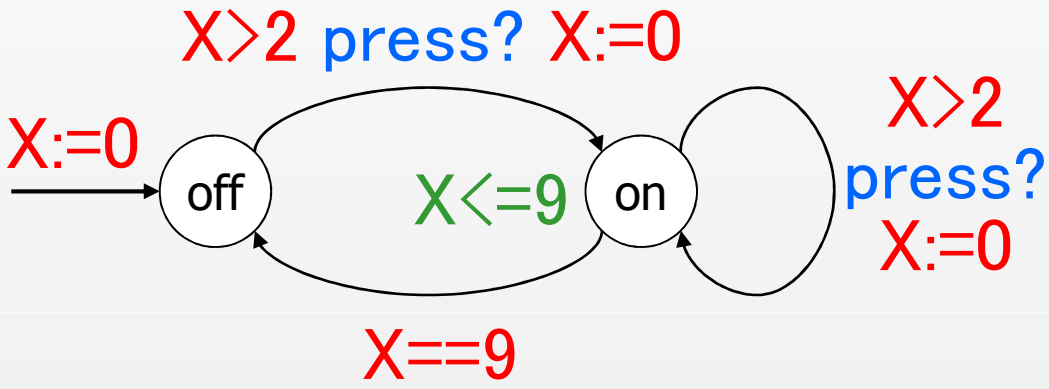
照明スイッチの例



- スイッチは最後に消灯後最低2時間単位経過後に点灯
- 点灯後9時間単位押されないと自動的に消灯

「press?」は受信イベント

照明スイッチの例



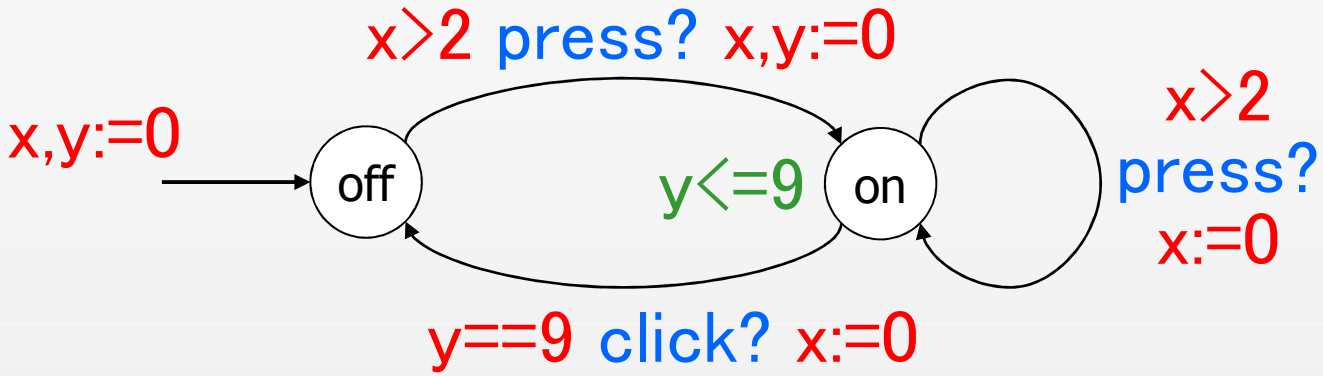
$$(off,x=0) \xrightarrow{3.5} (off,x=3.5) \xrightarrow{press}$$

$$(on,x=0) \xrightarrow{\pi} (on,x=\pi) \xrightarrow{press}$$

$$(on,x=0) \xrightarrow{\pi} (on,x=\pi) \xrightarrow{9-\pi}$$

$$(on,x=9) \longrightarrow (off,x=9)$$

照明スイッチの例(2時間変数)



$(off, x=y=0) \xrightarrow{3.5} (off, x=y=3.5) \xrightarrow{press}$

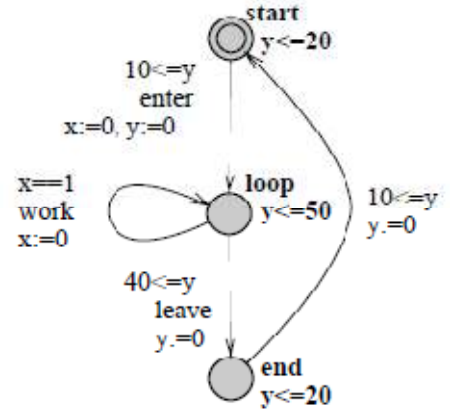
$(on, x=y=0) \xrightarrow{\pi} (on, x=y=\pi) \xrightarrow{press}$

$(on, x=0, y=\pi) \xrightarrow{3} (on, x=3, y=\pi+3) \xrightarrow{9-(\pi+3)}$

$(on, x=9-\pi, y=9) \xrightarrow{click} (off, x=0, y=9)$

時間オートマトン(まとめ)

- 時間制約を、x, yなどの時間変数に対し、x < 5、x − y < 3、などの式で表す
- 各遷移に、遷移条件、アクションラベル、およびリセット時間変数を指定する
- 各状態に、その状態にとどまるための時間制約を指定する
- 例

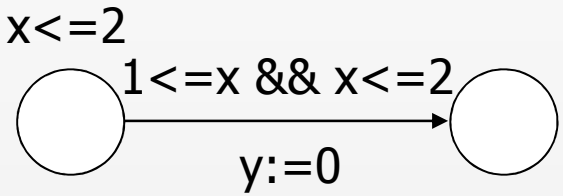


UPPAALとはどのようなツールか?

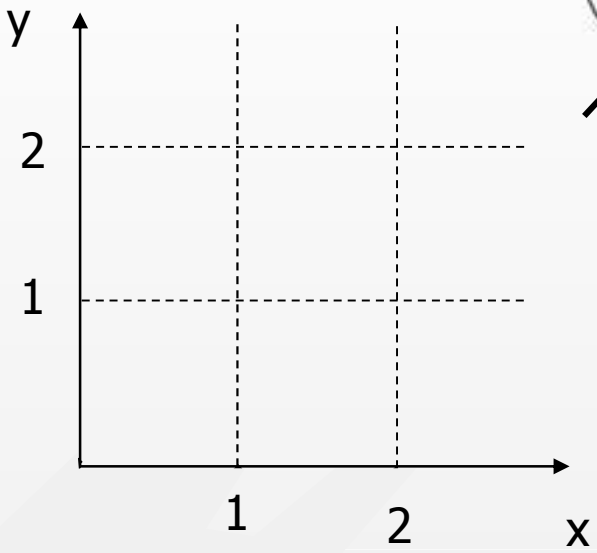
- モデル検査とは?
- モデル検査ツールUPPAAL
- 時間オートマトンとは?
- UPPAALの使い方
- 検証式

} 第2回

課題

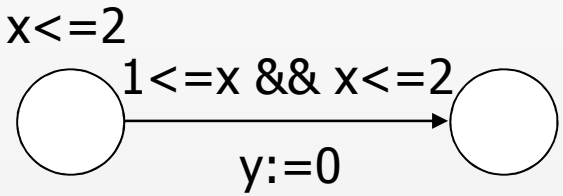


x,y:時間変数

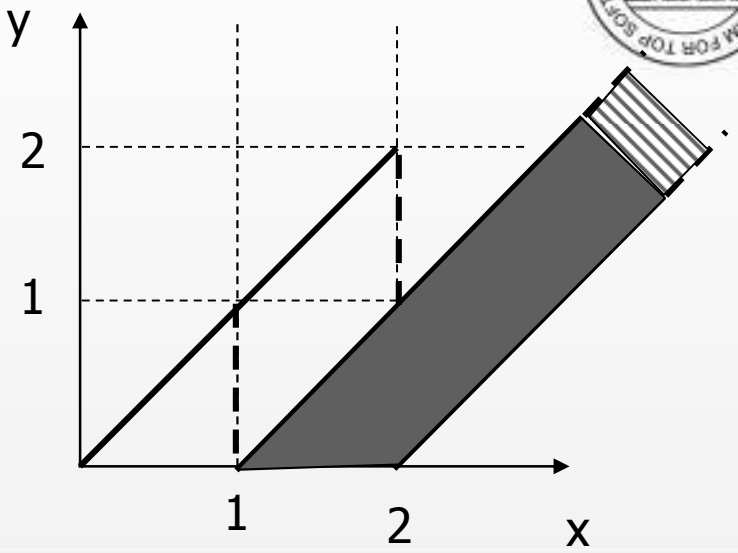


上図の時間オートマトンにおいて、x,yが取り得る値の範囲は右図の様になる。

課題



x,y:時間変数



上図の時間オートマトンにおいて、x,yが取り得る値の範囲は
右図の様になる。
P42の時間オートマトンにおける、x,yが取り得るの値の範囲を示せ。