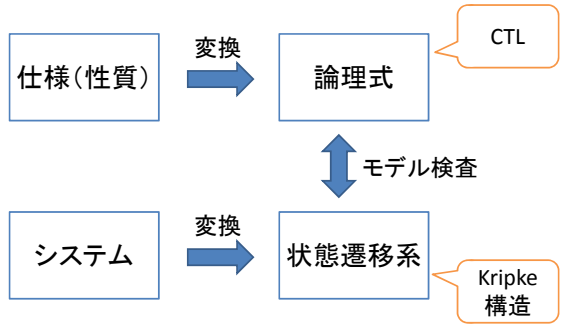
ソフトウェアモデル論（2010年度）
第13回・2010/12/13

桑原 寛明
情報理工学部 情報システム学科

モデル検査

- 状態遷移系として記述されたシステムが、論理式として記述された性質を満たすか否か、網羅的かつ機械的に検証する手法
- 利点
  - 網羅的、機械的、反例
- 例えば、プログラムが必ず停止すること、デッドロックしないこと、などを検証できる

モデル検査の手順

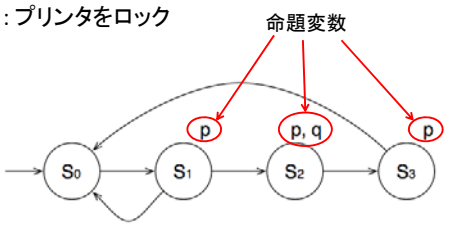


Kripke構造の定義

- Kripke構造 M = (S, R, L)
  - S : 状態の有限集合
  - R : 遷移関係
    - $R \subseteq S \times S$
    - R(s, s') : 状態 s から s' への遷移がある
  - L : ラベル付け関数
    - $L : S \rightarrow 2^{PV}$
    - PV は命題変数の集合
    - 各状態にその状態で真となる命題を表す命題変数を割り当てる関数

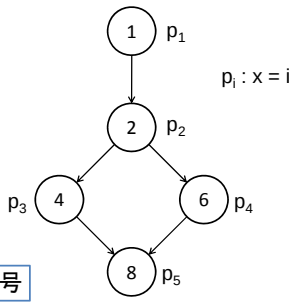
Kripke構造の例

- COPY = (S, R, L)
  - $L(s_0) = \emptyset$, $L(s_1) = L(s_3) = \{p\}$, $L(s_2) = \{p, q\}$
  - p : スキャナをロック
  - q : プリンタをロック



Kripke構造の例

```
1:  x = 1;
2:  x = 2;
3:  if (...) {
4:    x = 3;
5:  } else {
6:    x = 4;
7:  }
8:  x = 5;
```



状態は行番号

経路

- 遷移関係に従って移り変わる状態の列
- $\sigma = s_0 s_1 s_2 \cdots$
  - すべての $i \geq 0$ に対して $R(s_i, s_{i+1})$
- 関係 R(s, s') は、システムの状態が s の場合、次の時刻で状態が s' に変わると理解する

時相論理

- 命題論理は、ある瞬間の状況に関する命題のみ扱える
  - 前後（過去、未来）の状況との関連は扱えない
- 扱えるように拡張したものが時相論理
  - 「いつか停止する」
  - 「x の値はずっと正である」
    など

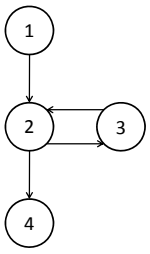
時間のとらえ方

- 離散 vs 連続
  - 離散: 単位時間を仮定
  - 連続: 任意の2時刻間に別の時刻の存在を仮定
- 点 vs 区間
- 未来 vs 過去
- 分岐 vs 線形
  - 分岐: 時間の流れによる状態変化のすべての可能性を同時に考慮
  - 線形: 一つの可能性のみを選択

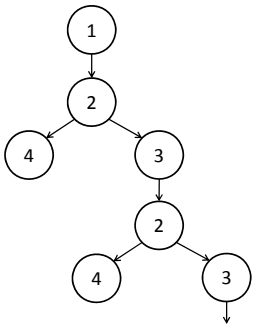
CTL（計算木論理）

- 離散、点、未来、分岐
- 計算木に対する論理
  - 計算木は、ある状態を開始状態とするすべての経路を1つにまとめた木構造
  - 開始状態が木構造の根
- 経路の選択
  - すべての経路において、ある経路において
- タイミングの選択
  - 次、将来のいつか、今後ずっと、ある時点まで

計算木



状態遷移



計算木

CTLの論理式

1. 命題変数は状態式
2. P, Q が状態式ならば ¬P, P∧Q, P∨Q, P→Q は状態式
3. P が経路式ならば **A** P, **E** P は状態式
4. P, Q が状態式ならば **X** P, **F** P, **G** P, P **U** Q は経路式
5. 以上が状態式と経路式のすべてであり、状態式がCTLの論理式のすべて

CTLの意味論

- M, s ⊨ P
  - Kripke構造 M = (S, R, L) における状態 s∈S を開始状態とする計算木においてCTL式 P が成り立つ
- PV は命題変数の集合であり p∈PV

CTLの意味論

- M, s ⊨ p
  - p∈L(s)
- M, s ⊨ ¬P
  - M, s ⊨ P でない
- M, s ⊨ P∧Q
  - M, s ⊨ P かつ M, s ⊨ Q
- M, s ⊨ P∨Q
  - M, s ⊨ P または M, s ⊨ Q
- M, s ⊨ P→Q
  - M, s ⊨ P ならば M, s ⊨ Q

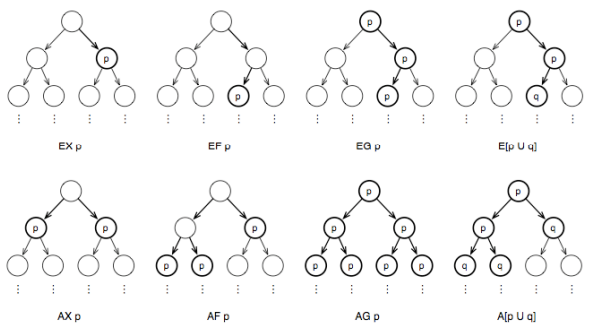
CTLの意味論

- M, s ⊨ **AX** P
  - σ(0)=s なるすべての経路σにおいて M, σ(1) ⊨ P
- M, s ⊨ **EX** P
  - σ(0)=s かつ M, σ(1) ⊨ P なる経路σが存在する
- M, s ⊨ **AF** P
  - σ(0)=s なるすべての経路σにおいてある i≥0 が存在して M, σ(i) ⊨ P
- M, s ⊨ **EF** P
  - σ(0)=s かつ M, σ(i) ⊨ P なる i≥0 が存在する経路σが存在する

CTLの意味論

- M, s ⊨ **AG** P
  - σ(0)=s なるすべての経路σにおいて任意の i≥0 に対して M, σ(i) ⊨ P
- M, s ⊨ **EG** P
  - σ(0)=s かつ任意の i≥0 に対して M, σ(i) ⊨ P なる経路σが存在する
- M, s ⊨ **A** [P **U** Q]
  - σ(0)=s なるすべての経路σにおいてある j≥0 が存在して M, σ(j) ⊨ Q かつ 0≤i<j に対して M, σ(i) ⊨ P
- M, s ⊨ **E** [P **U** Q]
  - σ(0)=s かつ、ある j≥0 が存在して M, σ(j) ⊨ Q かつ 0≤i<j に対して M, σ(i) ⊨ P なる経路σが存在する

CTLの意味論



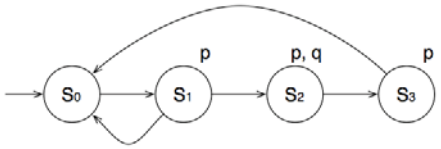
経路演算子、時相演算子

- ¬**AX** P = **EX** ¬P
- ¬**AF** P = **EG** ¬P
- ¬**AG** P = **EF** ¬P

- **EX**, **EG**, **EU** があれば他の演算子は表現可能
  - **EF** P = **E**[⊤ **U** P]　　⊤は恒真（任意の状態で真）
  - **A**[P **U** Q] = ¬(**EG** ¬Q ∨ **E**[¬Q **U** (¬P∧¬Q)])
  - **EX**, **EG**, **EU** の組み合わせに限らない

例

- COPY, $s_0 \models$ **EF**(p∧q)
  - p, q がともに成り立つ状態に到達できる経路がある
- COPY, $s_0 \models \neg$**AF**(p∧q)
  - すべての経路で p, q がともに成り立つ状態に到達できるわけではない

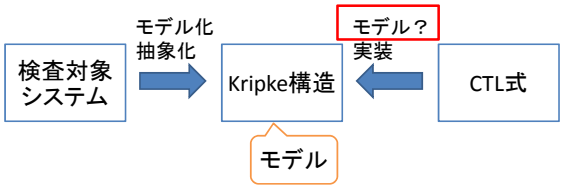


LTL（線形時相論理）

- 離散、点、未来、線形
- ある状態を開始状態とする1つの経路に着目する論理
- 詳細は省略

なぜ「モデル検査」と呼ぶか

- モデル検査は、Kripke構造がCTL式のモデルになっているか検査すること
- 一般的には、状態遷移系が（時相）論理式のモデルになっているか検査



モデル検査アルゴリズム

- Kripke構造 M、状態 s、CTL式 P
- CTL式の演算子は¬、∨、EX、EG、EU
- Check(M, P)
  - M の各状態に対して P の各部分式のうちで成り立つものを求める
- 最終的に s で成り立つ式の中に P が含まれていれば M, s ⊨ P

モデル検査アルゴリズム

case: $P \in PV$
  for all $s \in S$ **do**
    if $P \in L(s)$ **then** $label(s) := label(s) \cup \{P\}$
case: $P \equiv \neg Q$
  $Check(M, Q)$
  for all $s \in S$ **do**
    if $Q \notin label(s)$ **then** $label(s) := label(s) \cup \{\neg Q\}$
case: $P \equiv Q_1 \vee Q_2$
  $Check(M, Q_1)$
  $Check(M, Q_2)$
  for all $s \in S$ **do**
    if $Q_1 \in label(s)$ **or** $Q_2 \in label(s)$ **then** $label(s) := label(s) \cup \{Q_1 \vee Q_2\}$

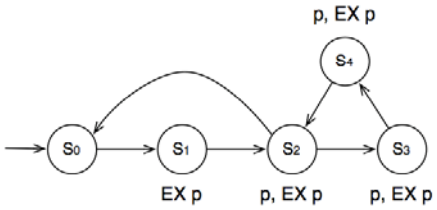
モデル検査アルゴリズム

- **EX** Q の場合
  - 一つ遷移すると Q が成り立つ状態に到達できる状態では **EX** Q が成り立つ

case: $P \equiv \mathbf{EX}\, Q$
  $Check(M, Q)$
  for all $s \in \{s \mid Q \in label(s)\}$ **do**
    for all s' **such that** $R(s', s)$ **do**
      $label(s') := label(s') \cup \{\mathbf{EX}\, Q\}$

モデル検査アルゴリズム

- **EX** Q の場合（続き）



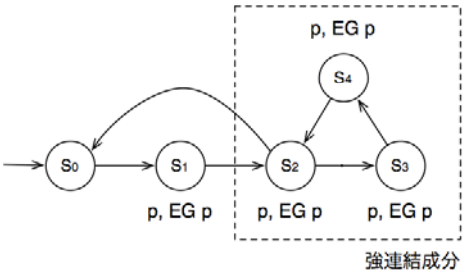
モデル検査アルゴリズム

- **EG** Q の場合
  - 常に Q が成り立っている経路を見つける
- Q が成り立つ状態のみに着目
- 強連結成分
- 強連結成分中のいずれかの状態に到達可能

```
Check(M, P)
S' := {s | P ∈ label(s)}
SCC := {C | C は S' の強連結成分 }
T := ⋃_{C∈SCC} {s | s ∈ C}
for all s ∈ T do label(s) := label(s) ∪ {EG P}
while T ≠ ∅ do
  choose s ∈ T
  T := T − {s}
  for all s' ∈ S' such that R(s', s) do
    if EG P ∉ label(s') then
      label(s') := label(s') ∪ {EG P}
      T := T ∪ {s'}
```


モデル検査アルゴリズム

- **EG** Q の場合（続き）



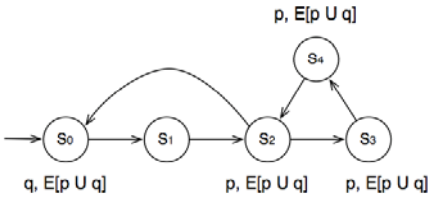
モデル検査アルゴリズム

- **E**[Q_1 **U** Q_2] の場合
  - Q_2 が成り立っている状態から遷移をさかのぼって Q_1 が成り立っている間 **E**[Q_1 **U** Q_2] が成り立つ

```
Check(M, P)
Check(M, Q)
T := {s | Q ∈ label(s)}
for all s ∈ T do label(s) := label(s) ∪ {E[P U Q]}
while T ≠ ∅ do
  choose s ∈ T
  T := T − {s}
  for all s' such that R(s', s) do
    if E[P U Q] ∉ label(s') and P ∈ label(s') then
      label(s') := label(s') ∪ {E[P U Q]}
      T := T ∪ {s'}
```

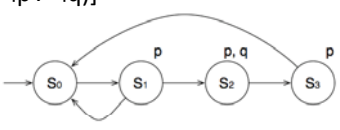

モデル検査アルゴリズム

- **E**[Q_1 **U** Q_2] の場合（続き）



例

- COPY, $s_0 \models$ **E**[⊤**U**¬(¬p∨¬q)]
  - **EF**(p∧q) か？



- label(s_0) = {⊤, ¬p, ¬q, ¬p∨¬q, **E**[⊤**U**¬(¬p∨¬q)}
- label(s_1) = {⊤, ¬q, ¬p∨¬q, **E**[⊤**U**¬(¬p∨¬q)}
- label(s_2) = {⊤, ¬(¬p∨¬q), **E**[⊤**U**¬(¬p∨¬q)}
- label(s_3) = {⊤, ¬q, ¬p∨¬q, **E**[⊤**U**¬(¬p∨¬q)}