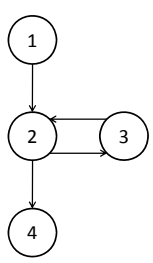
ソフトウェアモデル論（2011年度）
第14回・2012/01/06

桑原 寛明
情報理工学部 情報システム学科

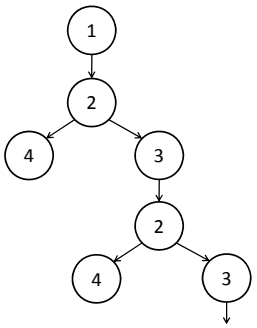
CTL（計算木論理） （復習）

- 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 （復習）



状態遷移



計算木

CTLの論理式 （復習）

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 （復習）

- 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

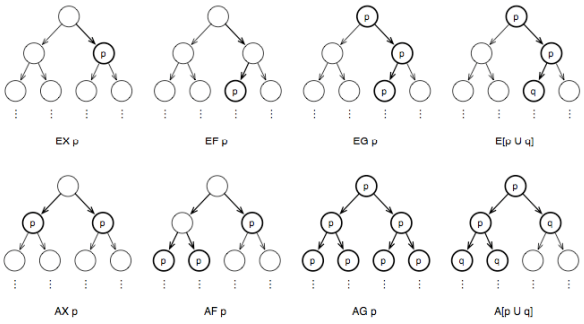
CTLの意味論 （復習）

- 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 （復習）

- 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 （復習）



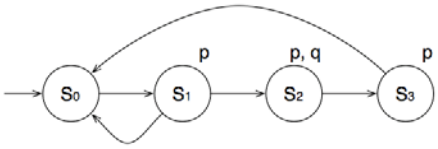
経路演算子、時相演算子 （復習）

- ¬**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**[T **U** P]　　T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 （復習）

- COPY, s_0 ⊨ **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 ⊨ ¬**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



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 の各部分式の中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** $label(s) := label(s) \cup \{Q_1 \vee Q_2\}$

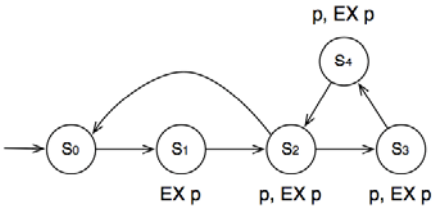
モデル検査アルゴリズム

- **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）



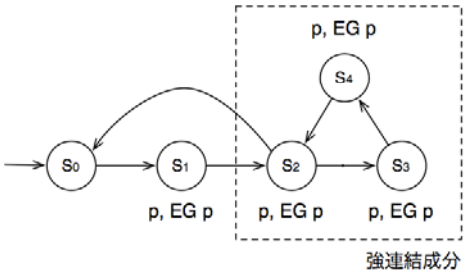
モデル検査アルゴリズム

- **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
  - 任意の2状態間に経路が存在
  - 極大
- 強連結成分中のいずれかの状態に到達可能

$Check(M, P)$
$S' := \{s \mid P \in label(s)\}$
SCC := $\{C \mid C$ は S'の強連結成分 $\}$
$T := \bigcup_{C \in SCC}\{s \mid s \in C\}$
for all $s \in T$ do $label(s) := label(s) \cup \{\mathbf{EG}\,P\}$
while $T \neq \emptyset$ do
  choose $s \in T$
  $T := T - \{s\}$
  for all $s' \in S'$ such that $R(s', s)$ do
    if $\mathbf{EG}\,P \notin label(s')$ then
      $label(s') := label(s') \cup \{\mathbf{EG}\,P\}$
      $T := T \cup \{s'\}$

モデル検査アルゴリズム

- **EG** Q の場合（続き）



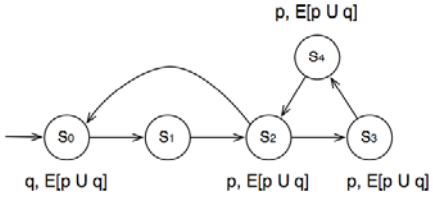
モデル検査アルゴリズム

- **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

$Check(M, P)$
$Check(M, Q)$
$T := \{s \mid Q \in label(s)\}$
for all $s \in T$ do $label(s) := label(s) \cup \{\mathbf{E}[P\,\mathbf{U}\,Q]\}$
while $T \neq \emptyset$ do
  choose $s \in T$
  $T := T - \{s\}$
  for all s' such that $R(s', s)$ do
    if $\mathbf{E}[P\,\mathbf{U}\,Q] \notin label(s')$ and $P \in label(s')$ then
      $label(s') := label(s') \cup \{\mathbf{E}[P\,\mathbf{U}\,Q]\}$
      $T := T \cup \{s'\}$

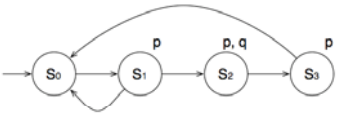
モデル検査アルゴリズム

- **E**[Q_1 **U** Q_2] の場合（続き）



例

- COPY, $s_0 \models \mathbf{E}[\top \mathbf{U} \neg(\neg p \vee \neg q)]$
  - $\mathbf{EF}(p \wedge q)$ 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)}

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