ソフトウェアモデル論（2010年度）
第14回・2010/12/20

桑原 寛明
情報理工学部 情報システム学科

今日の内容

- 前回の復習は次回に
- 並行プログラムとそのモデル検査
- レポートその10について
  - 自然演繹による証明
- 授業アンケート
  - 同時に定期試験について簡単に

定期試験

- 2011/01/24 (Mon)
- 2限（11:00 - 12:00）
- 以下の7題から4題選択して解答
  - 1. 用語説明
  - 2, 3. 有限オートマトンと正規表現
  - 4. チューリング機械
  - 5, 6. 命題論理
  - 7. モデル検査
- 年内に演習問題を公開（予定）

正しい証明木

- 木構造の節点は論理式
  - 根が結論
  - 葉が前提
    - 前提が集合の場合、各要素が少なくとも一回は出現する
      - 複数回出現してもよい
    - 前提に含まれない葉は仮定なので [] で囲まれる
- 葉を除く各節点は、子節点にいずれかの推論規則を適用して得られる論理式
  - 適用した推論規則を記す

自然演繹による証明の戦略 (?)

- 結論を導出できる推論規則は何か
- 前提に適用できる推論規則は何か
- ∨式の導出
  - ∨i が使えないか
  - ∨e が使えないか
  - ∨e にLEMを組み合わせられないか
  - PBCが使えないか
- 推論規則の適用に不足する式を仮定してみる
  - 例：¬P に ¬e を適用するために P を仮定する

練習問題 5.21

1. p∧(q∧r) ⊢ (p∧q)∧r

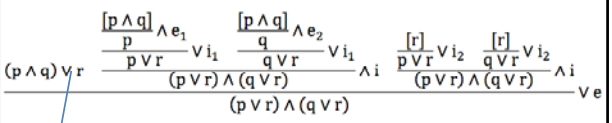
$$\dfrac{\dfrac{\dfrac{p \wedge (q \wedge r)}{p}\wedge e_1 \quad \dfrac{\dfrac{p \wedge (q \wedge r)}{q \wedge r}\wedge e_2}{q}\wedge e_1}{p \wedge q}\wedge i \quad \dfrac{\dfrac{p \wedge (q \wedge r)}{q \wedge r}\wedge e_2}{r}\wedge e_2}{(p \wedge q) \wedge r}\wedge i$$

p∧(q∧r)から r をどう導出するか

∧i で導出できる

練習問題 5.21

3. (p∧q)∨r ⊢ (p∨r)∧(q∨r)

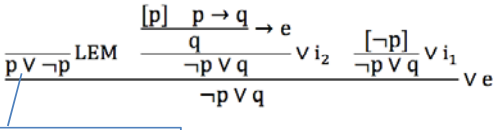


∨eを最後に適用する方法を考えた

※別証明もある

練習問題 5.21

7. p→q ⊢ ¬p∨q



∨eとLEMの組み合わせp→qに→eを適用するために∨¬p

並行プログラム

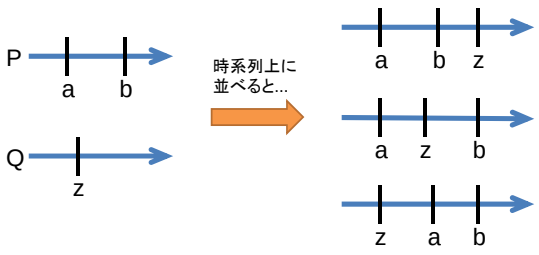
- 複数の計算を（見かけ上）同時に実行するプログラム
  - 並行：論理的に複数の計算を同時に実行
  - 並列：物理的に複数の計算を同時に実行
- 分散システム、クラスタ
- マルチプロセッサ、マルチコア
- マルチタスク、マルチプロセス、マルチスレッド

並行プログラムに固有の難しさ

- 同時に実行される各計算における命令実行のタイミング
  - 非決定性
- 同時に実行される計算同士の相互作用
  - ファイルなどの資源の共有
  - メッセージ通信
- 並行プログラムの動作が正しさを確認することは逐次プログラムに比べはるかに難しい

非決定性

- 並行に実行される各計算の動作を時系列上に並べる方法は一通りではない



非決定性

- 並行プログラムの実行系列は一通りでない
  - 同じ入力に対して同じ実行を行うとは限らない
- どの実行系列が実行されたかは実行が完了して初めてわかる
- 例えば
  - 初めに実行されるaまたはzを非決定的に選択
  - aの次に実行されるbまたはzを非決定的に選択
- すべての可能性を尽くしてテストすることは非常に困難

資源共有と相互排除

- 並行に実行される計算同士でファイルやネットワークなどを共有する
  - 変数の共有もありえる
- 同時使用はできないので排他制御が必要
  - セマフォ、モニタ
  - デッドロック、ライブロック

デッドロック

- P : スキャナ、プリンタの順にロックしてコピー
- Q : プリンタ、スキャナの順にロックしてコピー

1. P がスキャナをロック
2. Q がプリンタをロック
3. ??

メッセージ通信

- 並行実行される複数の計算の間でデータをやり取りする
- 同期通信
  - 送信側と受信側の準備が整ったら通信する
- 非同期通信
  - 送信側は受信側を気にせず送信
  - 受信側はメッセージが来ていれば受信、来ていなければ来るまで待機
- 通信プロトコル

並行プログラムのモデル化

- 並行プログラムにはモデル検査が有効
  - 非決定性による可能性を網羅することができる
- どのようにKripke構造でモデル化するか？
  1. 並行動作する個々の計算（プロセス）をモデル化する
  2. 合成して全体のモデルを得る

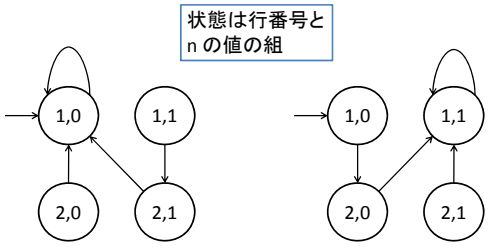
並行プログラムの例

プロセスP

```
1:  if (n == 0) goto 1;
2:  n = 0; goto 1;
```

プロセスQ

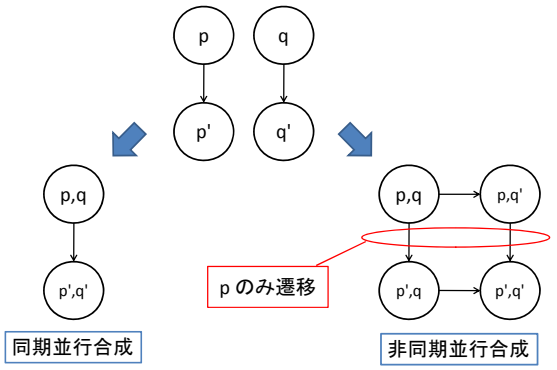
```
1:  if (n == 1) goto 1;
2:  n = 1; goto 1;
```



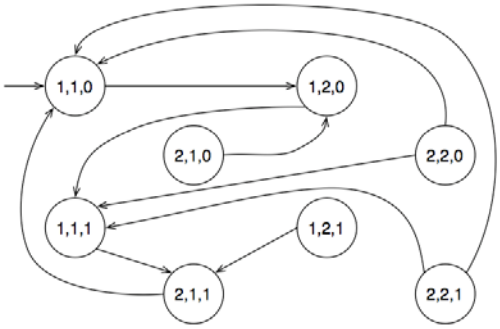
並行合成

- 並行動作するプロセスを一つにまとめること
- 同期並行合成
  - 各プロセスにおける状態遷移が同期して発生する
- 非同期並行合成
  - 各プロセスにおける状態遷移は互いに無関係に発生する
  - 通常は1回の遷移で1つのプロセスのみが状態遷移する

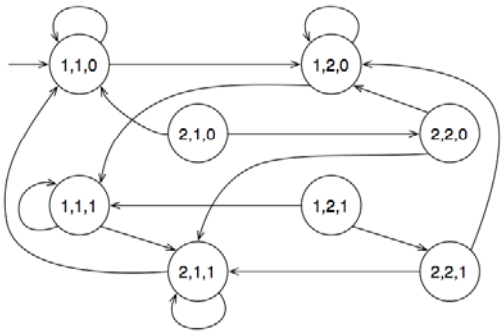
並行合成



同期並行合成の例



非同期並行合成の例



命題の割り当て

- 各状態に対してその状態で成り立つ命題の集合を割り当てる
- 例えば
  - 命題変数
    - P_i : プロセス P の i 行目を実行
    - Q_i : プロセス Q の i 行目を実行
    - N_i : 変数 n の値が i
  - 状態 (p, q, n) に命題集合 $\{ P_p, Q_q, N_n \}$ を割り当てる

調べたい性質の例

- プロセス P とプロセス Q がともに 2 行目を実行することはない
  - 同時に変数 n に代入は危険
- CTLで表現すると **AG**$\neg(P_2 \wedge Q_2)$
  - この例の場合では、状態 (2,2,0) および (2,2,1) に到達しないことと同義

モデル検査の実行

- モデル検査アルゴリズムを実行
- M_s, (1,1,0) $\models$ **AG** $\neg(P_2 \wedge Q_2)$
  - 同期並行合成と命題の割り当てによって得られるKripke構造 M_s
- M_a, (1,1,0) $\models$ **AG** $\neg(P_2 \wedge Q_2)$
  - 非同期並行合成と命題の割り当てによって得られるKripke構造 M_a

NuSMVを使って検査

```
MODULE main
VAR
p : {1, 2};  ← Pの行番号
q : {1, 2};  ← Qの行番号
n : {0, 1};  ← nの値
ASSIGN
init(p) := 1;
next(p) :=
  case
    p = 1 & n = 0 : 1;
    p = 1 & n = 1 : 2;
    p = 2 : 1;
  esac;
init(q) := 1;
next(q) :=
  case
    q = 1 & n = 1 : 1;
    q = 1 & n = 0 : 2;
    q = 2 : 1;
  esac;
init(n) := 0;
next(n) :=
  case
    p = 2 & q = 2 : {0, 1};
    p = 2 & q = 1 : 0;
    p = 1 & q = 2 : 1;
    p = 1 & q = 1 : n;
  esac;
```

論理式 → `SPEC AG !(p = 2 & q = 2);`

NuSMVを使って検査

```
% NuSMV concurrent_loop.smv
...Copyrightなどの表示...
-- specification AG !(p = 2 & q = 2)  is true
```