

ソフトウェアモデル論(2011年度)
第12回・2011/12/16

桑原 寛明
情報理工学部 情報システム学科

練習問題 5.21

1. $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$

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

Λiで導出できる

pΛ(qΛr)からrをどう導出するか

ソフトウェアモデル論(2011/12/16) 2

練習問題 5.21

3. $(p \wedge q) \vee r \vdash (p \vee r) \wedge (q \vee r)$

$$\frac{\frac{(p \wedge q) \vee r}{p \vee r} \vee i_1 \quad \frac{\frac{[p \wedge q] \wedge e_1}{p} \vee i_1 \quad \frac{[p \wedge q] \wedge e_2}{q} \vee i_1}{(p \vee r) \wedge (q \vee r)} \wedge i \quad \frac{\frac{[r] \vee i_2}{p \vee r} \vee i_2 \quad \frac{[r] \vee i_2}{q \vee r} \vee i_2}{(p \vee r) \wedge (q \vee r)} \wedge i}{(p \vee r) \wedge (q \vee r)} \vee e$$

Veを最後に適用する方法を考えた

※別証明もある

ソフトウェアモデル論(2011/12/16) 3

練習問題 5.21

7. $p \rightarrow q \vdash \neg p \vee q$

$$\frac{\frac{p \vee \neg p}{p} \text{LEM} \quad \frac{\frac{[p] \quad p \rightarrow q \rightarrow e}{q} \vee i_2 \quad \frac{[\neg p]}{\neg p \vee q} \vee i_1}{\neg p \vee q} \vee e}{\neg p \vee q}$$

VeとLEMの組み合わせ p→qに→eを適用するためにpV¬p

ソフトウェアモデル論(2011/12/16) 4

自然演繹の健全性 (復習)

- 論理式集合 P_1, \dots, P_n から論理式 Q が導出できるならば Q は P_1, \dots, P_n の論理的帰結である
 - $P_1, \dots, P_n \vdash Q$ ならば $P_1, \dots, P_n \models Q$
 - 証明の結果を信じてよい
- 証明は $P_1, \dots, P_n \vdash Q$ の証明木の高さに関する帰納法による
 - 高さが1の場合を示す
 - 高さが n 未満の場合に成り立つと仮定して n の場合を示す

ソフトウェアモデル論(2011/12/16) 5

自然演繹の完全性 (復習)

- 論理式 Q が論理式集合 P_1, \dots, P_n の論理的帰結ならば P_1, \dots, P_n から Q が導出できる
 - $P_1, \dots, P_n \models Q$ ならば $P_1, \dots, P_n \vdash Q$
 - すべての論理的帰結を証明できる
- 証明は以下の順
 - $P_1, \dots, P_n \models Q$
 - $\Rightarrow \models P_1 \rightarrow (P_2 \rightarrow (\dots (P_n \rightarrow Q) \dots))$
 - $\Rightarrow \vdash P_1 \rightarrow (P_2 \rightarrow (\dots (P_n \rightarrow Q) \dots))$
 - $\Rightarrow P_1, \dots, P_n \vdash Q$

ソフトウェアモデル論(2011/12/16) 6

モデル検査

ソフトウェアモデル論(2011/12/16)

7

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

ソフトウェアモデル論(2011/12/16)

8

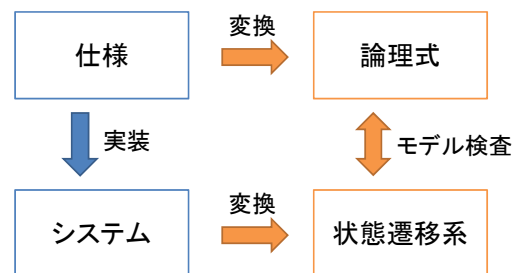
モデル検査の手順

1. 検査対象のシステムを状態遷移系を用いて記述する
 - 対象はプログラムや設計など
 - 「動作する」ものであれば何でも対象になる
 - 状態遷移系としてはKripke構造やオートマトンなど
2. 検査したい性質を論理式を用いて記述する
 - 時相論理や様相論理を用いる
3. 検査アルゴリズムを実行する
 - アルゴリズムを実装した様々なツールがある

ソフトウェアモデル論(2011/12/16)

9

モデル検査の手順



ソフトウェアモデル論(2011/12/16)

10

Kripke構造

- 状態遷移系の一つ
- 直観的には
 - オートマトンから記号を除去し、
 - 状態と状態間の遷移に着目
 - オートマトンの各状態にその状態で真になる命題を追加
 - 「xの値は1である」など
 - 命題は命題論理の範囲内で

ソフトウェアモデル論(2011/12/16)

11

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
 - すべての状態が遷移先を持つ
 - 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

ソフトウェアモデル論(2011/12/16)

12

経路

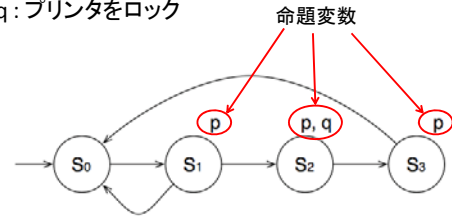
- 遷移関係に従って移り変わる状態の列
- $\sigma = s_0s_1s_2\dots$
 - すべての $i \geq 0$ に対して $R(s_i, s_{i+1})$
- 関係 $R(s, s')$ を「システムの状態が s であれば、次の時刻で状態が s' に変わる」と理解すればよい

ソフトウェアモデル論(2011/12/16)

13

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



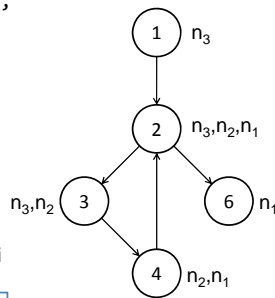
ソフトウェアモデル論(2011/12/16)

14

Kripke構造の例

```

1: n = 3; fact = 1;
2: while (n > 1) {
3:   fact *= n;
4:   n--;
5: }
6: print(fact);
    
```



状態は行番号

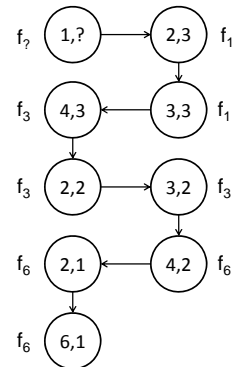
ソフトウェアモデル論(2011/12/16)

15

Kripke構造の例

```

1: n = 3; fact = 1;
2: while (n > 1) {
3:   fact *= n;
4:   n--;
5: }
6: print(fact);
    
```



$f_i : \text{fact} == i$

状態は行番号と変数 n の値の組

ソフトウェアモデル論(2011/12/16)

16

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

ソフトウェアモデル論(2011/12/16)

17

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

ソフトウェアモデル論(2011/12/16)

18

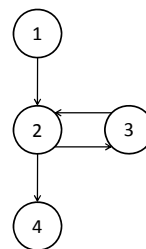
CTL(計算木論理)

- 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

ソフトウェアモデル論(2011/12/16)

19

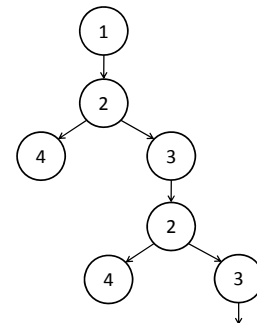
計算木



状態遷移

ソフトウェアモデル論(2011/12/16)

20



計算木