演習コースⅡ 形式手法と仕様記述

「形式手法と仕様記述」 学習チーム報告

Report by “Study” Teams for

“Formal Methods and Specification Description”

主査　　：栗田　太郎（株式会社フェリカネットワークス）
副主査　：石川　冬樹（国立情報学研究所）
研究員　：有賀　一輝（株式会社イクズアネックス）
　　：伊澤　崇史
　　：小田部　健（株式会社小野測器）
　　：羽田　裕（日本電気通信システム株式会社）
　　：平野　純子（ガイオ・テクノロジー株式会社）
　　：水野　徹平（富士ゼロックスアドバンストテクノロジー株式会社）
　　：山村　繁行（株式会社日立製作所）
　　：和田　圭司（株式会社メタテクノ）

報告概要
「形式手法」という語は総称であり，実際のアプローチや具体的な手法・ツールにおいては，目的や特徴，活用方法が非常に多種多様である．このため，自身の課題に応じて，適切なアプローチを学び，活用方法を定めていく必要がある．そこで本演習コースにおいて，希望する参加者はそれぞれの要求，悩み，興味に応じ，特定の手法・ツールを学んだ．それぞれの学習の後半においては，自身の問題意識に対応した活用のための検討を行い，理解を深めるように心がけた．

Abstract　The term “formal methods” refers to just a wide class of approaches. Applied approaches as well as concrete methods and tools have a variety of purposes, features and applications. It is therefore necessary to study a proper approach in response to problems of each, and then to define its applications. This exercise course let the participants to choose to study specific methods and tools, according to their requirements, concerns and interests. Each study is completed to enhance the understanding through investigation of applications to tackles problems that each participant is concerned about.

1.　概要（石川）

仕様をはじめとした開発上流の成果物における品質確保のため，国内産業界でも，形式手法への注目が高まっている．形式手法とは，端的には，文法と意味論が定められた言語を特に上流の成果物において活用するアプローチの総称である.「数理論理学」と呼ばれる理論基盤が与えられていることも定義としてよく言及される．しかし，プログラミング言語やそのコンパイラ，静的解析ツールなども数理論理学に基づいているため，実用上「数理論理学」にこだわる意味は無いであろう．

形式手法については，本演習コースにおける「実施報告」にて概説しているため，馴染みがない読者の方はそちらをご参照いただきたい．

上記にて「総称」という語を用いたように，形式手法と分類される手法・ツールは非常に多種多様である（VDM, B, Event-B, Alloy, SPIN, UPPAAL など）．航空や鉄道などで高信頼性を保証するために使われていることもあれば，一般的な情報システムにおいて厳密化・明確化による手戻り防止のために使われることもある．

そこで本演習コースにおいて，希望する参加者はそれぞれの要求，悩み，興味に応じてグループを作り，特定の手法・ツールを学ぶようにした．ここで，教科書を読んだり，例題を動かしたりしてみるくらいでは，原則を自分のものとし実際に活用していくには不十分である．このため，それぞれの学習の後半においては，自身の問題意識に対応した活用のための検討を行い，理解を深めるように心がけた．

以下の章では，下記の各グループにおける取り組みを報告する．

- VDM（全員共通の学習，2 章）： 構造化プログラミングやオブジェクト指向に基づいてのモデリングや，解釈実行を通したテストなど，一般の開発者にとって馴染みのある記述・検証方法を用いる手法である[Fit10,VDMT]．このため，最初の基礎学習において全員が取り組む対象とした．
- SMV（モデル検査ツール，3 章）：網羅的な状態探索に基づく検証を行うことができる[CVS09,NuS]．活用のための検討としては，半日など短期間でその位置づけなどを学ぶための教育コースを構築した．
- UPPAAL（実時間モデル検査ツール，4 章）： 時間制約も扱いつつ，網羅的な状態探索に基づく検証を行うことができる．活用のための検討としては，勘違いや理解不足が起こりやすい，時相論理による検証項目の記述に対し，多数の基本的な記述例を用意し，学習や活用において参照できるようにした[長谷川 12,UPP]．
- Alloy（有界モデル検査・モデル発見ツール，5 章）： 仕様に対して，それを満たすデータや実行列などの例を，定められた範囲内で網羅的に生成することができる．活用のための検討としては，テストケースの生成ロジックに対する挙動確認を試行した[中島 11, All]．

2. VDM による共通の基礎学習（小田部）

2.1 VDM に触れてみた感想

VDM++言語の学習は特に困難とは思わなかった．その理由としては，覚えるべき文法の量自体が少ないこと，プログラミング言語に近い言語であるためプログラムの設計や実装の経験があれば学習が容易であること，などが挙げられる．

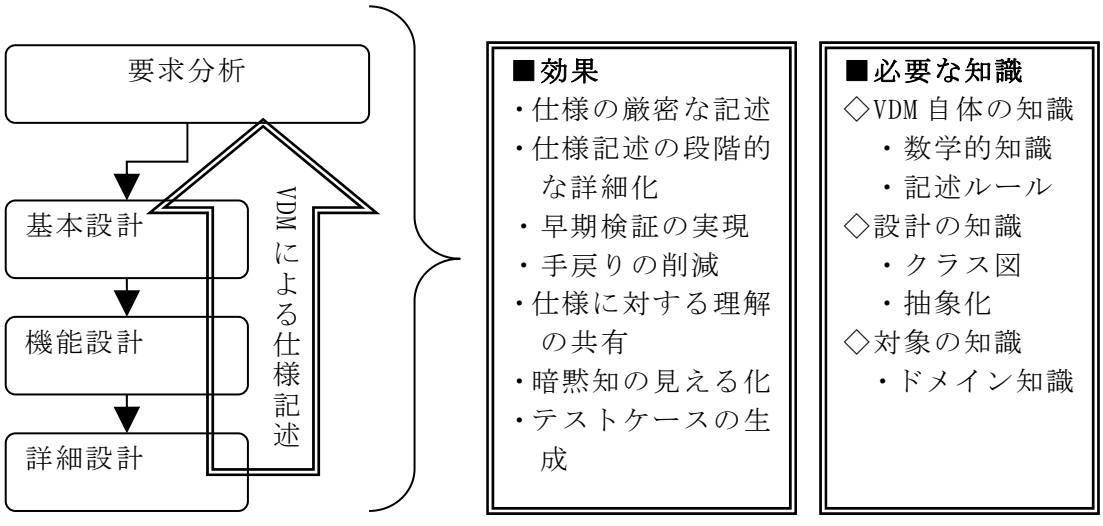

図 1 VDM の効果と必要な知識

2.2 VDM の特徴および効果

- 詳細設計以降で行っていた検証を要求分析の時点で可能とすることで，不具合の早期検出や手戻りの軽減が可能になる（図 1）
- 数学的な論理記述を用いることにより，自然言語より厳密に仕様を記述できる
- 仕様の詳細が未定の状態でも記述可能で，仕様の詳細度に応じた記述が可能である
- 厳密に書かれた成果物を読み取ることで記述対象への理解と技術の習得が促進され，知識の共有と継承へとつながっていく
- 自然言語で記述された成果物を整理し VDM で記述する過程で，関係者間による認識齟齬の解消や，暗黙知の見える化，仕様間の不整合の検出などが期待できる

2.3 VDM を使用する上での課題

- VDM 自体の習得の他に，VDM で記述するモデルを構築するのに必要な知識や技術（形式手法の技術とプログラムの設計技術）の習得が必要になる
- 記述するには言語仕様や記述対象への理解，抽象化の能力，モデル化やイディオムの知識や経験が必要になる
- VDM による記述の際は，形式モデルの目的設定や機能的振る舞いの分析，操作の抽出など，いくつものステップが必要になる

2.4 VDM の難しさのまとめ

私が VDM に触れて感じた仕様記述の難しさは，モデル構築の難しさであると言える．VDM の記述ルールの基礎は理解できたが，仕様書などの成果物から VDM で記述するモデルをどう構築するかについては，普段の業務で基本設計や詳細設計の経験が不足している私にはとても難しかった．VDM の有効活用について考えると，仕様書などの成果物からモデル構築が可能な人材が一人以上いれば，他の関係者は記述を読み取る程度の技術を習得することで VDM の恩恵を受けることができると考える．

3. モデル検査（有賀，羽田，和田）

3.1 モデル検査の特徴と学習の動機・進め方

モデル検査は，検証したいシステムの動作を状態遷移系（オートマトン）で表現し，その状態空間を網羅的に探索することによって，与えられた性質が満たされるか否かを判定する検証手法である．一般にモデル検査では，次のようにしてシステムを検証する．

① システムを状態遷移系として記述する．
② システムに要求する性質（動作仕様，検査項目）を論理式で記述する．
③ 状態遷移系が論理式を満たすことを示す．

モデル検査では，状態遷移系の動きを計算機の上で模倣（真似）して，あらゆる場合をしらみつぶしに調べることによって，状態遷移系が論理式を満たすことを示す．モデルが検査項目を満たさない場合は，反例（どのような状況で満たされないか）が出力される．

モデル検査は，設計段階での適用が可能なため，モデル検査の適用によって，設計のバグを早期に発見し，開発コストの削減を期待できる．これらの効果を狙い，筆者らは業務への適用を検討するため，モデル検査について学習することとした．

学習教材は，産業技術総合研究所システム検証研究センター著の「モデル検査[初級編]-基礎から実践まで 4 日で学べる-」[CVS09]とし，これを読み，例題を解くことによって自分自身の理解を深めた．

3.2. 学習結果の展開（教育コース作成）

学習後，筆者らで業務への適用を検討した結果，適用が望ましいと判断し，モデル検査を筆者らの社内に広めるための教育コースを作成することとした．

モデル検査には，SPIN,SMV,UPPAAL など，いくつかのツールがある．今回の教育コースでは，変数の挙動を一元的に管理できる点，モデル記述言語の構文が簡単であり処理が高速である点を考慮して SMV をターゲットとすることとした[NuS]．

教育コースは，モデル検査を知らない人向けに 4 時間程度で，簡単なシステムの仕様書を読んでモデル検査ができることを目標とした．受講者の習得度合や学習意欲を高めるため，インストラクショナルデザインの方法論である，ガニエの 9 つの教授事象[鈴木 07]と ARCS 動機づけモデル[鈴木 10]を利用した．

まず，ガニエの 9 つの教授事象（図 2）を参考に，内容，到達目標，進め方などを記載した教育コース設計書（図 3）を作成した．また，教授事象 G8 と受講者の声の教育コースへのフィードバックを狙い，アンケート（図 4）を作成した．

ガニエの9つの教授事象		
導入	G1	Gain Attention:学習者の注意を喚起する/情報の受け入れ態勢を作る
	G2	Inform Learners of the Objectives:学習者に目標を知らせる/頭を活性化し，重要な情報に集中させる
	G3	Stimulate Recall of Prior Learning:前提条件を思い出させる/今までに学んだ関連事項を思い出す
情報提示	G4	Present the Stimulus:新しい事項を提示する/何を学ぶかを具体的に知らせる
	G5	Provide Learner Guidance:学習の指針を与える/意味のある形で頭にいれる
学習活動	G6	Elicit Performance:練習の機会をつくる/頭から取り出す練習をする
	G7	Provide Feedback:フィードバックを与える/学習状況をつかみ，弱点を克服する
まとめ	G8	Assess Performance:学習の成果を評価する/成果を確かめ，学習結果を味わう
	G9	Enhance Retention and Transfer:保持と転移を高める/長持ちさせ，応用がきくようにする

図 2 ガニエの 9 つの教授事象

						ガニエ視点（進め方）									ARCS視点（教材）									
章	章の目標	内容	学ぶこと	到達目標	時間	進め方	G1	G2	G3	G4	G5	G6	G7	G8	G9	A1	A2	A3	R1	R2	R3	C1	C2	C3	S1	S2	S3
3	検査項目を記述する論理式の意味を理解し，LTL式の真偽を正確に判断できる．	3.1 LTL式の概要	・パスの定義と性質 ・LTL式のつくり方．与えられた式がLTL式であるかどうかの判定 ・LTL式がいつ真になるかの定義	・パスとLTL式の関係を説明できる． ・与えられた式がLTL式かどうか判定できる．	0:25	講義形式 グループ分け:しない 答えあわせ:する 模範解答の解説:しない 解答に対するコメント:する 課題: 1.LTL式の判定		○	○	○	○	○	○	○			○	○	○	○		○	○		↓	○	○
		3.2 例題「3進カウンタ」	・簡単なシステムに対して，さまざまなLTL式を使って検査する．	さまざまなLTL式の意味を理解し説明できる．	0:17	例題の説明 グループ分け:しない 答えあわせ:する 模範解答の解説:しない 解答に対するコメント:する 課題: 1.否定の検査式とその結果の説明 2.検査項目とLTL式の対応		○	○	○	○	○	○	↑		○	○	○	○	○		○	○		○	↑	○
		3.3 例題「階段びょんぴょん」	与えられたモデル対して，さまざまなLTL式を使って検査する．	簡単な検査項目に対応するLTL式が作れる．	0:18	例題の説明 グループ分け:しない 答えあわせ:しない 模範解答の解説:する 解答に対するコメント:する 課題: 1.検査項目からLTL式の作成，検査		○	○	○	○	○	○	↑		○	○	○	○	○	○	○	○		○	↑	○

図 3 教育コース設計書


モデル検査(SMV)コース アンケート （20xx. x. xx 開催）

アンケートにご協力をお願い致します。（□にレ印をご記入ください。） 裏面もありますのでご注意ください！

01. プロフィールをお聞かせください。

年 齢
□ 20 代 □ 30 代 □ 40 代 □ 50 代 □ 60 代

役 職
□ 担当 □ 主任 □ 課長級 □ 部長級 □ 事業部長以上

職 種
□ ソフトウェア開発 □ ハードウェア開発 □ 営業・販促 □ 品質保証 □ その他（ ） □ 全般 □ スタッフ

02. 総合的な感想をお聞かせください。

□ 大変良かった □ 良かった □ どちらともいえない □ よくなかった

03. 他の方に受講を勧めたいと思いますか？

□ 受講を勧める □ どちらともいえない □ 受講を勧めない

受講すると良いと思う方を選択してください（複数選択可）。

□ 部下（下位者） □ 同僚 □ 上司（上位者） □ その他（ ）

04. 講義時間について感想をお聞かせください。

□ 長い □ ちょうど良い □ 短い

05. 演習時間について感想をお聞かせください。

□ 長い □ ちょうど良い □ 短い

07. モデル検査(SMV)の考え方について理解できましたか？

□ 十分理解できた □ だいたい理解できた □ 一部理解できなかった □ 理解できなかった

一部理解できなかった方に伺います。理解できなかったのは、どのような部分でしょうか？

08. モデル検査(SMV)を利用することで、貴方の組織の品質向上が実現できると思いますか？

□ 品質向上できそう □ 品質向上の参考にはなるかどうかはわからない □ 参考にはならそうだが、品質向上 □ 参考になるかどうかわからない

図 4 アンケート

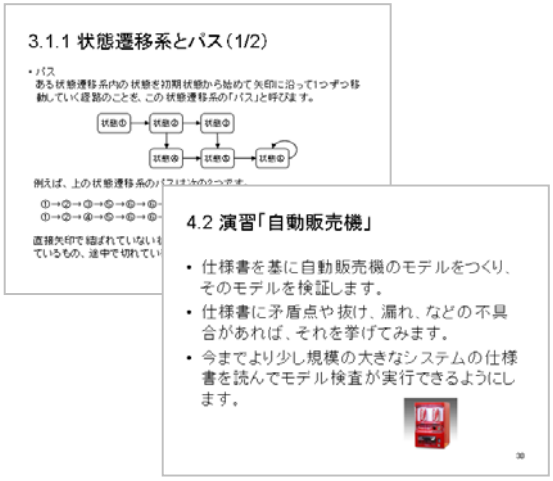

図 5 教材

次に，ARCS 動機づけモデルを参考に教材（図 5）を作成した．ARCS 動機づけモデルとは，アメリカの教育工学者ジョン・M・ケラー（John M.Keller）が提唱した，受講者の学習意欲を高めるためのモデルである．ケラーは，学習者が「なぜやる気がでないのか」を 4 つの側面からチェックして，それに応じて教育内容を検討するのが効果的であるとした．ARCS は 4 つの側面である，注意（Attention），関連性（Relevance），自信（Confidence），満足感（Satisfaction）の頭文字をとったものである．筆者らは，ARCS 動機づけモデルの 4 つの側面の一つ一つをさらに 3 つに分けた，12 個の項目（知覚的喚起,探究心の喚起，変化性，親しみやすさ，目的指向性，動機との一致，学習要求，成功の機会，コントロールの個人化，自然な結果，肯定的な結果，公平さ）を教材の検討項目として利用した．

3.3. まとめ

今回の学習では，モデル検査を学習し業務への適用を検討した．さらに筆者らの社内に広めるためモデル検査の教育コースを作成した．まず，教育コース設計書を作成し，それを元にスライド 80 枚程度の教材，アンケートを作成した．

今後は，教育コース設計書，教材，アンケートを使用して教育を行い，アンケート回答や受講者の反応を教育コース設計書にフィードバックし，教育コースの適正化を図る．また，受講者とモデル検査の適用について議論を深め，設計のバグを早期に発見し，開発コストの削減を目指したい．

4. 実時間モデル検査（平野）

4.1 時間に関する要件の検証

並列に動作する複数のプログラムが共通リソースに書き込みを行うような状況で動作する場合，デッドロック等のタイミングにまつわる様々な問題が発生し得る．また，リアルタイムシステムでは，要求された処理が所定の時間内に終わるかどうかも重要である．これらの問題をレビューやテストで検出することは非常に難しい．このような問題についての対策として効果が期待できるのが UPPAAL による性能検証である．

UPPAAL はリアルタイムシステムのモデリング，シミュレーションおよび検証を行うためのツールで，スウェーデンのウプサラ大学（Uppsala University）情報技術学部とデンマークのオールボー大学（Aalborg University）計算機科学基礎研究科とが共同で開発したものである[長谷川 12,UPP]．UPPAAL では，システムモデルを時間の概念を含んだ時間オートマトンでモデル化し，検証したい性質を時相論理 CTL(Computation Tree Logic)の式と

して記述する．すると，記述したモデルが検証したい性質を満たすかどうかを検証することができる．

UPPAAL はエディタ，シミュレータ，ベリファイアから構成されている．

エディタでは，GUI 操作によって時間オートマトンを記述し，モデルを作成する．時間オートマトンの記述については，特別なプログラミング言語を必要とせず，簡単にモデルを作れる．

シミュレータでは，メッセージ・シーケンス図によって，シミュレーションの進行状況を表示できる．

ベリファイア（検証器）では，エディタで作成したモデルについて，ここで記述した検証式の真偽を検査する．

4.2 UPPAAL のシステムモデルと検証式

UPPAAL の操作は前述の通り，非常に容易になっているが，UPPAAL を実際の開発に適用とするためには多くのハードルがある．その中の大きなハードルの１つとして，時間オートマトンと時相論理式を理解することが挙げられる．

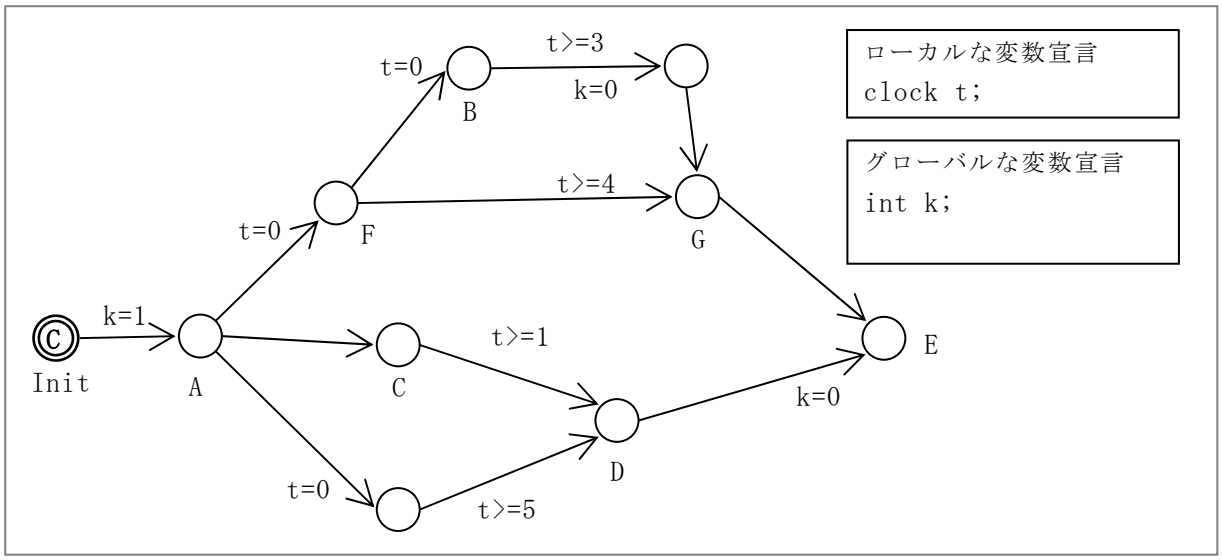

図 6 プロセス P のモデル

ここで，図 6 のモデルで次の検証式が成り立つかどうかを考えてみる．

E<> P.B　（あるパスでいつかは B に到達する）　・・・①
E<> P.C　（あるパスでいつかは C に到達する）　・・・②
A<> P.E　（全てのパスでいつかは E に到達する）　・・・③
A<> k==0　（全てのパスでいつかは k の値が 0 になる）　・・・④

一見すると，①～③は成り立って，④は Init→A→F→G→E のパスでは k=0 の代入がないので成り立たないように見える．しかし，UPPAAL のベリファイア（検証器）で検査すると①と②と④は真で，③は偽であるという結果になる．③については，A に留まったままのパスもあるという反例が挙げられる．④については，すべてのパスは Init からスタートし，Init においては k の値は 0 なので，検証式は真になる．⑤のように Init 以外のロケーションにおいてという条件を付加すると，正しい検証が行える．

A<> ((not P.Init) and k==0)　・・・⑤

このように，オートマトンと時相論理式の意味論を正しく理解していないと，勘違いや間違いによって正しい検証が行えない．そこで，本来はツールを使って検証する必要がない，極小規模なモデルを作成し，その性質を検証式で表現することを試みた．作成したモ

デルが，意図通りのものとなっているかを検証することにより，勘違いや間違いの要素を排除することが今回の目的である．

まず，UPPAAL の学習における参考書とした『UPPAAL による性能モデル検証』の「第 2 章 UPPAAL のシステムモデルと検証式」[長谷川 12]から文法的な要素を抽出し，作成するモデルのテーマを整理した．

表 1

テーマ	モデルに含める要素
遷移	条件なし，不変式あり，ガードあり，アージェントブロードキャストチャネルを使う遷移，ダミープロセス，コミットロケーション，アージェントロケーション
分岐とループ	条件なしの分岐，条件付きの分岐，時間制約のあるループ，変数の条件を付けたループ
一対一同期通信	同期と値の受け渡し，コミットロケーションと同期通信，アージェントロケーションと同期通信，アージャントチャネルによる同期通信
ブロードキャスト通信	同期と値の受け渡し

また，同様に検証に使用する検証式の形式も抽出した．

- A[] p：すべてのパスで常に条件 p が成り立つ（安全特性と呼ぶ）
- A<> p：すべてのパスでいつかは条件 p を満たす状態に到達する（活性特性と呼ぶ）
- E[] p：常に条件 p が成り立つパスが存在する
- E<> p：いつかは条件 p を満たす状態に到達するパスが存在する
- A[] not deadlock：デッドロックに陥らない
- p --> q：条件 p が成立するならば，条件 q が成り立つ状態にいつか必ず到達する
- P.X：プロセス P が状態 X に到達する
- A[] p imply q：条件 p が成立するならば，条件 q は常に成立する

例えば，遷移のテーマで“条件なし”のモデルの場合は，図 7 のようなプロセス P について，表 2 の検証式の真偽を UPPAAL で検査する．このとき，時間の単位は秒とする．

図 7 条件なしの遷移のプロセス P

表 2

検証式	結果	備考
A[] P.X	：偽	
A<> P.Y	：偽	UPPAAL ではずっと X に留まったままということがある
E[] not P.Y	：真	Y に到達しないパスもある．
E<> P.Y	：真	あるパスで，いつかは Y に到達する
E<> P.X and P.Y	：偽	t==0 で Y へ遷移しても，同時に２つのロケーションにあるとは判定しない
A[] P.X or P.Y	：真	
A[] P.t==0 imply P.X	：偽	遷移には時間がかからないので，0 秒の時に Y に到達するパスもある
P.t==0 --> P.X	：偽	

不変式，ガードで時間制約を付加したパターンでは，最小時間や最大時間についても検査し，ループを含むパターンではデッドロックしないことも検証対象とする．

4.3 結果とまとめ

参考書を読むだけでは，オートマトンと時相論理式の意味を正しく理解できていなかったことが分かった．いろいろなサンプルモデルを作成することにより，ツールの制限などが分かる効果もあった．UPPAAL は非常に便利で使いやすいツールである反面，勘違いしたままモデルを作成してしまうと正しい結果が得られない危険があるので，実践的なモデルを作成する前に今回のような実習を行うことは有効である．

5.Alloy（水野）

5.1 Alloy について

Alloy は形式記述言語のひとつでモデル検査機能を持つツール Alloy Analyzer が提供されている[中島 11,All]．一般的なモデル検査ツールである SPIN などが，仕様の正しさに対する網羅的な検査をするのに対し，Alloy は特定の範囲に絞ってとりうるパターンを網羅的に列挙するする．そのため正しさの証明はできないが，テストで行うよりはるかに膨大なパターンを検証できる．また，仕様を短い記述で柔軟に表現することができ，即座に結果を視覚的にフィードバックすることができる．

実際に記述してみると，仕様を部分的に表現した段階で検査し，不都合な結果があれば仕様を追加していくといったやり方を反復する書き方が可能である．この試行錯誤の方法は一般的な開発者にはなじみやすく，また仕様の不具合を見つける手法を，記述しながら練習することができる．

5.2 Alloy を利用する目的と方法

現在開発に取り組んでいるソフトウェアの一部の仕様について，顧客の要求に対して十分な仕様になっているかどうかを検証する必要があった．具体的には以下のような仕様である．機能の目的は入力された仕様データを元に，テストスクリプトを出力することである．ただし，ここではモデル化するため仕様を単純化している．

1. 空でない複数の集合 A{a1,a2,a3},B{b1,b2,b3},C{c1,c2,c3}がある．集合 A,B,C から要素を 1 つずつ取り出して値の組 T{a1,b3,c1}を複数作成する．同じ要素を何回でも使ってよい．
2. 集合 A,B,C 間の各要素に禁則関係を設定できる．{a1,b2}を禁則関係とすると同じ値の組の中に a1 と b2 は同時に存在できない．
3. ある要素が他の集合のすべての要素と禁則のとき．例えば{a1,b1},{a1,b2},{a1,b3}の禁則が設定されているとき T の要素に a1 が含まれる場合，集合 B の要素はくわえられない．この場合は null を使い {a1,null,c1(あるいは c2,c3)} といった値の組 T が作られる．

第 1 版の仕様は上記のようであった．しかし，禁則を以下のように設定した場合に問題が生じた．

{a1,b1},{a1,b2},{a1,b3},{b1,c1},{b2,c2},{b3,c3}

この場合 2 つの解釈が発生する．1 つは a1 が選択された場合，b1,b2,b3 いずれも選択できないので，間接的に禁則{a1,c1},{a1,c2},{a1,c3}が設定されているものとして，T は{a1,null,null}となる解釈．もうひとつは同様に a1 が選択された場合，B の要素選択時に null が選択できるため，T は{a1,null,c2}となる解釈．これを判断するにはユーザーに追加の情報を入力してもらう必要がある．そこで以下の仕様を追加した．

4. 禁則の一つ目の要素に{null,b1}のように，null を使えるようにする．

上記の例の場合，禁則{a1,b1},{a1,b2},{a1,b3},{b1,c1},{b2,c2},{b3,c3}に加えて，Bがnullになった場合の禁則を{null(B),c1},{null(B),c2},{null(B),c3}と設定すると，Tは{a1,null,null}となり，設定しない場合は{a1,null,c1(もしくは c2,c3)}になる．これで入力データの不足により，結果がわかれてしまう問題は解決した．しかし，他にもこういったパターンで不足している仕様が無いだろうか．それを検証するため，Alloy で仕様を記述した(表 3)．

表 3 Alloy による仕様記述

```
1  abstract sig A{}
2  one sig a1 extends A {}
3  one sig a2 extends A {}
4  one sig a3 extends A {}
5
6  abstract sig B {}
7  one sig b1 extends B {}
8  one sig b2 extends B {}
9  one sig b3 extends B {}
10
11 abstract sig C{}
12 one sig c1 extends C {}
13 one sig c2 extends C {}
14 one sig c3 extends C {}
15
16 sig Test{
17 aa: one A,
18 bb: one B,
19 cc:   one C
20 }
21 {
22 not (aa = a1 and bb = b2)
23 not (aa = a1 and bb = b2)
24 not (aa = a1 and bb = b3)
25 }
26 pred show{
27 }
28 run show
```

仕様の 1.2.を記述し，ツールを用いて様々な例を図示させることができた(図 8)．しかし，仕様 3.4.の記述までは至らなかった．しかし，パターンの列挙する様を見ることで，目的の達成に手ごたえを感じることはできた．

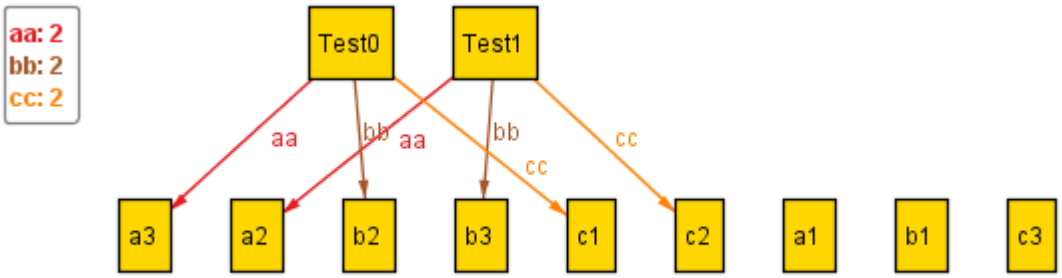

図 8 Alloy Analyzer におけるパターン図示の一例

5.3 Alloy まとめ

今回残念ながら目標の達成に至らなかった．さまざまな原因があると考えられるが，そのうちの一つに Alloy での記述方法が，C 言語をはじめとする一般的な手続き型プログラミングの手法とだいぶ異なっていることがあげられる．そのため，プログラミングに慣れた開発者でも，時間ができたときに教科書を参照しながら記述していくスタイルだとなかなか進まない．集中して習得する必要があると考えられる．ただ，今回のように仕様を部分的に記述した中途半端な状態でもパターンを列挙できるので，少しずつ不足する仕様を検証しながら追加していく作業には向いていると考えられる．

今後も引き続き仕様の記述と検証を Alloy にて行い，当初の目的を達成したいと考えている．

6. おわりに（石川）

各参加者は，自分自身の問題意識・動機に基づきそれぞれ手法やツールを学んだ．共通基礎としての VDM の学習を行った後の非常に短い期間の取り組みであった．しかし，実際に活用に向けた模索を行うことにより，効果や困難さをより強く実感できたのではないかと思われる．今年度の経験に基づき，より深い理解や活用に向けた取り組みを継続的に行って欲しい．

参考文献

[Fit10] ジョン・フィッツジェラルド，ピーター・ゴルム・ラーセン，ポール・マッカージー，ニコ・プラット，マーセル・バーホフ，酒匂 寛（訳）．VDM++によるオブジェクト指向システムの高品質設計と検証．翔泳社（2010）

[VDMT] SCSK 株式会社．VDM information web site．http://www.vdmtools.jp/

[CVS09] 産業技術総合研究所システム検証研究センター．モデル検査 初級編―基礎から実践まで 4 日で学べる（CVS 教程）．ナノオプトメディア（2009）

[NuS] NuSMV home page．http://nusmv.fbk.eu/

[鈴木 07] R.M.ガニエ，W.W.ウェイジャー，K.C.ゴラス，J.M.ケラー，鈴木 克明（監訳），岩崎 信（監訳）．インストラクショナルデザインの原理．北大路書房（2007）

[鈴木 10] J.M.ケラー，鈴木 克明（監訳）．学習意欲をデザインする―ARCS モデルによるインストラクショナルデザイン―．北大路書房（2010）

[長谷川 12] 長谷川 哲夫，磯部 祥尚，田原 康之，大須賀 昭彦（監修）．UPPAAL による性能モデル検証（トップエスイー実践講座）．近代科学社（2012）

[UPP] UPPAAL．http://www.uppaal.org/

[中島 11] Daniel Jackson，中島 震（監訳），今井 健男（訳），酒井 政裕（訳），遠藤 侑介（訳），片岡 欣夫（訳）．抽象によるソフトウェア設計－Alloy ではじめる形式手法－．オーム社（2011）

[All] Alloy Community．http://alloy.mit.edu/community/