組み込みソフトへの数理的アプローチ

第6回

時相論理の導入：CTLとLTL
── NuSMVでステート・マシンと実行系列を検査する

藤倉 俊幸

1 様相論理とそれを取り巻く状況

前回（2007年2月号，p.151）の様相論理は，美しい話だった．特に図11の世界間のアクセス関係を追加していくと最終的にKT5の同値関係になるところが良い．

この図を見ていると，数学的な同値関係が随分きつい関係であることが分かってくる．普段「同じ」という概念を使っているが，自然言語で使う「同じ」という概念がいろいろなものに分解できることなども見えてくる．しかし，だから何の役に立つのかを知らないとだめなのかと言われると，確かに知っていれば何かの役に立つかもしれないが，決定的に重要だとは言い切れないかもしれない．

たとえば，ポインタのつながり方も様相として表せる．ポインタのつながり方をクリプキ・モデルと考えて，次を指しているプログラム変数をそこで成立する命題とする．そして，ポインタが「指す」ということを様相と捉えるのである．これは，ヒープの管理やポインタを使ったプログラムの正しさを調べることなどに応用できる．

一方で，機能安全規格（IEC 61508）なるものがある．この中では，原子力発電所などで事故が起こるとただでは済まないシステムを開発する場合は，形式手法を使えという．ただ，IEC 61508をJIS C 0508-3に翻訳する際に，formal methodを「形式手法」ではなく「公式手法」と訳してしまったので，しばらく何のことか分からない状態が続いていた．しかし幸いなことに，最近は形式手法という用語も認識されるようになってきている．形式手法にもいろいろあるが，機能安全規格の中で使えと言っている手法の中には様相論理の一派の時相論理（Temporal Logic）も含まれている．

規格などに美しさを感じる人もいるかもしれないし，認証をすぐ取りたがる人もいる．時相論理はそういう人にも役に立つ．ITRONを作っていたころは，皆で日本の組み込み業界を良くしようという雰囲気があったが，最近は，ビジネス・チャンスとみる渡世人も大分増えてきた．すると，極端に言えば，美しさを求める人とお金を求める人がいっしょに仕事をする事態が発生する．

硬軟・貴賤を取り混ぜながら世の中は動いていくけれども，良い仕事をしたいと思っているまじめなエンジニアが，今日の仕事をこなすために時相論理を知らないとだめということはない．もっと先に知るべきことがあるという意見は，確かに説得力がある．この連載も，どんどん論理のつぼにはまって，このままでは抜け出せなくなりそうなので，心配してくれる人もいる．実は前回の図1の「この道を進むとどうなるか」を説明する論理系がまだ完成していなかったので，今回はそれを完成させる予定だった．しかし，あまりにもマニアックになりすぎるという批判も出ているので，この話題はやめにして，役に立つ方向にシフトしようと思う．そこで今回は，時相論理から話を始める．

2 CTL ──計算木論理とは何か？

● CTLは分岐を直接表現できる

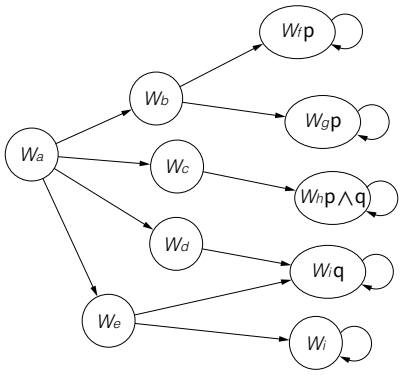
モデル検査で仕様を記述するために使われる時相論理には，二つの異なる論理系が使われる．一つはCTL（Computation Tree Logic：計算木論理）で，もう一つはLTL（Linear Temporal Logic：線形時相論理）である．このほかにもあるが，手軽に使えるツールがあるのはCTLとLTLである．たとえば，LTLなら以前の連載で紹介したSPINとLTSA．CTLならSMV．CTLとLTLの双方を使用することができるツールとして，今回紹介するNuSMVがある．これらの論理系の違いについては参考文献(1)や(2)が参考になる．

プログラムの実行は，`if`文などで制御が分岐するのでいろいろな動きをする．その動きを直接表現できるから，CTLのほうが分かりやすいという人もいる．一方，LTLには保存定理というものがある．これは，要求段階で証明したことが設計の過程でステート・マシンを非決定性から決定性に変更しても保存されるという便利な性質である．非決定性のステート・マシンを扱わなければならない要求検証ではLTLを使用し，設計がある程度進んだ段階や，ソース・コードをリバースして検証する場合などはCTLが良いのではないかと思う．

● NuSMVでCTLを使ってみる

たとえば，前回の図1の犬に出会うか，羊に出会うかのような

KEYWORD ── 様相論理，機能安全，IEC 61508，形式手法，CTL，計算木論理，LTL，線形時相論理，NuSMV




```
MODULE main
VAR
  world : { a,b,c,d,e,f,g,h,i,j
};
  p : boolean;
  q : boolean;

ASSIGN
  init(world) := a;
  next(world) := case
    world = a : { b, c, d, e} ;
    world = b : {f, g} ;
    world = c : h ;
    world = d : i ;
    world = e : { i, j} ;
    1 : world;
    esac;

  init(p) := FALSE;
  next(p) := case
    world in { f, g} : TRUE;
    1: FALSE;
    esac;

  init(q) := FALSE;
  next(q) := case
    world in {h, i} : TRUE;
    1: FALSE;
    esac;
```

変数定義
犬に会う
羊に会う
変数worldの初期値をaとする
変数の次の状態

図1
前回の問題をNuSMVモデルに変換したもの

分岐をしている道の構造も含めて考えるにはCTLが良い．CTLを使えるNuSMVでモデル化すると**図1**のようになる．

前回の図6に示した遷移系を利用してモデル化した．1行目のMODULE mainはNuSMVのオマジナイのようなものだ．2行目～5行目で使用する変数を定義している．

変数worldは，どの世界にいるのかを表すために利用する．この遷移系の世界はW_aからW_jまであったので，それをa～jの列挙型で定義する．変数pは「犬に会う」という命題で，W_fとW_g，W_hで真になる．変数qは「羊に会う」という命題でW_hとW_iで真になる．この遷移系の状態はこれら三つの変数値によって定義される．

6行目から最後までは，状態の変化のしかたを表す．init()は，それぞれの変数の初期値を定義する．最初の世界はW_aなのでworldの初期値はaになる．また，W_aではpとqは偽になる．

next()はそれぞれの変数の次の状態の値を定義する．8行目～15行目のnext(world)は，世界のつながり関係をそのまま表現している．W_aの次の世界は，W_bかW_c，W_d，W_eなので9行目のように表現する．集合型{b，c，d，e}で書いておくとNuSMVがこの中から適当に次の世界を選んでくれる．case文は，:（コロン）の前に書いた論理式が真のとき，:の後ろの式を評価してその結果を変数に代入する．NuSMVはcase文の上から順に真になる条件を探して，真になる条件が存在しないとエラーにする．それで，最後に1:を追加してかならず真になる分岐を用意するのが一般的である．C言語のdefaultのようなものである．ちなみに，複数の条件が真になるといちばん最初のもの（上のもの）が使われる．

● NuSMVのインストールと実行

NuSMVのインストールは非常に簡単である．NuSMVのWebサイト(3)からWindows版のインストーラをダウンロードしてダブルクリックすれば良い．現在のバージョンはv2.4である．NuSMVにはGUIがないので，コマンド・プロンプトとテキスト・エディタを使用する．**図1**のモデルをテキスト・ファイルに打ち込んでNuSMVにかけると，

```
DOS プロンプト > nusmv example.smv
```

コピーライトなどが表示されるだけで，一瞬で終了してしまう．モデル検査は検査式がないとシンタックス・チェック程度しかしてくれない．LTSAは合成したステート・マシンを見せてくれるが，NuSMVでは何も起こらない．

そこで，検査式を記述する．NuSMV用の検査式は**図2**に示した記号を使用して書く．たとえば，「かならずそのうち犬に会う」のであればAF pとなる．NuSMVに対しては，モデル・ファイルの中で，

```
SPEC AF p
```

と指定する．指定する場所は**図1**のモデル定義の後ろである．SPECはCTL式を指定するキーワードであり，実行結果は**図3**と**図4**のようになる．検査は初期状態で検査式が成立するかどうかを見ることで行われる．W_aからW_dまたはW_eに進んでしまうと犬に会うことはないのでAF pは成立しない．つまり，すべての道でF pが成立しない．**図3**の実行例では，$W_a \rightarrow W_e \rightarrow$

NuSMV記号	数学記号	
A	∀	かならず（すべての道で）
E	∃	可能性がある（そんな道もある） 少なくとも一つ道がある
X	○	次に
F	◇ < >	そのうち
G	□ [ ]	ずっと

時間の経過に関する様相

図2　NuSMVの時相論理記号