

モデル検査 Web サービスの SMV ファイルについて

改定履歴

2011-08-18	表紙	タイトル名を修正。
	—	“64bit 版モデル検査器”の表記を“モデル検査 Web サービス”に変更。
	表 4-1	その他の演算子として, union と in を追加。 論理演算子の + を削除。
	表 4-2	union と in, 及び時相論理演算子を追加したほか, 誤記を修正。

目次

1.	SMV ファイルについて	1
2.	モデル記述部	2
2.1	変数宣言部	3
2.2	状態遷移系記述部	6
2.3	FAIRNESS 条件記述部	10
3.	検査項目記述部.....	11
4.	使用可能な演算子	12

1. SMV ファイルについて

SMV ファイルは「XXXX.smv」のように、任意の名称 (XXXX) に小文字の拡張子「.smv」を付けて作成する。名称には日本語は使用できない。SMV ファイルの例を図 1-1 に示す。

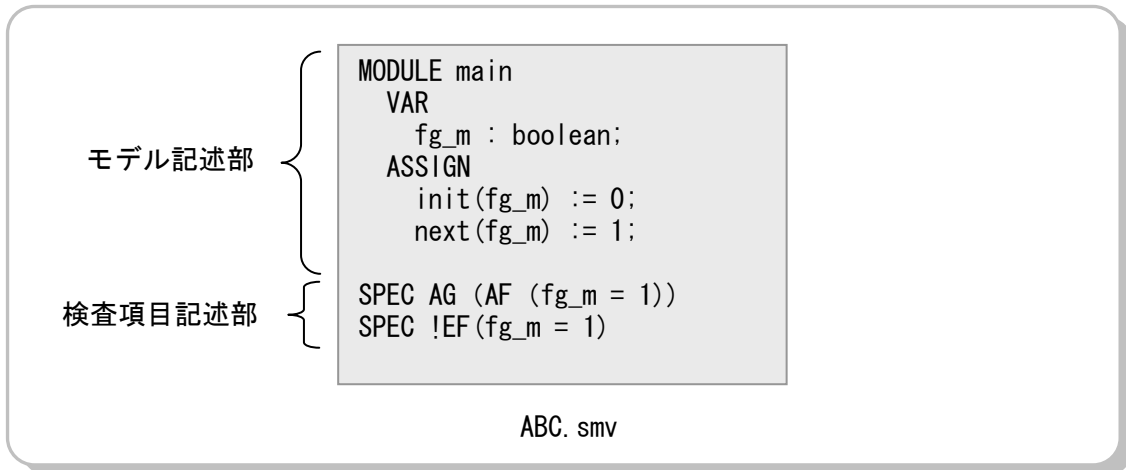


図 1-1 モデル検査支援ソフトウェア構成図

モデル検査 Web サービスでは1つの SMV ファイルのみ入力することができる。また、SMV ファイルには1つのメインモジュール (MODULE main) のみ記述することができる。サブモジュールには対応していない。

メインモジュールはモデル記述部と検査項目記述部とに分けられる。

コメント文について。

「--」以降の一行はコメント文となる。

「/*」と「*/」で囲まれた範囲は全てコメント文となる。

コメント文の例を図 1-2 に示す。

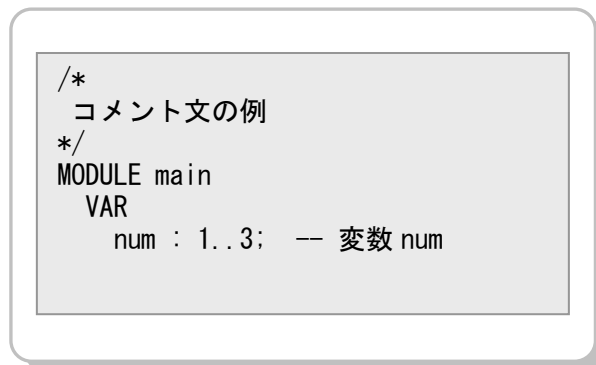


図 1-2 コメント文の例

2. モデル記述部

モジュールは1つの宣言部と2つの記述部で構成される。

- VAR (変数宣言部) : 使用する変数を宣言する。
- ASSIGN (状態遷移系記述部) : 状態遷移系を記述する。
- FAIRNESS (FAIRNESS 条件記述部) : FAIRNESS 条件を記述する。

FAIRNESS (FAIRNESS 条件記述部) は不要であれば省略可能である。

2.1 変数宣言部

変数宣言部の始まりには「VAR」と記述する。変数は変数名と変数の型を使って宣言する。文末には「; (セミコロン)」を付ける。

```
VAR
  変数 A : 変数 A の型;
  変数 B : 変数 B の型;
```

変数名は以下の規則に従わなければならない。

表 2-1 変数名の規則

NO	内容
1	使える文字は半角の「A~Z」「a~z」「0~9」「_ (アンダースコア)」のみ。
2	大文字と小文字は区別される。
3	途中に空白あるいは改行をいれることはできない。
4	先頭はアルファベット又はアンダースコアでなければならない。数字は使えない。
5	以下の名前は宣言できない。 <ul style="list-style-type: none"> ・ 記述部や宣言部を表す語 (「MODULE」「VAR」「DEFINE」「ASSIGN」「FAIRNESS」「SPEC」等) ・ 予約語 (「init」「next」「case」「esac」「union」「array」「of」「boolean」等) ・ 演算子 (「mod」「in」「union」) ・ その他 (「running」) ・ CTL 式の単独一文字「A、E、G、X、F、U」 ・ CTL 式の二文字「AG、AX、AF、EG、EX、EF」 ・ 単独の大文字一文字「H、O、S、T、V、Y、Z」

NO.5 については、NuSMV で禁止されているため、モデル検査 Web サービスでも禁止語としたものが含まれている。

変数の型には「boolean 型」「整数型」「シンボルセット型」の3種類がある。

(1) boolean 型

boolean 型の変数は真 (1) と偽 (0) のどちらかの値に遷移する変数である。以下の要領で宣言する。

```
VAR
  変数名 : boolean;
```

例) flag : boolean;

変数 flag は 1 または 0 の 2 つの値に遷移する変数である。

(2) 整数型

整数型の変数は使用する整数の範囲だけ宣言し、宣言された範囲内のいずれかの整数値に遷移する。小数値は扱えない。以下の要領で宣言する。

```
VAR
  変数名 : 始まりの値..終わりの値;
```

例 1) num : 0..7;

変数 num は 0, 1, 2, 3, 4, 5, 6, 7 のいずれかの値に遷移する。

例 2) num : 5..2;

変数 num は 5, 4, 3, 2 のいずれかの値に遷移する。

例 3) num : -2..2;

変数 num は -2, -1, 0, 1, 2 のいずれかの値に遷移する。

変数の範囲を指定する際に四則演算子は使用できない。図 2-1 のような記述は構文エラーとなる。

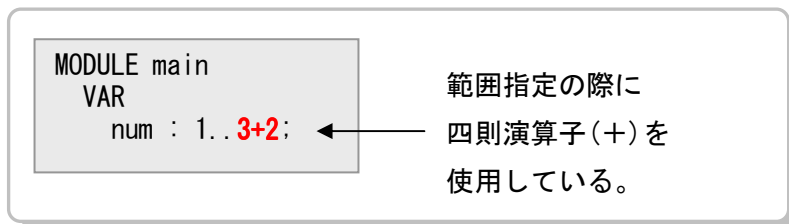


図 2-1 範囲を指定する際の構文エラーの例

(3) シンボルセット型

シンボルセット型の変数は「{ } (中括弧)」で囲まれたシンボルのうちのいずれかのシンボルに遷移する。以下の要領で宣言する。

VAR**変数名 : {シンボル 1, シンボル 2, …, シンボル N};**

例 1) Switch : {ON, OFF};

変数 Switch は ON, OFF のいずれかのシンボルに遷移する。

例 2) cpu : {wait, busy, ready};

変数 cpu は wait, busy, ready のいずれかのシンボルに遷移する。

2.2 状態遷移系記述部

状態遷移系記述部の始まりには「ASSIGN」と記述する。

ASSIGN

```
init(変数 A) := 変数 A の初期値;
next(変数 A) := 変数 A の状態遷移系
```

(1) 初期値の指定 (init 文)

以下のように記述する。文末には「; (セミコロン)」を付ける。

```
init(変数) := 初期値;
```

init 文の記述例を以下に示す。

例 1) `init(flag) := 0;`

変数 `flag` の初期値は 0 である。

例 2) `init(cpu) := wait;`

変数 `cpu` の初期値は `wait` である。

例 3) `init(num) := {1, 2};` (非決定性)

変数 `num` の初期値は値 1 あるいは値 2 のいずれかである。

例 4) `init(cpu) := {wait, busy};`

変数 `cpu` の初期値はシンボル `wait`、シンボル `busy` のいずれかである。

例 5) `init(cnt) := {1..5};`

変数 `cnt` の初期値は値 1 から値 5 のうちのいずれかの値である。

(2) 次の状態での値の指定 (next 文)

初期状態より後の状態での変数の値を指定する。next 文を用いる。

(a) 遷移条件が無い場合

初期値から次の状態に遷移する時に遷移条件が 1 つも無い場合は、以下のように記述する。文末には「; (セミコロン)」を付ける。

```
next(変数) := 値;
```

next 文の記述例を以下に示す。

例 1) next(flag) := 1;

変数 flag の次の値は値 1 である。

例 2) next(cnt1) := cnt2;

変数 cnt1 の次の値は、変数 cnt2 の持っていた値である。

(変数 cnt1 の N + 1 番目の値は、変数 cnt2 の N 番目の値である)

(b) 遷移条件がある場合

case 文を用いて場合分けを行う。以下のように記述する。

```
next(変数) := case
  条件 1   : 値 1;
  条件 2   : 値 2;
  ⋮
  条件 N-1 : 値 N-1;
  1       : 値 N;
esac;
```

意味を下記する。

①条件 1 が成立するならば変数の次の値は値 1 である。

②条件 2 が成立するならば変数の次の値は値 2 である。

⋮

④その他の場合（条件 1～条件 N-1 が成立しない場合）は変数の次の値は値 N である。

条件判定の優先順位は上から下の方向（① → ② → ⋯ → ④）である（条件 1 と条件 2 が同時に成立する場合は変数の次の値は値 1 である）。その他の場合（1：値）は必ず記述する必要がある。なぜなら、いかなる条件も成立しない場合には変数の次の値が決まらないからである。

「1」は「True」の意味であり必ず成立する。その他の場合に指定する値がなければ当該変数名を記述して前回値保持とすればよい。

case 文の使用例を以下に示す。

```
例) next(cpu) := case
    Pow = 1 : busy;           . . . ①
    Pow = 2 : {busy, ready}; . . . ②
    1       : cpu;           . . . ③
    esac;
```

- ①Pow = 1 が成立するならば変数 cpu の次の値は busy である。
- ②Pow = 2 が成立するならば変数 cpu の次の値は busy あるいは ready である。
- ③その他の場合 (Pow = 1 も Pow = 2 も成立しない場合) は前回値保持とする。
(条件 1 と条件 2 が同時に成立する場合は変数 cpu の次の値は busy である)

2.3 FAIRNESS 条件記述部

NuSMV で CTL 式を検査する際に記述する FAIRNESS 条件と同等の条件を記載できる。

FAIRNESS 条件記述部では条件式毎に「FAIRNESS」と記述する。文末の「; (セミコロン)」は不要である。また「FAIRNESS 条件式」のように 1 行で書いても良い。

FAIRNESS とは、「公平性」という意味である。FAIRNESS 条件記述部では、SMV に対して公平性を保った条件の下で (公平性を保った状態遷移系で)、検査項目を検査することを指定する。

「FAIRNESS 条件式 1」とは、条件式 1 が無限回成立するパスのみを検査する、という意味である。

```
FAIRNESS
  条件式 1
FAIRNESS
  条件式 2
```

条件式は変数宣言部で宣言した変数と論理演算子を用いて作成する。

例 1) FAIRNESS fg = ON

fg = ON が無限回成立するパスのみを検査する。

例 2) FAIRNESS cpu = busy & Power = ON

cpu = busy かつ Power = ON が無限回成立するパスのみを検査する。

3. 検査項目記述部

検査項目は CTL 式を記述することができる。CTL 式は「SPEC」の後に記述する。文末の「; (セミコロン)」は不要である。「SPEC CTL 式」のように 1 行で書いても良い。

```

SPEC
  CTL 式 1
SPEC
  CTL 式 2
    
```

利用可能な CTL 式と、その意味を表 3-1 に示す。

表 3-1 利用可能な CTL 式と意味

記号	基本パターン	意味
AG	AG(P)	全てのパスにおいて、全ての状態で P が成立する
AX	AX(P)	全てのパスにおいて、次の状態で P が成立する
AF	AF(P)	全てのパスにおいて、将来のある状態で P が成立する
AU	A[P U Q]	全てのパスにおいて、 Q が成立する状態より前の状態まで P が成立する状態が続く
EG	EG(P)	あるパスにおいて、全ての状態で P が成立する
EX	EX(P)	あるパスにおいて、次の状態で P が成立する
EF	EF(P)	あるパスにおいて、将来のある状態で P が成立する
EU	E[P U Q]	あるパスにおいて、 Q が成立する状態より前の状態まで P が成立する状態が続く

Until は「[] (大括弧)」で囲まなければならない。

4. 使用可能な演算子

64bit 版モデル検査器で使用可能な演算子を表 4-1 に示す。数値演算子は整数の四則演算、比較演算を行う。論理演算子は真 (1) と偽 (0) の値を持つ変数 (boolean 変数) あるいは式の計算を行う。

表 4-1 SMV 専用言語で使用可能な演算子

分類	記号	使用例	意味
数値演算子	+	$A + B$	整数 A と整数 B との和
	-	$A - B$	整数 A と整数 B との差
	*	$A * B$	整数 A と整数 B の積
	/	A / B	整数 A を整数 B で割った商 (結果が小数値とならないように注意する)
	mod	$A \text{ mod } B$	整数 A を整数 B で割った剰余
	=	$A = B$	整数 A と整数 B とは等値
	!=	$A != B$	整数 A と整数 B とは等値ではない
	<	$A < B$	整数 A は整数 B より小さい
	>	$A > B$	整数 A は整数 B より大きい
	<=	$A <= B$	整数 A は整数 B 以下
>=	$A >= B$	整数 A は整数 B 以上	
論理演算子	&	$P \& Q$	論理式 P と論理式 Q の論理積 (and)
		$P Q$	論理式 P と論理式 Q の論理和 (or)
	!	$!P$	論理式 P の論理否定
	->	$P \rightarrow Q$	論理式 P ならば論理式 Q ($!P Q$ と同じ)
その他の演算子	union	$S \text{ union } T$	集合 S と集合 T の和集合
	in	$P \text{ in } R$	集合 R に集合 P が含まれれば真 集合 R に集合 P が含まれなければ偽

論理否定 (!) 以外の演算子は前後に必ず半角のスペースを挿入すること。

数値演算子と論理演算子の結合の優先順位を表 4-2 に示す。

表 4-2 演算子の結合の優先順位

優先順位	演算子
順位 1	!
順位 2	*, /, mod
順位 3	+, -
順位 4	union
順位 5	in
順位 6	=, !=, <, >, <=, >=
順位 7	AG, AX, AF, AU, EG, EX, EF, EU
順位 8	&
順位 9	
順位 10	->

優先順位が同一の演算子は、左から演算を行う。ただし、優先順位に頼らず、極力「()括弧」を用いて計算順序を明文化することを推奨する。