

形式手法概説

2008年3月

株式会社レンタコーチ

<http://homepage2.nifty.com/rent-a-coach/>

講座概要

◆受講対象者

- 形式手法を学ぼうとするソフトウェア技術者
- 形式手法の概要を知りたいマネージャ層

◆習得目標

- 形式手法の体系と主要な手法の特徴の理解
- 形式手法を深く学ぶための足がかりの習得

内容

- ◆形式手法に関する最近の動き
- ◆形式手法の体系
- ◆主要な形式手法の特徴
- ◆形式手法の活用
 - 産業界での適用に向くツール
 - オブジェクト指向との関係
 - 簡易形式手法
 - 国内での活用例
- ◆形式手法に関する情報源

形式手法とは何か？

- ◆ 論理学、集合論、代数学などの数学を基礎としたシステムの仕様記述手法、検証手法などの総称
- ◆ あいまいさを排除できことから、あるいは上流工程での検証を可能とすることから、品質向上のための手段として注目されている
- ◆ 1970年代に欧州の研究機関などで生まれたが、現在では産業界での適用に関心が向いている

形式手法: Formal Methodsの訳

第1章 形式手法に関する最近の動き

- ◆機能安全規格IEC61508
- ◆国内メディア
- ◆国内ツールベンダ

機能安全規格IEC61508(JIS C 0508)

- ◆ 1998年から2000年にかけて発行され、電気・電子・プログラマブル電子安全関連系の機能安全を取り扱う規格で、次の構成をとる：
 - 第1部：一般要求事項
 - 第2部：電気・電子・プログラマブル電子安全関連系に対する要求事項
 - 第3部：ソフトウェア要求事項
 - 附属書A 技術及び手法選択の指針
 - 第4部：用語の定義及び略語
 - 第5部：安全度水準決定方法の事例
 - 第6部：第2部及び第3部の適用指針
 - 第7部：技術及び手法の概観
- ◆ 第3部附属書Aで、形式手法(JISでは公式法と訳されている)がSIL4に対応する技法として推奨されている

IEC61508推奨技法と適用フェーズの対応表(一部)

フェーズ 推奨技法	安全 要求 仕様	ソフトウェア設計・開発				PE統 合	安全 妥当 性確 認	修正	適合 確認	機能 安全 評価
		アーキ テク チャ	ツ ー ル・言 語	詳細 設計	試験・ 統合					
コンピュータ支援仕様 ツール										
コンピュータ支援設計 ツール										
半形式手法	1	1		1						*
形式手法	1	1		1					*	
構造化手法		1		1						
障害検出・診断										
エラー検出・訂正符号										
異常表明プログラミング		2								
安全バッグ		2								
多種プログラミング		2								

*:決定表/真値表、形式証明、形式監査、記号実行 :シミュレーション/モデリングの要素技法のひとつ

国内メディアに現れた最近の動き

2004年9月	日経ITプロフェッショナル	「要件を厳密に定義する形式手法の実際」で形式手法の適用例を紹介
2005年2月	ニュースサイト	形式手法ツールSCADEの国内販売
2005年12月	日経エレクトロニクス	特集「ソフトウェアは硬い」でIEC61508を紹介し、形式手法を解説
2006年7月	日経コンピュータ	特集「バグゼロを目指し脚光を浴びる形式手法」で形式手法を紹介
2006年12月	日経BP社	「組込みソフトウェア2007」を出版し、形式手法を特集
2007年3月	ニュースサイト	形式手法の普及に向けてSCADE販売元CDAJとイーソルが提携
2007年6月	ニュースサイト	ドイツ製のモデル検査ツールが自動車業界のユーザ会で紹介された
2007年10月	ニュースサイト	キャッツ、スウェーデンの企業とモデル検査ツール販売で総代理店契約を締結

SCADE販売元CDAJとイーソルとの提携

モデル・ベースの組み込みソフトウェア開発ツール「SCADE」の国内販売を手掛けるシーディー・アダプコ・ジャパン(CDAJ)は、組み込みソフトウェアの受託開発を手掛けるイーソルと提携した。SCADEは[フランスEsterel Technologies, Inc.](#)が開発したツールで、Lustre言語やEsterel言語で記述したモデルの形式検証(モデル検査)が可能である。また、モデルからの自動コード生成機能が機能安全規格「[IEC 61508](#)」の認証を受けているという特徴がある。今回の提携を受けて、自動車や医療機器、半導体製造装置、ロボットなどのソフトウェア開発現場に向けて、イーソルのコンサルティングやSCADEの販売において両社で協業していく。

イーソルは、これまでモデル検査のコンサルティングにおいて、「LTSA」や「Uppaal」などの無償ツールを中心に利用してきた。例えば、「モバイルFeliCa」の開発を手掛けるフェリカネットワークスは、イーソルのコンサルティングを受けて、並行システムの検証にモデル検査ツールのLTSAを使っている。こうした無償ツールに加えて、SCADEを利用できるようになることで「コンサルテーションの領域が拡大できる」と(イーソルの藤倉氏)と期待しているという。

出典: Tech-ON!

ドイツ製のモデル検査ツールが自動車業界へ

[ドイツOSC Embedded Systems AG](#) (OSC ES社) は、自動車の制御系設計用ツールの最大手ドイツdSPACE GmbHの日本ユーザー会で講演し、OSC ES社の検証ツール「EmbeddedValidator」について説明した。EmbeddedValidatorは、dSPACE社の自動コード生成ツール「[TargetLink](#)」向けの制御系モデルに対し、形式検証(モデル検査: model checking)を行うツールである。ある状態への到達性(reachability)解析や各種プロパティの検証、変数のオーバーフロー・チェック、テスト・ケース生成などが可能である。

ドイツBMW社やDaimlerChrysler社、ドイツPorsche社が、既に自動車の電子制御ユニット(ECU)の開発に同社のツールを利用、日本国内では日産自動車最大のユーザーであるという。自動車業界以外では、Airbus社などの航空機メーカーや医療機器メーカーでも採用例があるという。

OSC ES社は当初、モデル検査ツールであるEmbeddedValidatorを中心に手掛けていたが、2007年4月にはテスト・ケース生成ツール「EmbeddedTester」も投入した。また、日本の自動車メーカーへのサポートを強化するべく、日本法人を2007年に設立している。

出典: Tech-ON!

キャッツ、UPPAALの販売代理店に

キャッツ株式会社(社長・上島康男、横浜市港北区)は18日、モデル検査ツールのリーディングカンパニーであるUPPAAL International AB社(CEO・Dr. Wan Yi、本社スウェーデン)と同社のモデル検査ツール「UPPAAL(ウパール)」の販売を行うことに基本合意した。キャッツはUPPAALの販売に加え、モデル検査を行うための「モデリング導入支援サービス」を合わせて提供していく。

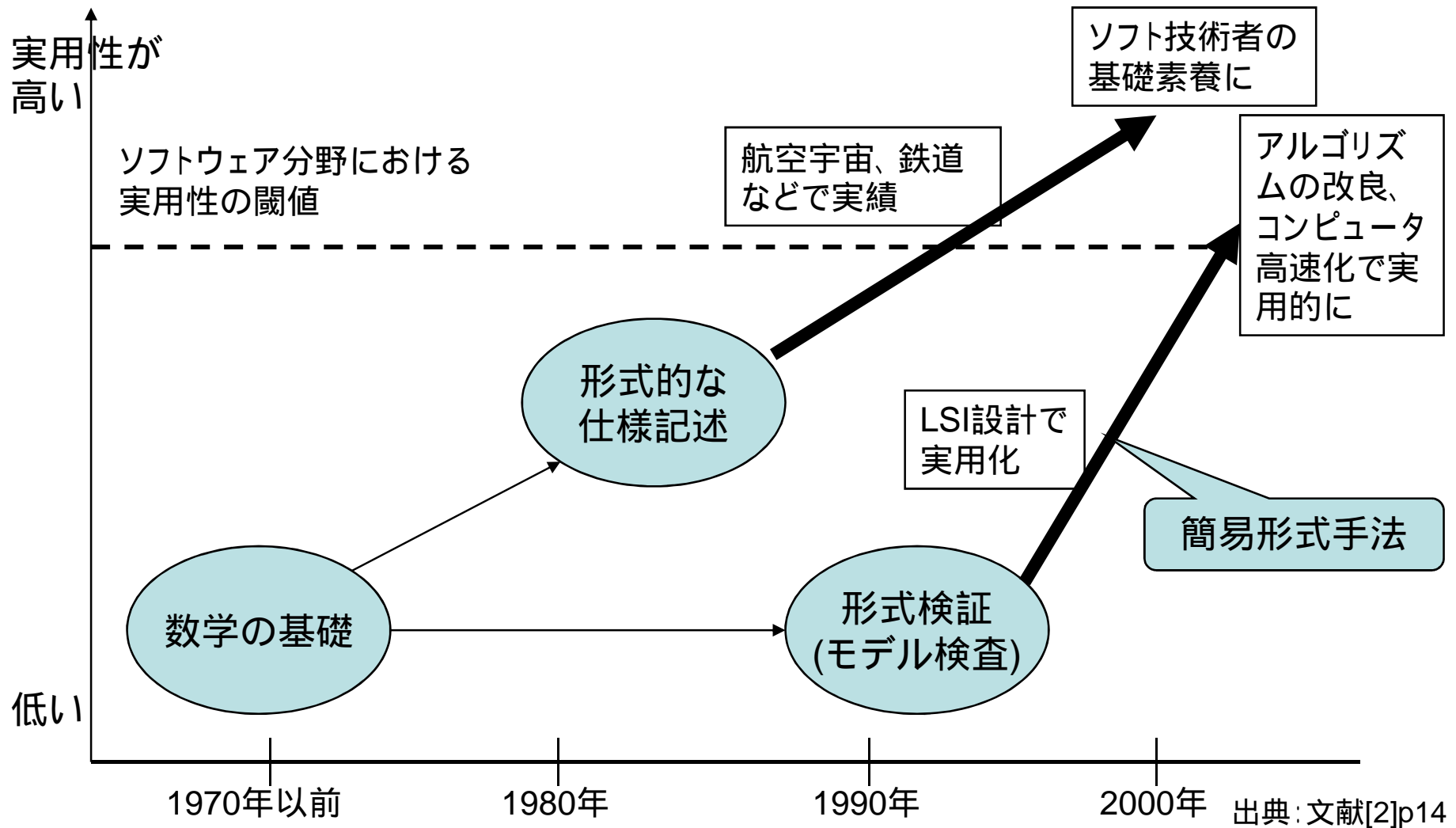
UPPAALはデンマークのオルボー(Aalborg)大学とスウェーデンのウプサラ(Uppsala)大学が開発した形式手法に基づいたモデリングと、モデル検査を行うツール。グラフィカルなシミュレーションによってリアルタイムシステムの妥当性検査と検証を行なえるなど、時間記述付きの状態遷移モデルを作成することによって、リアルタイム制御における不具合がないことを自動検査する機能を備えている。UPPAAL International AB社は、モデル検査技術を専門に取扱う会社として開発の中心メンバーだったDr. Wan Yiが2007年に設立した。

出典：<http://it.nikkei.co.jp/business/news/release.aspx?i=172894>

第2章 形式手法の体系

- ◆ 生い立ちと発展
- ◆ 分類
- ◆ 一覧

形式手法の生い立ちと発展



形式手法の分類

分類	解説	代表例
仕様記述	仕様を数学的に記述し、検証する手法。	
モデル規範型	仕様記述に論理学、集合理論などを用いる。	B、VDM、Z
性質規範型 (代数仕様)	仕様記述に代数学を用いる。	OBJ
モデル検査	システムの振る舞いモデルを作成し、特定の性質を検査する手法。	SMV SPIN
その他		
定理証明	定理証明を目的として、検証を完全に行うことを特徴とする手法。	HOL
プロセス代数	並行プロセスの振る舞いを代数的に記述する手法。	CSP、CCS
同期型言語	クロックに同期する記述言語。	Esterel Lustre
時相論理	時間を取り扱う論理の記法。	CTL、LTL
簡易形式手法	バグ出しを目的とする手法。	ESC

仕様記述手法一覧

名称	種別		
	手法	記法	ツール
B-Method			
B-Toolkit			
LOTOS			
LSC			
OBJ			
OCL			
Overture			
PerfectDeveloper			
RAISE			
RAISE tools			
RODIN platform			
SPARK			
VDM			
VDM Tools			
VDM++			
VDM-SL			
Z notation			
Z/EVES			
ZETA			

IEC61508
推奨手法

モデル検査手法一覧

名称	種別		
	手法	記法	ツール
Cadence SMV			
FDR2			
Garakabu			
JPF			
Murphi			
NuSMV			
SCADE			
SDV (開発コードはSLAM)			
SPIN			

その他の手法一覧

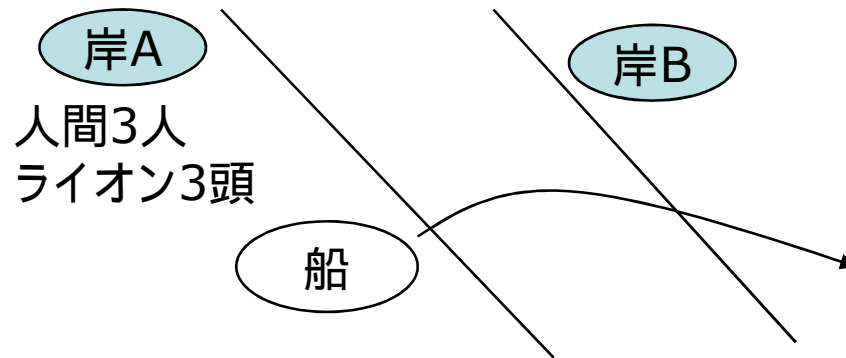
名称	種別		
	手法	記法	ツール
CCS			
CPN Tools			
CSP			
Esterel			
HOL			
JML			
Lustre			
Petri Nets			
ProofPower			
PVS			
SDL			
Sequence diagrams			
Signal/Polychrony			
Temporal Logic(これは総称)			

半形式手法一覧

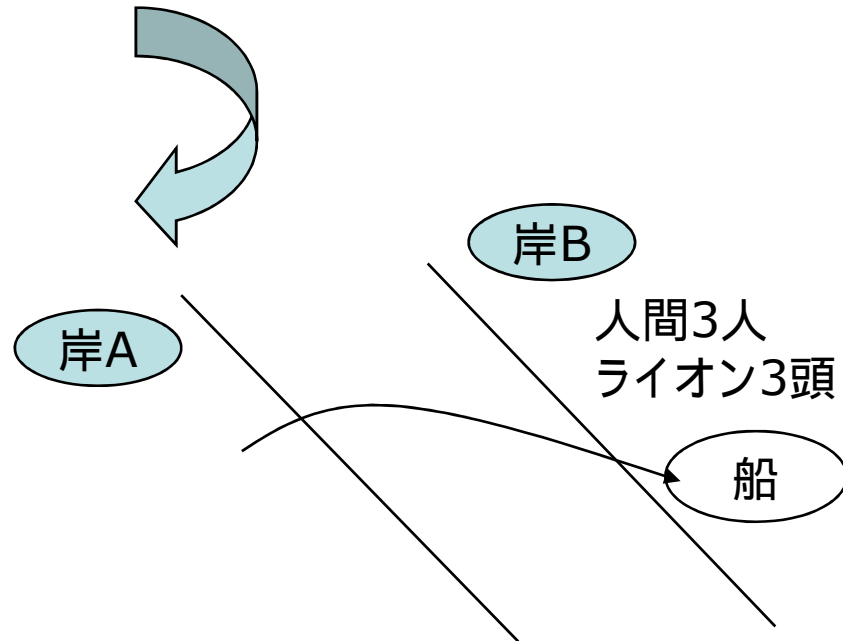
名称	種別		
	手法	記法	ツール
Data flow diagrams			
Decision tables			
Finite state machines/ state transition diagrams			
Logic/Function block diagrams			
Sequential function chart			
Time Petri nets			
Truth tables			

備考:IEC61508において半形式手法として紹介されている手法。

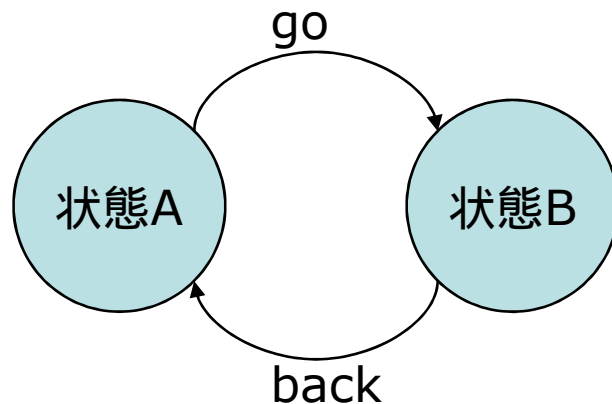
例題：パズルを解く



岸Aから岸Bへ船で全員を移動させる。
人間よりライオンが多くなると危険。
人間もライオンも船を操縦できる。
船の定員は2名。



モデリングと検証条件



状態A: 船が岸Aにある状態。

状態B: 船が岸Bにある状態。

注: 船の中は安全。

変数として:

Ma: 岸Aでの人間の数

La: 岸Aでのライオンの数

Mb, Lb: 岸Bでの人間とライオンの数

Side: AかB

操作として:

go: 岸AからBへの移動

back: 岸BからAへの移動

検証条件は $\text{not}(P \cup Q)$ として、その反例を探せばよい:

P: $(Ma \geq La \text{ or } Ma = 0) \text{ and } (Mb \geq Lb \text{ or } Mb = 0)$

Q: $Mb = 3 \text{ and } Lb = 3$

Promelaによる記述 (一部のみ)

```
inline go() /*船に乗る乗り方は5通り*/
{ atomic{
  if      /*条件が成立するものの中から非決定的に選ばれる*/
  ::(Ma>=2) -> Ma=Ma-2; Mb=Mb+2
  ::(Ma>=1 and La>=1) -> Ma=Ma-1; Mb=Mb+1; La=La-1; Lb=Lb+1
  ::(Ma>=1 and La>=0) -> Ma=Ma-1; Mb=Mb+1
  ::(La>=2) -> La=La-2; Lb=Lb+2
  ::(La>=1 and Ma>=0) -> La=La-1; Lb=Lb+1
fi
Side=B
}
}
```

```
active proctype example()
{ Ma=3;Mb=0;La=3;Lb=0;Side=A;
do
  ::(Side==A) -> go()
  ::(Side==B) -> back()
od
}
```

第3章 主要な形式手法の特徴

◆仕様記述手法

- VDM
- B-method
- Z notation
- OBJ, CafeOBJ
- LOTOS

◆モデル検査

- SMV
- SPIN

◆その他

- HOL
- CCS
- CSP
- Temporal Logic

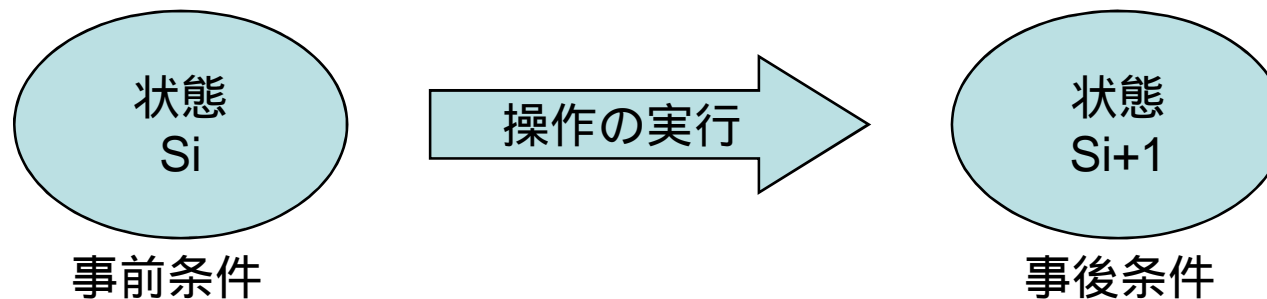
選択基準： IEC61508推奨技法プラスアルファ

VDM (Vienna Development Method)

特徴	<ul style="list-style-type: none"> ●プログラム仕様を形式的に記述し、開発する手法。 ●抽象的なデータ定義と操作仕様から始めて、順次、データ構造を具体化し、アルゴリズムに展開することによってプログラムを作成する。 ●この記述言語であるVDM-SLは、1996年にISO/IEC13817-1として国際標準になっている。 ●オブジェクト指向版の記述言語としてVDM++があり、これはESPRIT計画のAFRODITEプロジェクトで開発された。 ●モデルベースのシステムの記述には適するが、リアルタイムシステムなどのタイムベースのシステムの記述には向いていない。 ●コンパイラをはじめとして、多くのアプリケーションの記述に適用されている。
開発元	IBMのウィーン研究所で1970年代中頃に開発された。当初、PL/Iコンパイラの形式仕様記述と検証が目的で、その研究の過程で誕生した。
資料入手先	<ul style="list-style-type: none"> ●http://www.vdmportal.org/twiki/bin/view ●http://en.wikipedia.org/wiki/Vienna_Development_Method ●http://www.vdmtools.jp/
ツール入手先	国内のCSKシステムズが、VDM-SL及びVDM++に対応するVDM Toolsという商品を販売している。

事前条件と事後条件による仕様記述

- ◆ある操作の機能振舞いを実行前の状態から実行後の状態への遷移ととらえ、それぞれの状態を事前条件、事後条件として記述する。
- ◆Design by Contractという考え方に基づく。



陰関数表現と陽関数表現

	陰関数表現	陽関数表現
使い方	機能仕様を宣言的に記述	実行可能なプロトタイピング
データ	論理式で自由に規定	事前定義の型のみ
式	論理式	実行可能な式
ツール機能	構文チェック、型チェック	
	検証条件の生成 (対話的な証明)	インタープリタ

出典:文献[2]p127

VDM-SLの記述例(データ型定義)

types

IntSet = set of int;	集合型
String = seq of char;	列型
CharCode = map char to nat;	写像型
pos = real * real;	直積型
posR :: x:int y:int	複合型

写像型の演算子の例

定義域 dom m

値域 rng m

和写像 m1 munion m2

VDM-SLの記述例(関数記述)

割算 (陰関数表現として)

functions

Quotient(x,y:nat) z:nat*nat

pre y>0

post let mk_(q,r)=z in

x=q*y+r and

0<=r and r<y

階乗 (陽関数表現として)

functions

fact : nat -> nat

fact(n)==

case n :

0 -> 1,

others -> n*fact(n-1)

end

出典:文献[7]p66

VDM-SLの記述例(状態記述)

住所録に対する状態定義

```
state AddressBook of
```

```
  book:map Name to Address
```

```
  init s == s = mk_AddressBook({|->})
```

```
end
```

住所を登録する操作

```
operations
```

```
  AddAddress(name:Name, address:Address)
```

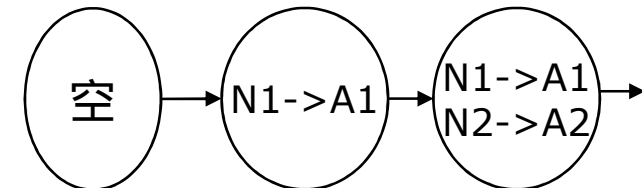
```
  ext wr book:map Name to Address
```

```
  pre name not in set dom book
```

```
  post book = book~ munion {name |-> address}
```

住所録

Name	Address
中村	所沢市
山田	国立市



book~は操作前の状態

出典:文献[7]p114

VDMの適用事例(その1)

現在はCSKシステムの一部門となっている日本フィッツが、証券会社のバックオフィス用のパッケージソフトウェアの開発にVDM手法を適用した。
開発規模はJava/C++で約8万行で、その仕様記述には、VDM++で約4万3千行。最初なので、多少、仕様記述の量は多く、効率が悪い。しかし、出荷後の品質と開発生産性は次のように向上した：

	1000行あたりの欠陥数
サブシステム1	0
サブシステム2	0.05
VDM++非適用部分	0.67

		見積	実績	実績/見積
サブシステム1	工数	38.5人月	14人月	36%
	期間	9ヶ月	3.5ヶ月	39%
サブシステム2	工数	147.2人月	60.1人月	41%
	期間	14.3ヶ月	7ヶ月	49%

出典：文献[3]

VDMの適用事例(その2)

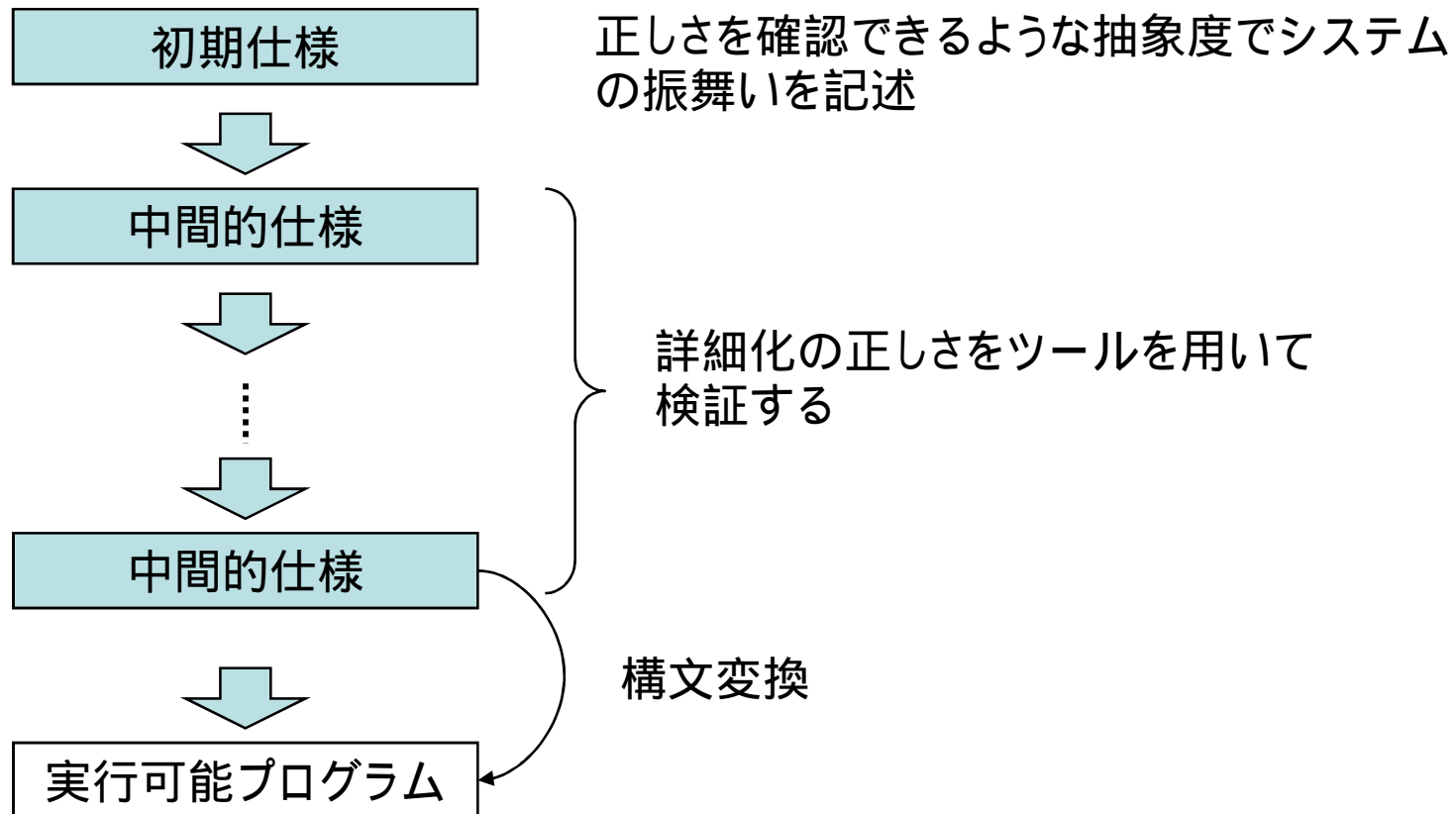
開発対象	モバイルFeliCa ICチップのファームウェア
開発要員	フェリカネットワークス。メンバー数は50から60人、平均年齢約30歳
記述対象	仕様のあいまいさをなくするために、外部仕様の記述に形式手法VDMを適用。
プロジェクトの流れ	2004年1月、プロジェクト発足。 2004年5月、VDM Toolsの講習受講。 2004年7月、仕様策定フレームワークの検討開始。 2004年10月、仕様開発開始。 この間、8ヶ月 2005年5月、外部仕様書初版完成。 2006年6月、コードリリース。 2006年8月、量産開始。
記述量	約10万行。作成したプログラムは、約11万行(C/C++)。
開発効率	8ヶ月間の仕様記述期間、一人平均して、約1900行/1ヶ月
効果	開発工程で検出した不具合原因において、 仕様不明瞭は1.8% 仕様の記述誤りは、0% 仕様策定段階で形式仕様の評価が可能となり、この段階で検出した不具合が6件。

出典:文献[5]

B-method

特徴	<ul style="list-style-type: none">●形式的仕様記述からプログラムコードを生成する手法。●段階的詳細化を全面的に採用。●検証条件をツールが自動検証できる比率が比較的高い。●産業界での適用例として、パリの地下鉄、空港内のシャトル電車の自動運航制御が公表されている。
開発元	Jean-Raymond AbrialがOxford大学でZ-notationに引き続いて開発。現在は、ETH Zurichの教授。詳細は： http://en.wikipedia.org/wiki/Jean-Raymond_Abrial
資料入手先	<ul style="list-style-type: none">●http://vl.fmnet.info/b/●http://en.wikipedia.org/wiki/B-Method●http://www.b-core.com/index.html
ツール入手先	B-Core(UK)がB-Tool及びB-Toolkitを販売している。詳細は： http://www.b-core.com/index.html 研究用としてはB4Freeが次のサイトから入手できる： http://www.b4free.com/

段階的詳細化



出典: 文献[2]P126

B-Methodの適用事例

項目 \ 適用対象	パリの地下鉄	空港のシャトル
Adaの行数	8万6000行	15万8000行
証明の数	2万7800個	4万3610個
自動検証できなかった比率	8.1%	3.3%
工数	7.1人月	4.6人月

出典:文献[2]P128

Z notation

特徴	<ul style="list-style-type: none"> ●プログラム仕様を形式的に記述するための言語。 ●ZF集合論から名前を取って命名された。 ●公理的集合論、ラムダ計算、一階述語論理で使われる数学的記法に基づいて、プログラム仕様と意図する振る舞いをシステムの状態と操作として記述する。 ●IBM社のOLTPモニタであるCICSのAPI仕様の記述に適用された。 ●2002年にはISO/IEC13568として国際標準となり、OSIネットワーク管理モデルなどの仕様記述に用いられている。 ●オブジェクト指向の拡張版としてZ++、Object-Z、TCOZがある。
開発元	Jean-Raymond Abrial等が1980年頃にOxford大学で開発した。
資料入手先	<ul style="list-style-type: none"> ●http://vl.zuser.org/ ●http://en.wikipedia.org/wiki/Z_notation ●http://www.zhmicro.com/ ●http://www.usingz.com/ ●http://www.itee.uq.edu.au/~smith/objectz.html ●http://www.comp.nus.edu.sg/~dongjs/tcoz.html
ツール入手先	Community Z Toolsがオープンソースとしてツールを提供している： http://czt.sourceforge.net/

Zの記述例 (VDMで用いた住所録)

```

__ AddressBook _____
| addressBook: Name + Address
| _____

```

住所録の定義

初期状態定義は省略。

```

__ AddAddress _____
| ΔAddressBook
| name?: Name
| address?: Address
| _____
| name? not dom(addressBook)
| addressBook' = addressBook {name? |-> address?}
| _____

```

住所登録操作

変数名'は操作後の状態。

出典:文献[7]P169

OBJ

特徴	<ul style="list-style-type: none">●代数仕様に基づく形式仕様記述言語。●同系列としてOBJ2、OBJ3、CafeOBJ、BOBJが総称される。●OBJのモジュールシステムはAdaやLOTOSに影響を与えた。●定理証明などの研究用、教育用に主に用いられている。
開発元	OBJは総称であり、最初のものは、1970年代中頃にCalifornia大学のJoseph Goguenが開発した。 http://www.cs.ucsd.edu/users/goguen/
資料入手先	<ul style="list-style-type: none">●http://vl.fmnet.info/obj/●http://www.cs.ucsd.edu/users/goguen/sys/obj.html
ツール入手先	OBJ3に関するものが入手可能で、次のサイトにその情報が掲載されている： http://www.cs.ucsd.edu/users/goguen/sys/obj.html

CafeOBJ

特徴	<ul style="list-style-type: none">●代数仕様による形式仕様記述言語。●国内で研究開発された形式手法。●OBJの後継として、書き換え論理 (rewriting logic)、隠蔽代数 (hidden algebra) という新しいパラダイムを追加。
開発元	北陸先端科学技術大学院大学の二木厚吉教授らのプロジェクトが開発。二木教授は、2007年10月にZIPCを販売しているキャッツの研究所所長を兼務。
資料入手先	● http://www.ldr.jaist.ac.jp/cafeobj/
ツール入手先	次のサイトから入手できる： http://www.ldr.jaist.ac.jp/cafeobj/system.html

CafeOBJの記述例

```
-- flag object
mod* FLAG { *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag -- methods
  bop up?_ : Flag -> Bool -- attribute
  var F : Flag
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
}
-- proof of (rev rev f) =b= f
open FLAG
var B : Bool
eq not not B = B .
op f : -> Flag .
red (rev rev f) =*= f .
close
```

出典: <http://www.ldl.jaist.ac.jp/cafeobj/lib/flag.mod>

LOTOS (Language of Temporal Ordering Specifications)

特徴	<ul style="list-style-type: none"> ●Temporal orderingに基づく仕様記述言語。 ●OSIのプロトコル記述に活用され、1990年にISO/IEC8807として国際標準となっている。 ●プロセス代数CCS、CSP、それに抽象データ型言語ACT ONEに基づいている。 ●事象が発生する順序を記述でき、OSIだけでなく、逐次、平行又は分散システムに適用されている。 ●研究活動は、今なお、E-LOTOSとして引き継がれている。
開発元	ISOにおける活動の一環として1980年代後半に開発された。
資料入手先	<ul style="list-style-type: none"> ●http://www.cs.stir.ac.uk/~kjt/research/well/ ●http://en.wikipedia.org/wiki/Language_Of_Temporal_Ordering_Specification
ツール入手先	ツールの入手先は次のサイトに記載されている： http://www.cs.stir.ac.uk/~kjt/research/well/

SMV (Symbolic Model Verifier)

特徴	<ul style="list-style-type: none">●有限状態システムのモデル検査ツール。●検証条件の記述にCTL(Computational Tree Logic)を用いる。●二分決定木(BDD: Binary Decision Diagram)というデータ構造を利用して、効率的な探索を図る。●Symbolic model checkingという探索アルゴリズムを使用する。●LSI回路設計の検証用として実用化された。●後継ツールとして、NuSMV、Cadence SMVがある。
開発元	Carnegie Mellon大学におけるModel checkingグループが開発。
資料入手先	<ul style="list-style-type: none">●http://www.cs.cmu.edu/~modelcheck/smv.html●http://www.cs.cmu.edu/~modelcheck/tour.htm
ツール入手先	次のサイトから入手できる： http://www.cs.cmu.edu/~modelcheck/smv.html

モデル検査の概要

- ◆ システムの振舞い仕様を状態遷移モデルとして記述する。
- ◆ システムが満たすべき仕様を検証条件(プロパティ)として記述する。
- ◆ 状態遷移させて、検証条件に違反しないことを網羅的に調べる。逆に、違反するケースを探し出す。
- ◆ 高速化が課題：
 - 探索アルゴリズムやデータ構造を工夫して、「効率化」を図る。
 - 探索空間を狭める「抽象化」を施す。

検査できる性質

◆安全性(safety)

- 状態遷移系の危険状態を検証条件として記述し、その状態に到達するか否かを調べる。

◆活性 (liveness)

- 状態遷移系に要求される特定の振る舞いを検証条件として記述し、それが成立するか否かを調べる。

◆検証条件は、時間を取り扱える時相論理を用いて記述される。

- たとえば、危険な状態には決して到達しない。
- 要求されている状態にはいつか到達する。

Symbolic model checking

- ◆状態の記述に記号論理学を使い、論理式をBDDに変換する。
- ◆初期状態から遷移したすべての状態の集合を論理式で表し、検証条件に対応する論理式との交わりを計算する。
- ◆状態数10の120乗まで検査可能。

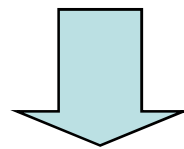
Binary Decision Diagram

3状態の集合:

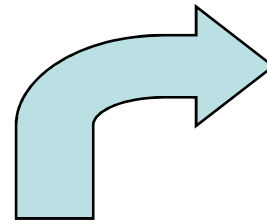
(notR1 notL1 notR2 notL2)

(R1 notL1 notR2 notL2)

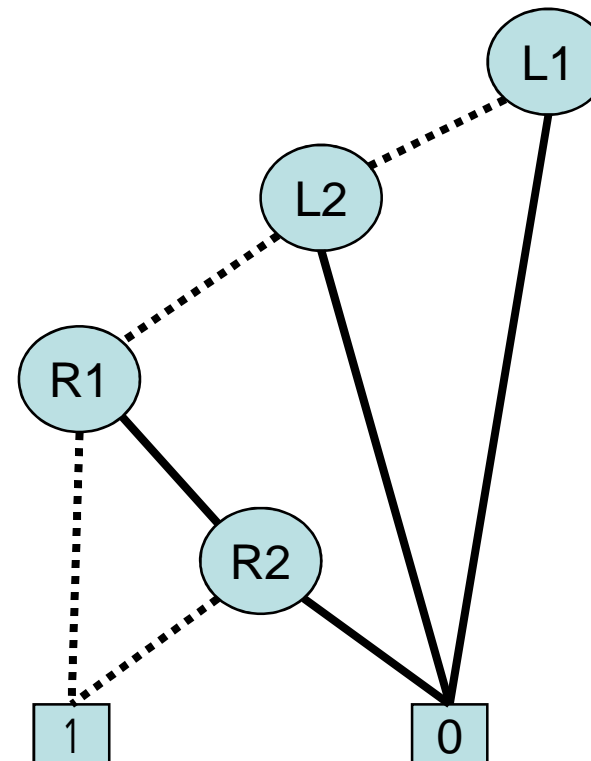
(notR1 notL1 R2 notL2)



notL1 notL2 (notR1 notR2)



BDDによる表現



実線は真、点線は偽を表す。

出典: 文献[2]P143,144

SMVの記述例

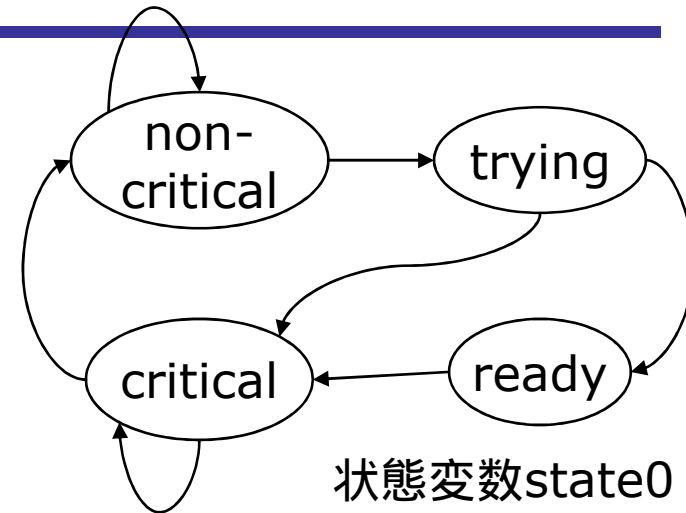
```

MODULE prc(state0, state1, turn, turn0)
ASSIGN
init(state0) := noncritical;
next(state0) :=
case
  (state0 = noncritical) : {trying,noncritical};
  (state0 = trying) & ((state1 = noncritical) | (state1 = trying) |
    (state1 = ready)): ready;
  (state0 = ready): critical;
  (state0 = trying) & (state1 = trying) & (turn = turn0): critical;
  (state0 = critical) : {critical,noncritical};
1: state0;
esac;

```

以下、略

出典 : <http://www.cs.cmu.edu/~modelcheck/tour.htm>



状態変数state0
の遷移図イメージ

SPIN

特徴	<ul style="list-style-type: none">●分散ソフトウェアシステムのモデル検査ツール。●非同期分散アルゴリズムに向くPromela (Process Meta Language) で対象は記述される。●検証条件はLTL (Linear Temporal Logic) で記述される。●モデル検査だけでなくシミュレータの機能も持ち、実行履歴を取ることができる。●Partial order reduction等の方法を用いて、メモリ節約と高速化を実現している。●通信プロトコルなどの通信分野で実用化された。●ACM賞を2001年に受賞した。●文献や教科書が多く、代表的なモデル検査ツールである。
開発元	Gerard J. Holzmann等が、1980年にベル研究所で開発。
資料入手先	<ul style="list-style-type: none">●http://spinroot.com/spin/whatispin.html●http://en.wikipedia.org/wiki/SPIN_model_checker
ツール入手先	オープンソースとして次のサイトから入手できる： http://spinroot.com/spin/whatispin.html

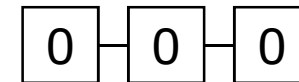
Partial order reduction

- ◆モデル検査における探索アルゴリズムのひとつで、抽象化と呼ぶ方法に分類されている。
- ◆並行動作する非同期プロセスが複数ある場合、プロセス間に通信がなければ状態遷移は独立していると考えて、それぞれの中の状態遷移だけに対象を絞る。
- ◆主に通信プロトコルの検証に適している。

SPINの記述例

```
int digit0=0,digit1=0,digit2=0;
active proctype counter()
{ do
    ::(digit0==2)-> atomic{
        digit0=0;
        if
        ::(digit1==2) ->
            digit1=0;
            if
            ::(digit2==2) ->
                digit2=0;
            ::else ->
                digit2++
            fi
        ::else ->
            digit1++
        fi }
    ::else ->
        digit0++
od}
```

3桁3進カウンタ



atomic{}は連続実行区間を示す。最小桁が2の場合の処理をここで記述。

最小桁が2以外では、最小桁をインクリメント。

出典:文献[8]p117

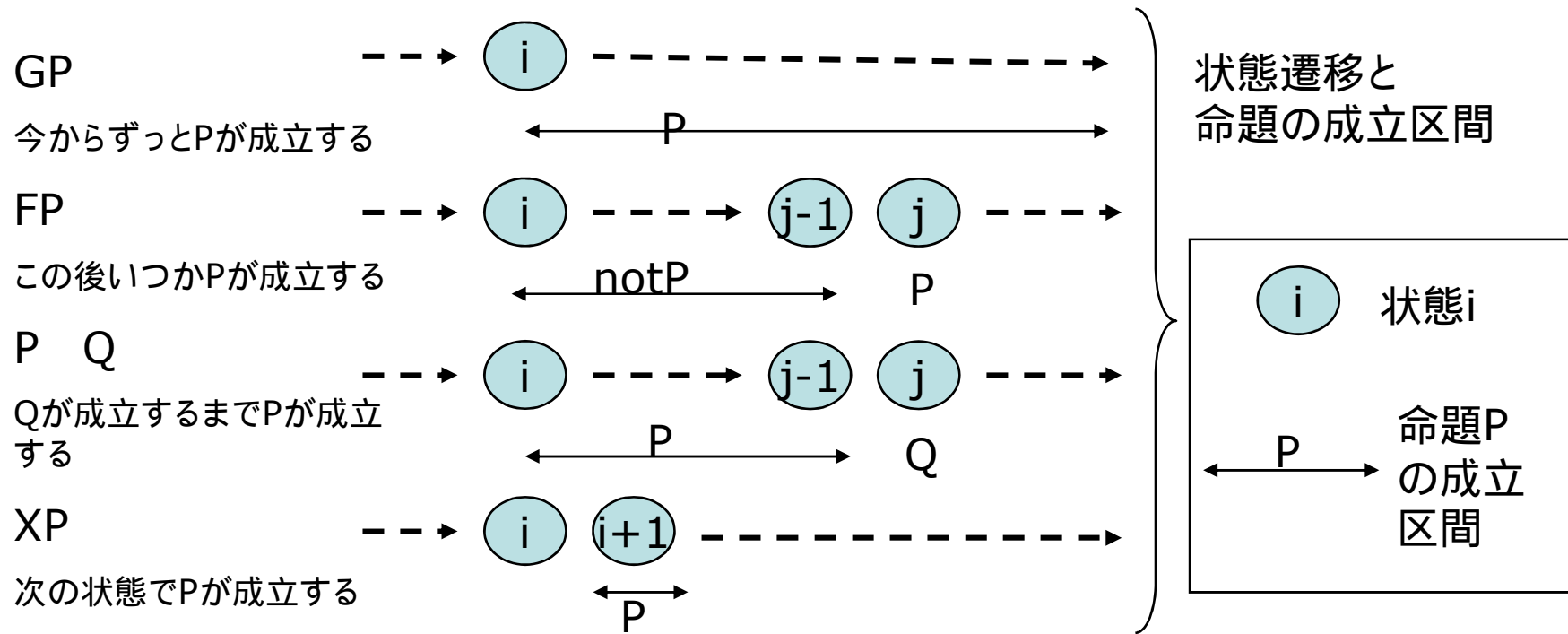
Temporal Logic

特徴	<ul style="list-style-type: none">●モデル検査において検証条件を記述するための記法。●時間軸を含む論理関係を表現できる。●これは総称であり、CTL、LTL等の記法が提案されてきている。
開発元	アイデアは1960年代に遡り、その後、多くのコンピュータ科学者が研究を続け、発展してきた。
資料入手先	http://en.wikipedia.org/wiki/Temporal_logic
ツール入手先	モデル検査ツールに内蔵されている。

Temporal logicを用いると、次のような表現ができる：

- Until: Qが成立するときまで、Pが成立する。
- Next: 次の状態ではPが成立する。
- Finally: いつかPが成立する。
- Globally: いつでもPが成立する。

LTLの記述例



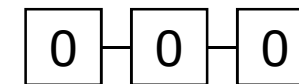
押しボタンが点灯したら、いずれ必ず歩行者信号機は青になる：
 $G((\text{押しボタン} = \text{点灯}) \rightarrow F(\text{歩行者用信号} = \text{青}))$

出典:文献[8]p10

検証条件の記述例

```
int digit0=0,digit1=0,digit2=0;
active proctype counter()
{ do
    ::(digit0==2)-> atomic{
        digit0=0;
        if
        ::(digit1==2) ->
            digit1=0;
            if
            ::(digit2==2) ->
                digit2=0;
            ::else ->
                digit2++
            fi
        ::else ->
            digit1++
        fi }
    ::else ->
        digit0++
    od}
```

3桁3進カウンタ



検証目的: 各桁がすべて1になることがあることを調べたい。



検証条件: その逆を取って、各桁がすべて1になることはないとする。



$\text{notF}(\text{digit0}=1 \ \text{digit1}=1 \ \text{digit2}=1)$

出典: 文献[8]p118

HOL

特徴	<ul style="list-style-type: none">●自動的に定理を証明するツール。●定理の記述に高階論理 (higher order logic) を用いる。●定理証明やLSI記述及び検証に適用されている。
開発元	1988年にHOL88という名前でCambridge大学で開発された。詳細は次のページを参照： http://www.cl.cam.ac.uk/research/hvg/HOL/history.html
資料入手先	<ul style="list-style-type: none">●http://vl.fmnet.info/hol/●http://www.cl.cam.ac.uk/research/hvg/HOL/●http://en.wikipedia.org/wiki/Higher-order_logic
ツール入手先	最新版はHOL4と呼ばれ、オープンソースとして次のサイトから提供されている： http://hol.sourceforge.net/

CCS (Calculus of Communicating Systems)

特徴	<ul style="list-style-type: none">● 平行動作する通信プロセスの振る舞いを記述する手法。● プロセス通信のメカニズムはCSPと同様。● Pi-calculusに引き継がれている。
開発元	Robin MilnerがEdinburgh大学で1980年代に開発。
資料入手先	<ul style="list-style-type: none">● http://en.wikipedia.org/wiki/Calculus_of_Communicating_Systems● http://www.lfcs.inf.ed.ac.uk/reports/86/ECS-LFCS-86-7/index.html● Communication and Concurrency, Robin Milner, Prentice-Hall, 1989
ツール入手先	不明。

CSP (Communicating Sequential Processes)

特徴	<ul style="list-style-type: none">● 平行動作するプロセス群の振る舞いを記述する手法。● 1980年代初期に開発され、INMOS社マイクロプロセッサTransputer、プログラミング言語occam等に活用された。
開発元	C. A. R. ("Tony") HoareがOxford大学で1978年に始めて開発した。
資料入手先	<ul style="list-style-type: none">● http://vl.fmnet.info/csp/● http://www.wotug.org/csp.shtml● http://en.wikipedia.org/wiki/Communicating_Sequential_Processes● "Communicating Sequential Processes", published by Prentice-Hall, 1985
ツール入手先	英国のFormal SystemsがFDR2というツールを提供している： http://www.fsel.com/software.html

第4章 形式手法の活用

◆ 産業界での適用に向くツール

- VDM Tools
- SCADE
- SDV
- LTSA
- Garakabu(開発中)
- UPPAAL

◆ オブジェクト指向との関係

◆ 簡易形式手法(LFM:Lightweight Formal Methods)

◆ 国内での活用例

VDM Tools

特徴	<ul style="list-style-type: none">●形式仕様記述言語VDM-SLとそのオブジェクト指向版VDM++をサポートするツール。●国内においても証券会社のバックオフィス用のパッケージソフトウェアなどの開発に適用されている。
開発元	デンマークのIFAD社が1980年代に開発し、現在は、国内のCSKシステムズが引き継いでいる。
資料入手先	<ul style="list-style-type: none">●http://www.vdmportal.org/twiki/bin/view●http://en.wikipedia.org/wiki/Vienna_Development_Method●http://www.vdmttools.jp/
ツール入手先	国内のCSKシステムズが販売している。関連情報の入手は次のサイトから： http://www.vdmttools.jp/

主な機能

VDM++、VDM-SLの統合開発環境であるVDMToolsは、形式技術と実用技術を巧みに統合し、現在は以下の機能を提供します。

- ・仕様の構文チェック
- ・仕様の型チェック
- ・証明課題生成機能
- ・実行可能仕様のインタプリタとデバッグ機能
- ・実行可能仕様のコードカバレッジ計測機能
- ・Rational Rose との連動機能
- ・VDM-SL、VDM++の清書ツール
- ・実行可能仕様からJavaやC++コードの生成機能(オプション)
- ・JavaからVDM++への変換(オプション)
- ・CORBA APIとの連動機能(オプション)

出典 : <http://www.vdmttools.jp/download/vdmttools>

SCADE

特徴	<ul style="list-style-type: none"> ● Safety Criticalなソフトウェアを開発するためのツール。 ● 二つの同期型言語Lustre、Esterelをサポートして、データフローと状態遷移の仕組みを統合している。 ● 安全規格DO-178B、IEC61508に準拠するコードを生成できる。 ● SCADE Suite、SCADE Driveという商品名で販売されていて、産業界での適用例が多い。
開発元	<p>1980年代後半にLustreベースのSCADEが開発され、2001年にEsterel TechnologiesがこれにEsterelサポートを統合して現在に至る。</p> <p>http://www.esterel-technologies.com/</p>
資料入手先	<ul style="list-style-type: none"> ● http://en.wikipedia.org/wiki/SCADE ● http://www.esterel-technologies.com/products/scade-suite/ ● http://www.esterel-technologies.com/products/scade-drive/ ● モデル検査を取り入れた欧州発のソフト開発環境「SCADE Drive」、小西晃輔著、組み込みソフトウェア2007 モデルに基づく開発方法論のすべて、日経BP社
ツール入手先	<p>国内ではシーディーアダプコジャパンが販売している。詳細は次のページから：</p> <p>http://www.cdaj.co.jp/product/070000scade/index.html</p>

産業界における適用例

- Primary Flight control system (FCS) of the Airbus A380
- Secondary FCS of the Airbus A340
- Automatic Pilots for Eurocopter: EC-135, EC-145, EC-155, EC-225.
- More than 10 nuclear power plants in Europe
- Eurostar ITCS (Interlocking and Train Control Systems)
- Falcon 7X Flight Control and Braking System
- Pratt & Whitney PW610, PW615F and PW535 engine's FADEC (resp. in the Eclipse, Cessna Citation Mustang and Citation Encore+)

出典 : <http://en.wikipedia.org/wiki/SCADE>

SCADE Drive™の紹介

SCADE Drive™は、形式手法をコア技術としたセーフティクリティカルな組込み制御ソフトウェア開発支援ツールです。グラフィカルな制御仕様の設計(モデルベース開発)、形式検証、シミュレーション、自動Cコード生成までの工程を1つのツール環境で実現します。自動Cコード生成は、世界的にも稀なIEC-61508 SIL3/SIL4認証を取得し、制御モデルと自動生成されるCコードとの一貫性が保証されています。これによって、ユニットテストに要する時間の大幅な削減ばかりでなく、より信頼性の高い組込み制御ソフトウェア開発を可能にします。

出典 : <http://www.cdaj.co.jp/product/070000scade/000119.html>

SDV (開発コードはSLAM)

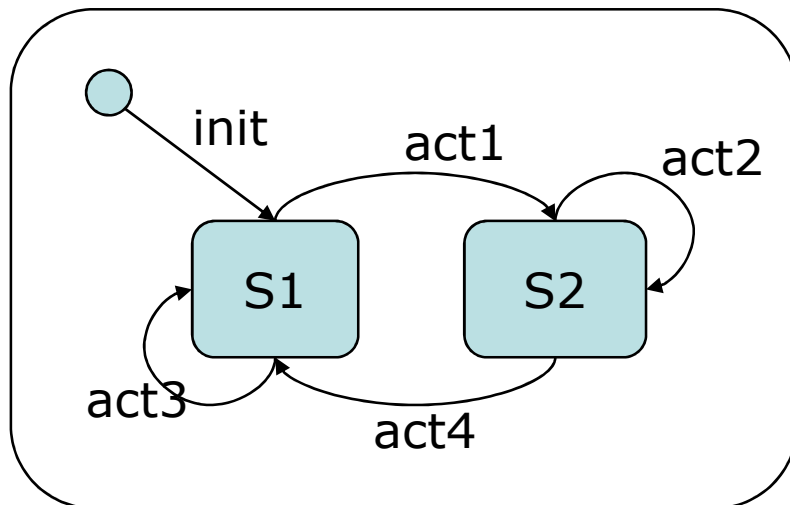
特徴	<ul style="list-style-type: none">●Windowsデバイスドライバのバグを検出するツール。●C言語のコードを対象に、API使用方法を自動検証する。●インタフェースの記述にSLIC(Specification Language for Interface Checking of C)という言語を使用している。
開発元	Microsoft ResearchにおけるSLAMプロジェクトが開発。
資料入手先	<ul style="list-style-type: none">●http://research.microsoft.com/slam/●http://en.wikipedia.org/wiki/SLAM_project
ツール入手先	Windowsデバイスドライバ開発キットに含まれる。

LTSA (Labelled Transition System Analyzer)

特徴	<ul style="list-style-type: none"> ●状態遷移系のモデル検査ツール。 ●時相論理を使わず、比較的簡単に使えることが特徴。 ●プロセス代数FSP(Finite State Processes)を基盤にしている。 ●Eclipse用のプラグインが用意されている。 ●国内ではイーソルがコンサルティングサービスを提供している。
開発元	英国Imperial Collegeが開発。
資料入手先	<ul style="list-style-type: none"> ●http://www.doc.ic.ac.uk/ltsa/ ●http://www.doc.ic.ac.uk/ltsa/eclipse/ ●http://edutool.com/ltsa/ ●組込みソフトウェアの設計 & 検証、藤倉俊幸著、CQ出版社
ツール入手先	<p>ツールそのものは無償で、次のサイトから入手可能:</p> <p>http://www.doc.ic.ac.uk/ltsa/ http://www.doc.ic.ac.uk/ltsa/eclipse/</p>

プロセス代数を使った記述例

UML状態チャート



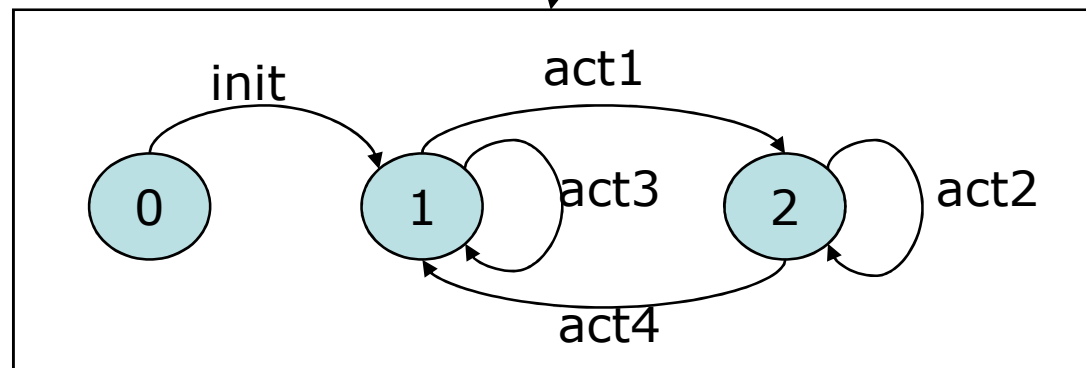
FSP形式

```

P1=(init->S1)
S1=(act1->S2 | act3->S1)
S2=(act4->S1 | act2->S2)
    
```

演算できる表記として:
 $S1 = act1 \cdot S2 + act3 \cdot S1$

LTS形式



出典:文献[8]P195

Garakabu

特徴	<ul style="list-style-type: none"> ●状態遷移表ベースのモデル検査ツール。 ●組込みソフトウェア開発で用いられている状態遷移表ツールZIPCとの連携が可能。 ●検証できる性質は、不可セルの検出、デッドロックの検査、制約条件の検査、及び必要条件の検査である。 ●国産ツールであり、現在も開発継続中。
開発元	<p>文科省の知的クラスター創生事業の一環として、ZIPCを開発販売しているキャッツ、福岡知的クラスター研究所、九州大学が共同開発した。2007年9月から第2期として、車載組込みソフト向けに開発を継続。</p> <p>http://www.zipc.com/index.html http://www.fleets.jp/index.html http://www.kyushu-u.ac.jp/top.php</p>
資料入手先	<ul style="list-style-type: none"> ●モデル検査ツールGarakabuを開発、日経エレクトロニクス2006年1月30日号、同2月13日号 ●ESEC2006及びET2006に出展
ツール入手先	未発売。キャッツが外販を検討中。

UPPAAL

特徴	<ul style="list-style-type: none">●リアルタイムシステムの妥当性確認と検証を行うツール。●モデルを入力するためのGUIとモデル検査機能を備えている。●検証条件の記述に時間オートマトンを用いていて、時間制約を検証できる。●国内でキャッツが販売提携を発表している。
開発元	デンマークのオルボー (Aalborg) 大学とスウェーデンのウプサラ (Uppsala) 大学が開発した。開発の中心メンバーだった Dr. Wan Yi が 2007 年に設立した UPPAAL International AB 社が販売している。
資料入手先	<ul style="list-style-type: none">●http://www.it.uu.se/research/docs●http://www.uppaal.com/●http://www.it.uu.se/research/group/darts/uppaal/tutorial-jp.pdf
ツール入手先	ツールそのものは無償で、次のサイトから入手可能： http://www.uppaal.com/

オブジェクト指向との関係

- ◆ UML記述のあいまいさを形式手法が補う。UML2.0に形式手法の成果が盛り込まれた。
 - OCL
 - シーケンス図
- ◆ オブジェクト指向言語が対象になる。
 - VDM++
 - JPF
 - JML、ESC/Java2
 - SPEC#
- ◆ UML記述から形式手法への変換ツールの登場。
- ◆ テストデータ、テストシナリオの自動生成によって、モデル駆動型開発、テストとの関連も強い。

OCL (Object Constraint Language)

特徴	<ul style="list-style-type: none">●モデル記述言語UMLにおけるモデルの制約を記述する言語。●表記内容のあいまいさを排除するために、UML2.0に取り入れられた。●OMGのモデル変換標準QVTが採用し、これに準拠する言語に基本部として組み込まれている。
開発元	IBM社が1990年代後半にUMLの形式仕様記述言語として開発。
資料入手先	<ul style="list-style-type: none">●http://www-306.ibm.com/software/awdtools/library/standards/index.html●http://en.wikipedia.org/wiki/Object_Constraint_Language●http://www.csci.csusb.edu/dick/samples/ocl.html●特集UMLエキスパートへの道 第4部正式リリースが迫るUML2.0の実像、日経ITプロフェッショナル2003年12月号
ツール入手先	次のサイトからパーサーを入手できる： <ul style="list-style-type: none">●http://www-306.ibm.com/software/awdtools/library/standards/ocl-download.html●http://dresden-ocl.sourceforge.net/

LSC (Live Sequence Charts)

特徴	<ul style="list-style-type: none">●シーケンス図を用いる形式検証を可能とする記法。●プロセス又はオブジェクト間のメッセージ通信を記述するMSC (Message Sequence Charts) を形式検証に使えるように拡張している。●UML2.0においてシーケンス図の強化に採用されている。
開発元	デンマークOldenburg大学のW.Damm等が2001年頃から提唱。
資料入手先	<ul style="list-style-type: none">●http://ses.informatik.uni-oldenburg.de/publications/spp-lncs-use-charts.pdf●http://www.info.fundp.ac.be/~ybo/docs/icse02/scesm-ybo.pdf●http://netail.net/?date=200602
ツール入手先	不明。

VDM++

特徴	●仕様記述言語VDM-SLのオブジェクト指向拡張版。
開発元	ESPRIT計画のAFRODITEプロジェクトで開発された。
資料入手先	● http://www.vdmportal.org/twiki/bin/view ● http://en.wikipedia.org/wiki/Vienna_Development_Method ● http://www.vdmttools.jp/
ツール入手先	国内のCSKシステムズが、VDM-SL及びVDM++に対応するVDM Toolsという商品を販売している。関連情報の入手は次のサイトから： http://www.vdmttools.jp/

JPF (Java Path Finder)

特徴	<ul style="list-style-type: none">●プログラムの検証ツール。●Javaのプログラムに注釈として検証条件を記述する。●Javaバイトコードが検証対象。
開発元	NASA Ames Research Centerで1999年以降に開発されてきている。
資料入手先	● http://javapathfinder.sourceforge.net/
ツール入手先	オープンソースとして次のサイトから入手できる： http://javapathfinder.sourceforge.net/

UMLから形式手法への変換

UML側の記法	形式手法側の記法	開発元
クラス図、状態図	Promela	富士ゼロックス。 2006年9月に学会発表済み。
		NEC、 北陸先端科学技術大学院大学。 2007年9月のSPLC国際学会で最新成果を発表(*1)。
	独自	日本IBM。 開発中。

* 1 : <http://techon.nikkeibp.co.jp/article/NEWS/20070927/139778/>

簡易形式手法

- ◆形式的に仕様を記述されたプログラムの証明ではなく、バグ出しを目的とする手法。
- ◆モデル検査の実績に刺激されて、1990年代半ばに誕生。
- ◆その一例がプログラムの静的解析ツール：
 - ESC/Java2
 - Spec#
 - VARVEL

LFM: Lightweight Formal Methods

プログラム検証対象の比較

対象	仕様記述 手法	モデル検査	静的解析
要求			—
設計			
ソースコード	—	(自動生成)	

:主対象。

JML (Java Modeling Language)

特徴	<ul style="list-style-type: none">●Javaで書かれたプログラムの仕様記述言語。●注釈として事前条件、事後条件、不変条件を記述する。●Eiffel言語で開発されたDesign by Contractという考えを採用し、Javaモジュールとその呼び出し側との約束事を記述している。
開発元	Iowa州立大学でGary T. Leavens等が開発。
資料入手先	<ul style="list-style-type: none">●http://www.cs.iastate.edu/~leavens/JML/●http://secure.ucd.ie/products/opensource/ESCJava2/docs.html●http://en.wikipedia.org/wiki/JML
ツール入手先	オープンソースとして共通のツールを次のサイトから入手できる： http://sourceforge.net/projects/jmlspecs/ 他に、ESC/Java2などのツールも提供されている。

JMLの記述例

```
public class BankingExample {
    public static final int MAX_BALANCE = 1000;
    private int balance;
    private boolean isLocked = false;
    //@ invariant balance >= 0 && balance <= MAX_BALANCE;
    //@ assignable balance;
    //@ ensures balance == 0;
    public BankingExample() { ... }
    //@ requires amount > 0;
    //@ ensures balance = ¥old(balance) + amount;
    //@ assignable balance;
    public void credit(int amount) { ... }
    //@ requires amount > 0;
    //@ ensures balance = ¥old(balance) - amount;
    //@ assignable balance
    public void debit(int amount) { ... }
    //@ ensures isLocked == true;
    public void lockAccount() { ... }
    //@ signals (BankingException e) isLocked;
    public /*@ pure @*/ int getBalance() throws BankingException { ... }
}
```

出典: <http://en.wikipedia.org/wiki/JML>

ESC/Java2 (Extended Static Checker for Java)

特徴	<ul style="list-style-type: none">●Javaで記述されたプログラムの静的解析ツール。●仕様記述言語JMLで書かれた注釈をもとにして、実行時エラーを検出。
開発元	Dublin大学のKindSoftwareというグループが開発。 http://secure.ucd.ie/
資料入手先	● http://secure.ucd.ie/products/opensource/ESCJava2/docs.html
ツール入手先	KindSoftwareがツールを提供している： http://secure.ucd.ie/products/opensource/ESCJava2/

Spec#

特徴	<ul style="list-style-type: none">●C#を拡張した仕様記述言語とその静的解析ツール。●JML同様にDesign by Contractの考えを踏襲して、事前条件、事後条件、不変条件の記述を可能としている。
開発元	Microsoft ResearchでMichael Barnett等の研究プロジェクトが開発。
資料入手先	<ul style="list-style-type: none">●http://research.microsoft.com/specsharp/●http://en.wikipedia.org/wiki/Spec_sharp
ツール入手先	次のサイトから入手できる： http://research.microsoft.com/specsharp/

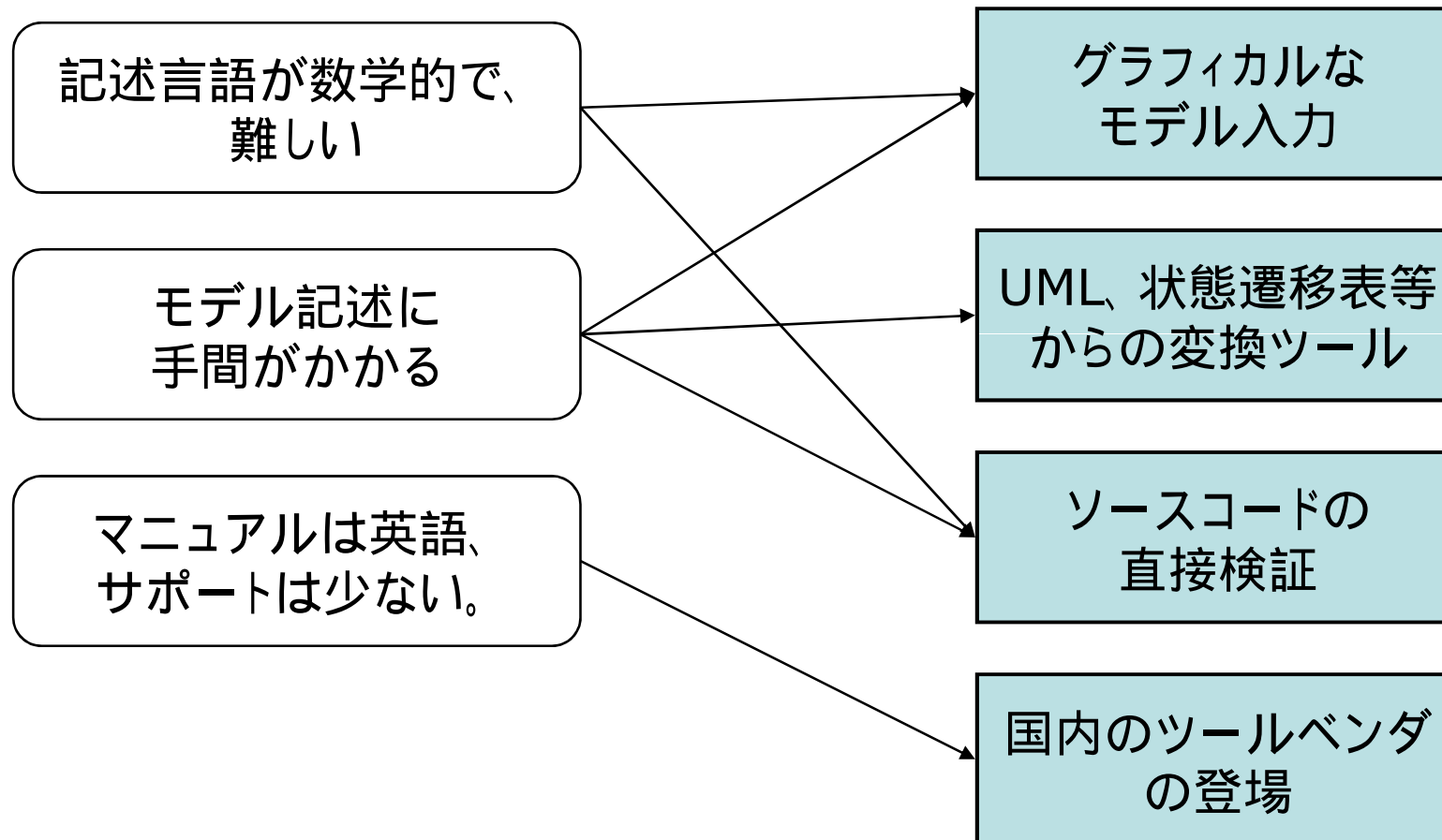
VARVEL

特徴	<ul style="list-style-type: none">●C言語で記述されたソースコードを対象とするモデル検査ツール。●静的なプログラム検証が可能。●検証条件は、無効ポインタ参照、配列境界違反などの4種類のみ。●今後、事前条件、事後条件、不変条件の記述をサポートする計画。
開発元	NECが開発し、社内で活用している。
資料入手先	<ul style="list-style-type: none">●http://www.nec.co.jp/techrep/ja/journal/g07/n02/070214.html●形式手法によるC言語検証ツールVARVEL、NEC技報 Vol.60 No.2/2007
ツール入手先	未発売。組み込みソフトウェア統合開発ツールに追加する予定がある。

国内での活用例

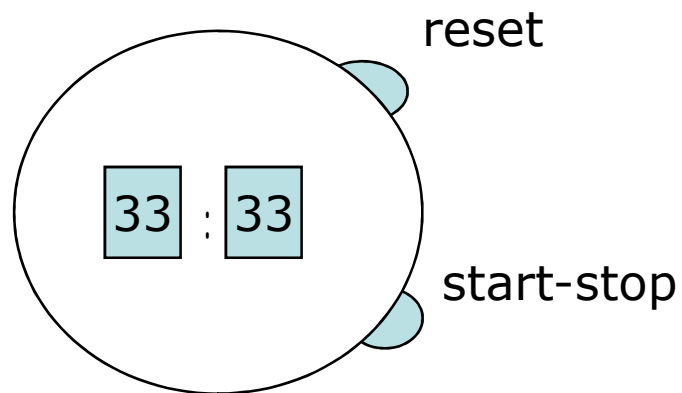
対象ソフトウェア	利用手法	概要
モバイル電子マネー	VDM	<ul style="list-style-type: none"> ◆目的は仕様記述のあいまいさの排除 ◆仕様記述には日本語、UMLを併用 ◆仕様設計段階で仕様ミスを指摘
設備保全システム	SMV	<ul style="list-style-type: none"> ◆目的は稼働中又は開発中の不具合解析 ◆ソースコード上で原因箇所を指摘
複写機	SPIN	<ul style="list-style-type: none"> ◆目的はバグ発見 ◆UML状態図から検査モデルへの変換ツールを開発
証券会社のバックオフィス	VDM	<ul style="list-style-type: none"> ◆目的は仕様記述のあいまいさの排除 ◆仕様記述にはUMLクラス図等を併用 ◆品質と生産性が向上
宇宙分野	SpecTRM	<ul style="list-style-type: none"> ◆目的は仕様ミスの発見 ◆自然語によるモデル記述ツールとその結果から検査モデルへの変換ツールを開発

形式手法の活用障壁とその改善



練習問題:仕様の検証

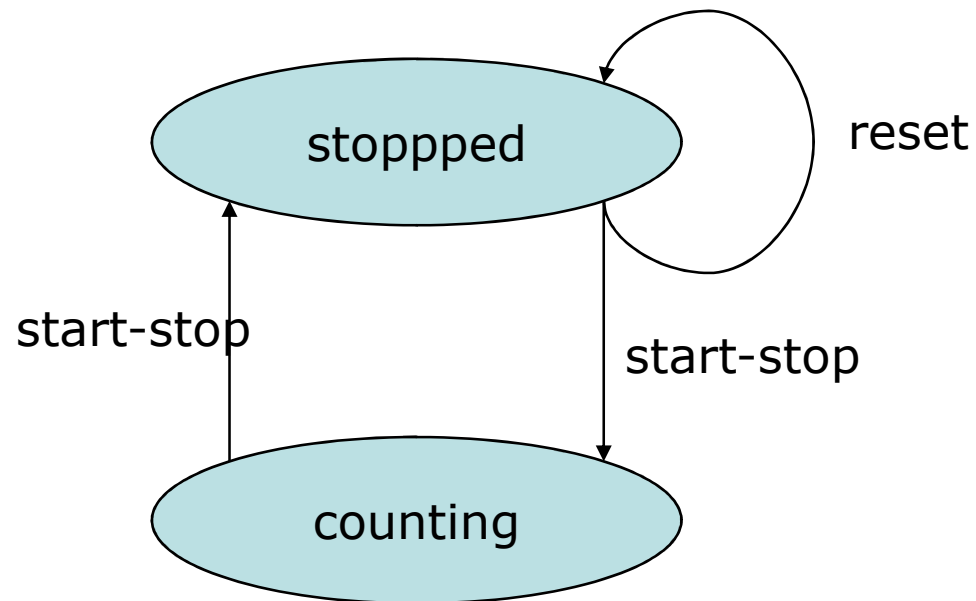
Stop-watch



要求仕様として:

- 1) S-Sボタンを押すと、カウントを開始し、もう一度押すと、停止する。
- 2) カウントが停止しているときにresetボタンを押すと、表示を初期化する。

最初モデリングと検証条件



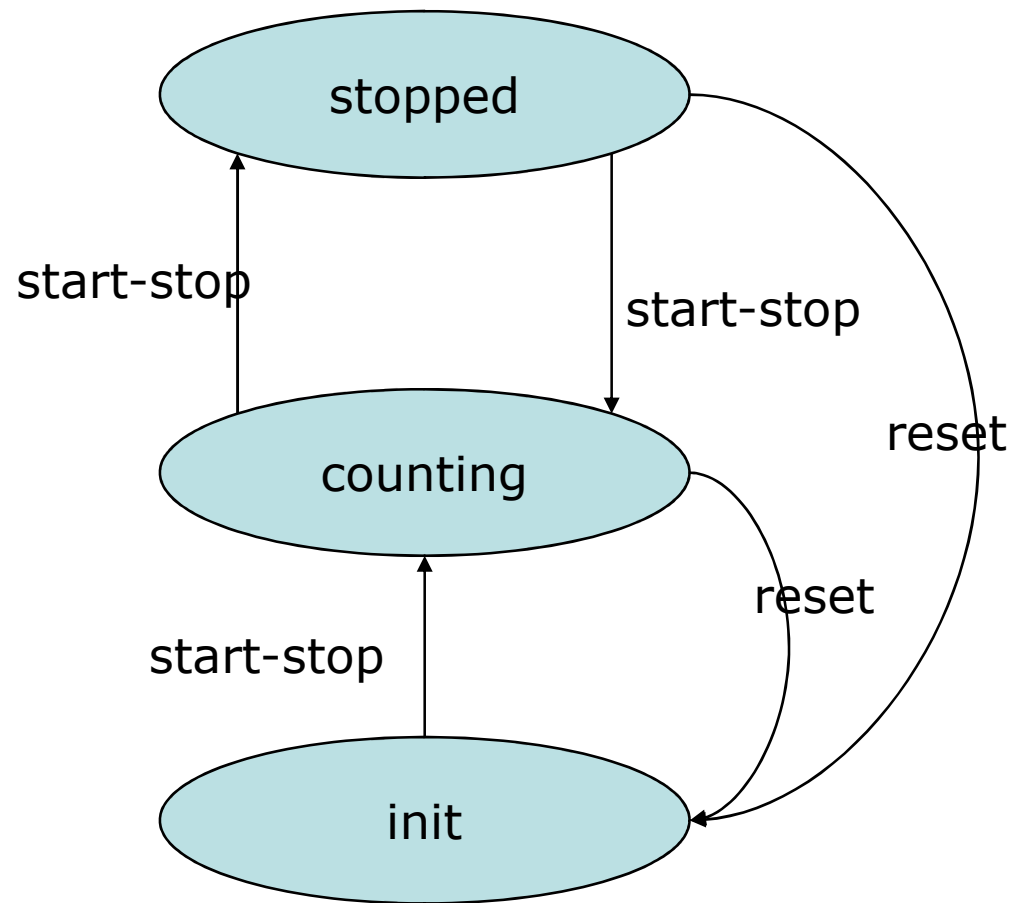
モデル検査すると、counting状態でresetを押しても、表示がゼロにならない反例が出る。

検証目的: resetを押せば、必ず表示がゼロになることを確認する。

検証条件として:

$G((\text{reset} = \text{true}) \quad (\text{表示} = \text{ゼロ}))$

モデルの見直しと仕様修正



検証条件が間違っていた。

しかし、要求仕様にも不備があることに気づく。

それで、仕様を次のように修正する：

カウント中でも、resetボタンを押せば、カウントが停止し、表示がゼロになる。

情報サイト

◆形式手法に関する資料を集めている仮想図書館
(Formal Methods Virtual Library)

■ <http://vl.fmnet.info/>

◆形式手法に関するフリー百科事典

■ http://en.wikipedia.org/wiki/Category:Formal_methods

■ http://en.wikipedia.org/wiki/Formal_methods

国内のツールベンダ等

- ◆ SCAD Eを販売しているCDAJ
 - <http://www.cdaj.co.jp/>
- ◆ VDM Toolsを販売しているCSKシステムズ
 - <http://www.csk.com/systems/>
 - <http://www.vdmtools.jp/>
- ◆ 入門講座を提供している豆蔵
 - <http://www.mamezou.com/training/ift.html>
- ◆ 設計コンサルテーションを行うイーソル
 - <http://www.esol.co.jp/rcs/index.html>
- ◆ UPPAALを販売提携したキャッツ
 - <http://www.zipc.com/>

国内の研究者、推進者等

- ◆ 産業技術総合研究所システム検証研究センター
 - <http://unit.aist.go.jp/cvs/>
- ◆ 国立情報学研究所
 - <http://www.nii.ac.jp/>
- ◆ モデル検査に関する研究会
 - <http://www.modelcheck.jp/>
- ◆ 九州大学
- ◆ 奈良及び北陸先端科学技術大学院大学
- ◆ JAXA

参考資料

◆いまさら聞けない形式手法入門、@IT

- <http://monoist.atmarkit.co.jp/fembedded/special/fm/fm01.html>

◆新言語進化論(連載)、ThinkIT

- <http://www.thinkit.co.jp/free/article/0711/6/2/>

引用文献

- [1] JIS規格書、電気・電子・プログラマブル電子安全関連系の機能安全、日本規格協会発行
- [2] 組込みソフトウェア2007:モデルに基づく開発方法論のすべて、日経BP社発行
- [3] 要件を厳密に定義する「フォーマルメソッド」の実際、日経ITプロフェッショナル2004年9月号、日経BP社
- [4] 特集「バグゼロを目指し脚光を浴びる形式手法」、日経コンピュータ2006年7月24日号、日経BP社
- [5] 仕様書の記述力を鍛える、日経エレクトロニクス2007年2月12日号、日経BP社
- [6] ソフトウェア信頼性向上の検証技術、組込みソフトウェア2006 p46-53、日経BP社
- [7] プログラム仕様記述論、荒木啓次郎他著、オーム社
- [8] 4日で学ぶモデル検査(初級編)、産業技術総合研究所システム検証研究センター著、NTS
- [9] 組み込みソフトウェアの設計 & 検証、藤倉俊幸著、CQ出版社

まとめ

- ◆ 形式手法はあいまいさの排除、上流工程での検証を可能とし、ソフトウェア品質を向上させる有力な手段である。
- ◆ 論理学や集合論等を手段として仕様を厳密に記述して検証する手法、システムの動作モデルを記述してその特定の性質を検査する手法等に分けられる。
- ◆ 産業界での適用事例の公表、機能安全IEC61508の浸透、商用ツールベンダの登場等で、難解で手間がかかる等の活用障壁は下がってきた。
- ◆ 産業界向けのツール、国内サポートサービス等を利用して形式手法への取組みを始めることが可能。