日本ソフトウェア科学会第 27 回大会（2010 年度）講演論文集

64bit 版モデル検査器とモデル検査 Web システム

篠崎 孝一　早水 公二

形式手法の１つであるモデル検査は，産業界からソフトウェア開発の品質向上に有効な手法として注目されている．しかし，研究・評価用のモデル検査器が公開されているだけで，インストールや状態爆発への対応が必要であり，様々なユーザの検証ニーズや制約条件に対応できるサービスとしては考えられていない．そこでモデル検査の利便性を上げる取り組みとして，誰もが Web サイトから直接，強力なモデル検査器を利用できるシステムを開発中である．システムはメモリ空間を拡張するために 64bit 用に新たに開発したモデル検査器を，産業技術総合研究所連携検証施設「さつき」の大容量メモリ高速演算クラスタ上で実行することで，状態爆発の回避を考慮している．

1　はじめに

モデル検査はシステムが取り得る状態を網羅的に検査して不具合や予期せぬ動作を発見する手法である．ソフトウェアの全数検査を実現することができるため，品質向上のための新しい手段として近年，情報システム開発企業の注目を集めている．

我々は 2002 年からモデル検査に着目し，ソフトウェア開発における実用化・実践適用に取り組んできた．既にメルコ・パワー・システム株式会社では，自社で開発したソフトウェアへの適用事例を 30 件以上も蓄積しており，多くのシステムでモデル検査を使って不具合の発見あるいは原因究明に成功したため，設計者，開発者からは高い評価を得ている．これらの事例は，モデル検査の普及に役立つものと考えて，シンポジウムや Web ページにより公開している[1][2][3][4][5]．

Koichi SHINOZAKI, 関西電力株式会社電力技術研究所, Power Engineering R&D Center, Kansai Electric Power Co., Inc.
Koji HAYAMIZU, メルコ・パワー・システムズ株式会社, Melco Power Systems Co.,Ltd.

一方，企業において新しい技術を導入する場合，技術習得の難易度から社会的信頼まで様々な配慮が求められる．近年，国内では企業の技術者を対象としたモデル検査セミナー[6][7]が開催されるようになり技術習得は容易になってきた．しかし，海外の大学や研究会が評価・研究用に公開しているソフトウェア（モデル検査器）を実務に用いるライセンス上の不安，企業内 PC へのインストールやメンテナンス不在によるセキュリティへの不安，さらに状態爆発への対応といった多くの課題が残されている．そこで，我々はこれらの課題解消と一層の利便性向上を図るために 64bit 版モデル検査器の開発を行い，さらに Web サイトから強力なモデル検査環境として利用できるモデル検査 Web システムを開発中である．

本システムは，64bit 版 OS 対応により利用可能なメモリ領域を拡大した強力なモデル検査器を，産業技術総合研究所連携検証施設「さつき」（以下，「さつき」と記す）の大容量メモリ高速演算クラスタに搭載した上で，モデル検査を Web サービス化するもので，ライセンス等の課題を解決すると共に，産業

界へのモデル検査普及にも役立つと期待している．

2 64bit版モデル検査器の開発

モデル検査の評価が高まるにしたがって，開発現場からは数百頁規模の仕様書や，数万行のソースコードといった，より大きなシステムを検査対象としたいとの要望が増えてきている．

一方，モデル検査は全数検査を行うが故に状態爆発と呼ばれる問題を抱えている．状態爆発は，検査対象の状態空間の指数的な増加に伴って，現実的な時間で検査が終了しない，あるいは計算機のメモリを消費し尽くして仮想メモリを使い始めることで処理が極端に遅くなってしまう問題である．特にメモリ使用量については，現在公開されているモデル検査器のほとんどが 32bit 版 OS 対応のため，4(GB)のメモリしか利用できない制限がある．このため，検査対象の状態空間を小さくする抽象化に相当な労力を要する．そこで我々は，利用可能なメモリ領域を大幅に拡大した 64bit 版モデル検査器を，基本設計から始めて新規開発した．既存のモデル検査器の改良ではなく，新規に開発することでソフトウェアを商用利用する際に直面するライセンス問題や，将来的な改変による影響を回避できるメリットもある．さらに，独自のアルゴリズムを採用することでメモリの効率化と処理の高速化に取り組んだ．

2.1 データ構造（BDDの採用）

開発した 64bit 版モデル検査器では，状態遷移系で表現されたモデルをメモリ上に展開する際のデータ構造として BDD (Binary Decision Diagram) を採用している．BDD は二分決定グラフとも呼ばれ，ブール変数のみからなる論理式を，1 つの節点を根(root)とする非環状のグラフの形で表現したものである．

例として，論理式「a&c|b&d」の BDD 表現を図 1 に示す．節点から下方に引く 2 本の矢印線は，左側がブール変数の値が 0 の場合(0 枝)，右側が 1 の場合(1 枝)を表現している．この例では，変数の順序が上位から a→b→c→d となっており終端節点 0，1 を除いた節点数は 6 である．ここで，変数の順序を上位から a→c→b→d となるように変更して，同じ論理式を BDD で表現すると図 2 となり，節点数は 4 となる．このように，BDD では同じ論理関数でも変数の順序を変更するだけでグラフとして表現する際に必要な節点数が変化する．節点数を減少できれば，処理に必要なメモリが減少してメモリ使用量を効率化でき，処理時間短縮も期待できる．BDD を用いたモデル検査器としては，カーネギー・メロン大学の SMV[8]を始め，Cadence SMV[9]，NuSMV[10]が知られているが，BDD の節点数削減や処理の高速化に役立つ様々なアルゴリズムや経験則が存在し，これらを利用することで一層の性能向上が期待できる BDD ベースのモデル検査器を開発することとした．検査項目は，BDD 上での検査に向いている CTL(Computation Tree Logic)式により記述する．

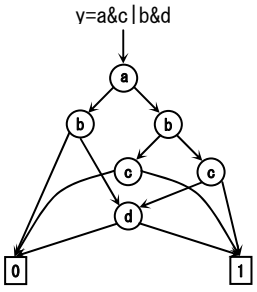

図 1 y=a&c|b&d の BDD(その 1)

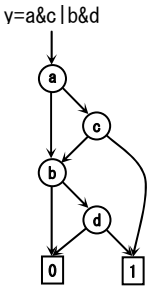

図 2 y=a&c|b&d の BDD(その 2)

2.2 メモリの効率化と処理の高速化

64bit 版モデル検査器にはメモリ使用量を効率化する，あるいは，処理を高速化する幾つかのアルゴリズムを実装している．これらアルゴリズムの効果は，常に万能ではなく検査するモデルに影響されることから，試験的に実装した上で，実例に基づいた評価用モデルを使ったモデル検査を行い，効果が高いものを採用している．さらに複数のアルゴリズムを採用することで，それぞれを単体で利用するよりも，さらに大きな相乗効果が得られている．実装したアルゴリズムの詳細は次節以降で説明する．

2.3 変数のグルーピング

前述したように BDD には，節点数の削減を可能にする経験則がある．以下に示す 2 つの経験則に従って BDD を構築すると節点数を減少できることが知られている．

1. 互いに関係の深い変数を隣接させる．
2. 論理式の値に大きな影響を与える変数を上位レベルに配置する．

「変数のグルーピング」は，上記 2 つの経験則のうち，経験則 1 に従って，モデルの中で宣言した変数の依存関係を考慮して隣接させる，すなわちグルーピングすることで，節点数を削減する手法である．モデルとは状態遷移系を，専用言語によってプログラム形式で記述したファイルである．図 3 にモデルの例を示す．変数 num は 1 から 8 の値を取るので，BDD を構築する際には，3 つのブール変数にビット分割される．

本手法では，変数同士の代入あるいは条件式での参照など，相互に関係し合う変数群をグルーピングする．グルーピングの例を図 4 に示す．

図 4 において，矢印の実線で繋がった 2 つの変数は，矢印の方向に代入あるいは参照があることを示す．この例では，グループ 1 とグループ 2 ができることを示している．グループ間の上下関係は，所属する変数の数が多いグループほど上位とする．例では，グループ 2 に所属する変数(4 変数) > グループ 1 に所属する変数(3 変数)であるので，グループ 2 を上位とする．具体的には，グループ 2 に所属する変数（正確にはビット分割後の変数）を BDD の上位に配置し，グループ 1 に所属する変数を下位に配置する．図 1 と図 2 の例で示したように，経験則 1 に従って変数の配置を変更するだけで，BDD の節点数を削減することができる．

```
MODULE main
VAR /* 変数宣言部 */
 num : 1..8;
 fg  : boolean;
 cpu : {stop, busy, wait};
 pow : 1..3;
  ⋮
ASSIGN /* 状態遷移 */
 init(num) := 1;
 next(num) := case
  fg=0 & num<8 : num+1;
    ⋮
 esac;
        ⋮
```

図 3 モデルの例

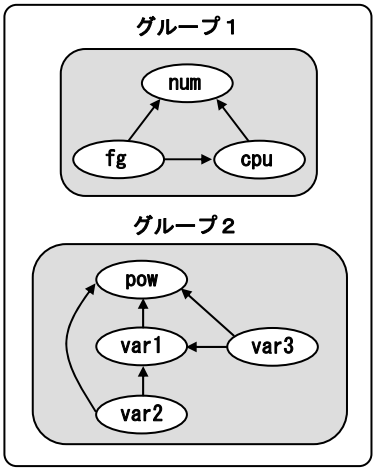

図 4 変数のグルーピング

2.4 変数の並び替え

「変数の並び替え」は，経験則 2 に従ってモデルの中で宣言した変数の順序を入れ替えることで，BDD の節点数を削減する手法である．手順を下記する．

1. 変数の出現回数を算出する．
2. 出現回数の降順に変数を並び替える．

変数の出現回数の算出では，モデルの中で当該変数が何回登場するかを数える．そして，出現回数の降順に変数を並び替えて「変数の順序」を決定する．出現回数の多い変数は，モデル全体の「論理式の値に大きな影響を与える変数」であると考える．本手法は「変数のグルーピング」と組み合わせて利用することで大きな相乗効果を発揮する．つまり，「変数のグルーピング」でグループとしての上下関係を決定した後に，本手法によってグループ内の変数の順序を決定することで，BDD の節点数を削減する 2 つの経験則を共に活用することができる．

2.5 不要変数の除外

モデル検査ではモデルが取り得る全ての状態を網羅的に検査するが，検査式である CTL 式に登場する変数に影響を与えない変数は，その挙動を無視することができる（CTL 式の変数が他の変数に影響を与えるか否かは問わない）．「不要変数の除外」は，挙動を無視できる変数を最初から BDD 構築の対象外とすることで節点数を削減する手法である．削減の手順を以下に示す．

1. CTL 式に登場する変数を列挙する．
2. モデル内での変数関係図を作成する．
3. 手順 1 の変数に影響を与えない変数を除外する．

手順 2 の変数関係図は「変数のグルーピング」で作成した依存関係の図（図 4）を用いれば良い．

例として，図 4 のモデルで CTL 式「AG(var2=0 -> AF(var1=1))」を検査する場合を考える．手順 1 では変数 var1 と var2 を列挙できる．手順 2 の変数関係図は図 4 そのものである．手順 3 で変数 var1 と var2 に影響を与えない変数として，変数 pow と，変数 num, fg, cpu が挙げられる．変数関係図で変数 A に影響を与えない変数とは，変数 A に向かう矢印が存在しない変数である．この例では，合計 4 つの変数を BDD 構築の対象外とすることができるため，その分，BDD の節点数を削減できる．なお，検査式が複数ある場合には，検査式毎に本手法を適用してモデルを小さくしてから検査することで，より大きな効果を得ることができる．

2.6 不要節点の削除

「不要変数の削除」ではモデルで宣言された変数そのものを BDD 構築の対象外とした．一方，本手法では構築後の BDD から不要な節点を洗い出して削除することで節点数を削減する．BDD 同士の論理積，論理和等を計算する過程で不要な節点が出現することを利用している．一例として論理式「a|b」を表す BDD と論理式「a|!b」を表す BDD の論理積を計算する場合を考える．計算によって論理式「(a|b)&(a|!b)」を表す BDD が構築されるが，本論理式は変数 a の値によってのみ真偽値が決定されるため変数 b は不要である．BDD による計算結果を図 5 に示す．図 5 より節点 b が不要となることがわかる．削減の手順を以下に示す．

1. 全ての BDD 節点をチェックし，0 枝と 1 枝の指す節点を列挙する．
2. 手順 1 で列挙した節点以外の節点を削除する．

手順 1 で他の節点から参照されている，すなわち必要な節点を列挙し，手順 2 で必要な節点以外=不要な節点を削除する．

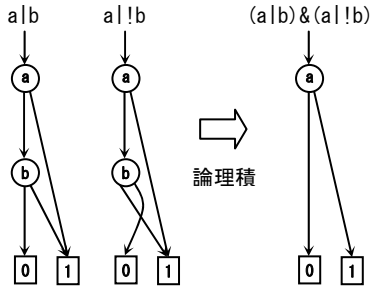

図 5 BDD による計算結果

2.7 演算結果の記憶

「演算結果の記憶」は，過去に行った BDD の節点同士の演算結果を記憶しておき，次に同じ節点同士の演算が必要な場合に，実際の演算を行わずに記憶した結果を利用する手法である．本手法は節点の情報を余分に記憶するため，メモリの使用量が増加するデメリットがあるが，処理全体の高速化を図ることができるため，検査時間短縮のメリットが得られる．

2.8 Conjunctive Partitioning

64bit 版モデル検査器では，通常処理の場合，モデルに記述された変数毎の状態遷移系を BDD で表現した後に，それらの論理積を求めてモデル全体すなわちシステム全体の状態遷移系を構築する．しかしながら，この方法では変数の数が増えるに従って，あるいは，変数毎の状態遷移系が大きくなるに従って，全体の遷移系も肥大化し大量のメモリを必要とする．モデルにもよるが，α，β，γの BDD の論理積を表す BDD をΣとしたとき，Σの節点数は，α，β，γの単体での節点数の総和よりはるかに大きくなる場合が多い．「Conjunctive Partitioning」は全体の遷移系を表す BDD を構築せずに，変数毎，あるいは一部の変数の論理積を表す BDD だけを求めて演算を進める手法である．BDD に対する演算を f()で表したとき，f(Σ)は f(α・β・γ)と書ける．このとき，f(α・β・γ) = f(α)・f(β)・f(γ)が成立するならば，Σを求めることなく，α，β，γの単体の BDD に対して演算を行うことができる．本手法は，上記の等式が成立するようなα，β，γがモデルに存在し，演算 f()を実施する必要がある場合に適用することができる．一般に，Conjunctive Partitioning を適用すると，合計の演算時間は長くなる傾向があるが，Σの BDD の節点数が巨大な値となるような場合は，それを構築する必要がない本手法は BDD の節点数削減に非常に有効である．

3 64bit版モデル検査器の動作確認

開発した 64bit 版モデル検査器は，基本的な動作を既存の 32bit 版モデル検査器との比較により確認している．また，試験用モデルを用いて，モデル検査の出力結果と手計算による結果が一致すること，反例が出力された場合は妥当な反例であることを確認している．さらに，超大容量メモリ環境での動作を確認するため，「さつき」の大容量メモリ高速演算クラスタを用いた検証実験を行った．同クラスタはメモリを 1(TB)搭載しており，64bit 化したモデル検査器の動作検証に適している．実験では，過去の適用事例用に作成したモデルを部分変更して状態数の非常に大きな検証用モデルを作成し，モデル検査実行中の BDD 節点数の増加状況を調べた（図 6）．実験の結果，検査結果を得ることは出来なかったが順調に節点数を増やして，約 1771 分で約 325 億個の BDD 節点の構築に成功し，メモリ 1(TB)を問題なく利用できることを確認した．これまでの経験から，BDD 節点数はモデルの取り得る状態数としては，数百兆状態から数京状態に該当すると考えられる．

4 64bit版モデル検査器の適用事例

64bit 版モデル検査器を実製品のソフトウェア(C 言語で記述された約 1000 行のプログラム）のモデル検査に適用した事例を紹介する．従来の 32bit 版モデル検査器で検査しても，4(GB)のメモリを消費し尽くし検査結果が得られていない事例である．

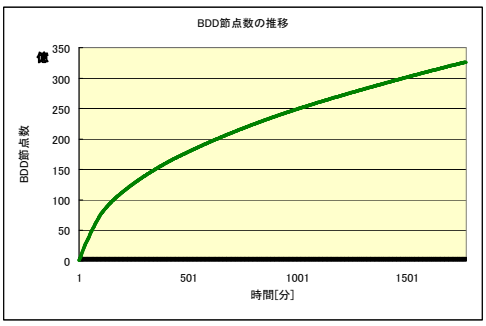

図 6 BDD の節点数と時間のグラフ

この事例について，２章で述べたメモリの効率化や処理の高速化の手法を適用した場合と，全く適用しない場合のモデル検査を行い，BDD の節点数の増加傾向を比較した結果を図 7 に示す．効率化無しの場合は，約 600 分を経過した時点で検査結果が得られる見込みがないため強制終了した．効率化有りの場合は，約 98 分で約 8 億個の BDD 節点を構築し，検査結果を得ることができた．メモリは約 52(GB)利用した．さらに，出力された反例を解析することで適用したプログラムの不具合を発見できた．本事例により，32bit 版モデル検査器で発生した状態爆発を 64bit 版モデル検査器で回避できることが確認された．

5 モデル検査Webシステムの開発

5.1 モデル検査Webシステム開発の背景

３章の 64bit 版モデル検査器の動作確認で利用した「さつき」は，組込みシステムの信頼性向上を目的として，検証環境が提供されているもので，共同研究契約を締結してオンサイトでクラスタ計算機資源の利用や検証サービスの利用が可能である．

一方，企業におけるモデル検査の活用シーンを考えると，何時でも何処でも手軽に利用したいニーズが多いと考えられる．モデル検査を Web サービス化すると，これらのニーズに対応できる上に，モデル検査器のライセンスやインストールの問題が解消できる．

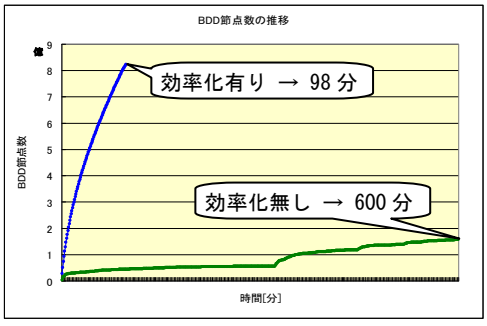

図 7 BDD の節点数と時間の関係(適用事例)

さらに，これからモデル検査を学ぶ学習者や導入活用の可能性を検討・評価する企業にとっても利便性が高いサービスと考えられる．

そこで，「さつき」を運営する産業技術総合研究所組込みシステム技術連携研究体の協力を得て，64bit 版モデル検査器を「さつき」環境で利用できる Web サービスを実現するモデル検査 Web システムの開発に着手した．

5.2 モデル検査Webシステムの概要

モデル検査 Web システムは，2010 年 8 月末に一般ユーザ向けに公開／サービス運用開始することを目指して開発を進めている．ユーザは，図 8 に示すようにインターネット上に公開予定の Web サイトから本システムにアクセスしてサービスを利用できる．システムが提供するサービスの流れを以下に示す．

1. ユーザが検査したいモデルを送信する．
2. 「さつき」上の 64bit 版モデル検査器がモデルを自動的に検査する．
3. ユーザがインターネットを介して検査結果を取得する．

モデルは 64bit 版モデル検査器が対応している SMV 言語により記述し，検査項目は時相論理式(CTL)により記述するが，モデル検査の学習でハードルの一つとなる検査式（CTL）は，Web サイト上の画面から入力支援することを検討している．

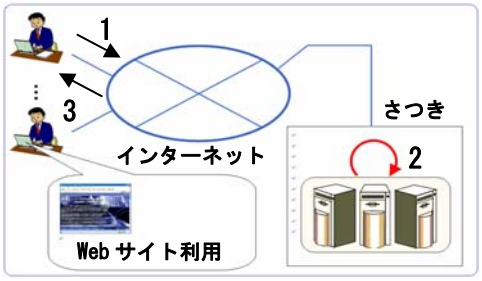

図 8 モデル検査 Web システムのイメージ図

一般企業からの利用を考慮して，「システム内部にユーザを特定・推測できるデータ，検査モデル，検査結果の記録を一切残さない」かつ「受付から検査完了／通知まで全て人手が関与せず自動的に処理が流れる」ことを開発方針とし，データの通信経路のセキュリティ強化にも努めている．また，万一の場合に備えて，モデルの変数名が意味を持たないように匿名化することをユーザに推奨していく．

サービス開始に伴い，モデル検査 Web システムは大小様々なモデルを扱うことになるが，モデル検査では，実行前や実行途中段階において検査完了までの処理時間を想定することができない．そこで，有限な計算環境を多人数で共同利用する方法として「１サービス当りの処理時間設定と超過分の打ち切り」や「処理時間予約制」等を検討している．

本 Web サービスのみでは，全てのモデル検査ニーズに応えることは不可能であり，「モデルがうまく書けない」「状態爆発が収まらない」といったユーザの支援には，産業技術総合研究所の検証サービスを，処理時間が足りないユーザには，共同研究契約による「さつき」のオンサイト利用を，それぞれ Web サイト上で推奨して，適切に分業化を図る．

5.3 Webサービスのメリット・デメリット

本システムを利用することで，ユーザは社内の PC 環境における問題（性能，セキュリティ，運用制約）を解消して，モデル検査器のインストールの手間も掛けず，海外サイトのライセンス条項にも頭を痛めず，モデル検査を体験・実践することが可能となる．

さらに，Web アクセスだけで，状態爆発にも強い 64bit 版モデル検査器を「さつき」の大容量メモリ高速演算クラスタ上で利用できるメリットは大きい．

一方，検査モデルを送信しても処理時間超過により結果が得られない場合には，検査項目やモデルの変更が必要であるが，セキュリティの視点から Web システム側では人手による変更支援サービスは提供されず，全て，ユーザでの作業となる．

また，情報システムのセキュリティ問題，企業情報の Web 流出事故が注目される社会状況下で，開発システムのモデルを社外の環境で検査すること自身をセキュリティリスクと見られる可能性は残る．

しかし，名前等の属性がなく，変数名も匿名化されたモデルから企業名や開発内容を読解することは困難であることから，モデル検査の Web サービス利用に対してユーザの理解が得られると考えている．

6 おわりに

モデル検査の課題の１つである状態爆発の回避を主たる目的として 64bit 版モデル検査器を開発した．開発したモデル検査器は，産業技術総合研究所連携検証施設「さつき」の大容量メモリ高速演算クラスタ上で動作確認を行ったうえ，適用事例によって状態爆発の回避に有効であることを確認した．

両者の連携によるメリットを広く産業界に提供するべく，誰もがアクセスできるモデル検査 Web システムを開発中である．

64bit 版モデル検査器とモデル検査 Web システムが，モデル検査の産業界への普及促進・利用拡大に役立つことを期待している．

謝辞モデル検査器のアルゴリズムについて京都産業大学コンピュータ理工学部の平石裕実教授から御指導を頂きました． 厚く御礼申し上げます．また，「さつき」を利用するにあたって，多大なご協力とご指導を頂きました産業技術総合研究所組込みシステム技術連携研究体(http://cfv.jp/cvs/)の皆様に感謝いたします．

参考文献

[1] 篠崎孝一，水口大知，石井健志：組み込みソフトウェア開発のイン-デザインモデル検査，ソフトウェアテストシンポジウム 2004 予稿集， Jan.2004，pp. 81-84.

[2] 篠崎孝一，太田弘，早水公二，星野光勇：モデル検査のデバッグへの適用，ソフトウェアテストシンポジウム 2006 in 東京予稿集，Jan.2006，pp. 113-117.

[3] 篠崎孝一，早水公二：モデル検査の開発現場への導入，計測自動制御学会誌「計測と制御」第 48 巻 第 11 号 2009 年 11 月号，Nov. 2009，pp. 828-833.
[4] 早水公二，篠崎孝一，高橋孝一，渡邊宏：モデル検査器を用いた自動検針システムの仕様検証，産業技術総合研究所算譜科学グループ研究速報，AIST-PS-2003-005，June 2003.
[5] 早水公二，篠崎孝一，星野光勇：モデル検査の実用化に向けた取り組みと事例報告，第二回システム検証の科学技術シンポジウム予稿集，AIST-PS-2005-017，Oct 2005，pp. 79-86.
[6] 組込みシステム産業振興機構：組込み適塾（http://www.kansai-kumikomi.net/）
[7] 国立情報学研究所：トップエスイープロジェクト，（http://www.topse.jp/）
[8] Carnegie Mellon University, The SMV system
(http://www.cs.cmu.edu/~modelcheck/smv.html/)
[9] Ken McMillan, Cadence SMV
(http://www.kenmcmil.com/)
[10] NuSMV, (http://nusmv.irst.itc.it/)