NuSMV Seminar 3: Internals, Algorithms, nuXmv

Overview, SAT-based Bounded MC, nuXmv Intro

Di Marco, Okwieka

University of Rome "La Sapienza"

May 15th, 2023

NuSMV Architecture Overview

NuSMV 2 Evolution

Cimatti et al. (2002): *Integrating BDD-based and SAT-based Symbolic Model Checking*

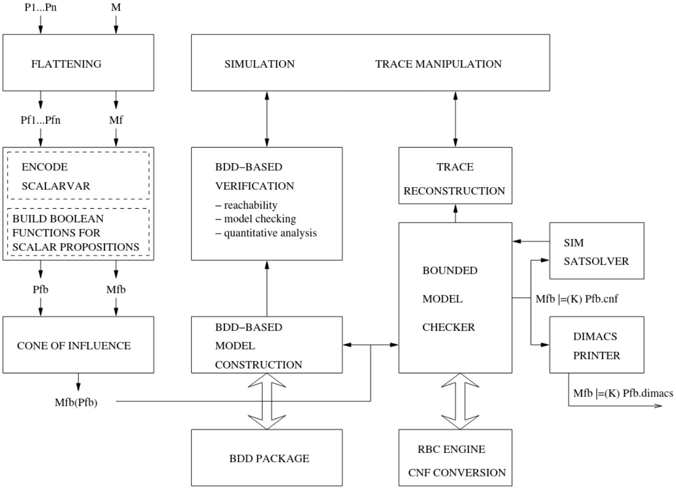
Original BDD-based SMV at CMU (Carnegie Mellon University) as PhD thesis of Ken L. McMillan (1992)

NuSMV 1 (1999): reimplementation with added LTL, interactive mode, invariants, model partitioning

NuSMV 2 (2002): Open-Source evolution, with Bounded Model Checking across whole input language

NuSMV Architecture

Cimatti et al. (2002): *Integrating BDD-based and SAT-based Symbolic Model Checking*



NuSMV Model Construction

Cimatti et al. (2002): *Integrating BDD-based and SAT-based Symbolic Model Checking*

Input: model, set of propositions

Three steps common for both BDD-based and Bounded Model Checking

1. **Flattening:** Parse, type-check, definition cycles check, instantiate modules and processes
2. **Boolean Encoding:** Encode scalars as set of Boolean variables
3. **Cone of Influence:** Prune model to parts relevant for propositions

Output: finite state machine (initial states, invariant states, transition relation)

NuSMV BDD-based Symbolic Model Checking

BDDs sensitive to variable ordering

$\Longrightarrow$ NuSMV allows specifying manual ordering, or static heuristics

Model can be *partitioned* into conjunction of set of BDDs using various heuristics

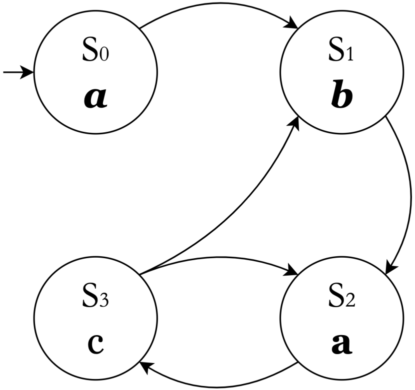
$\Longrightarrow$ mitigates space-use explosion

BDDs with fixpoint algorithms used for:

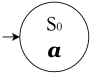
- Reachability Analysis
- Fair CTL Model Checking
- LTL Model Checking via Tableau Construction to CTL

SAT Bounded Model Checking

Bounded Model Checking (1)

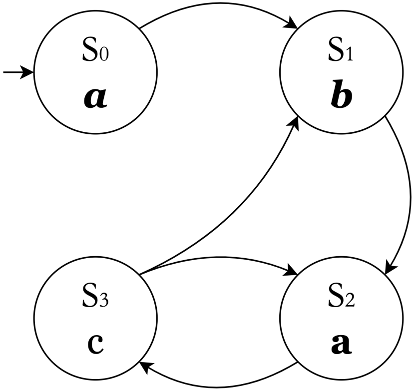


- $\varphi := \mathbf{G}\,(\,a \longrightarrow \mathbf{F}\,b\,)$
- $\neg\varphi := \mathbf{F}\,(\,a \wedge \mathbf{G}\,\neg b)$
- $k = 0$

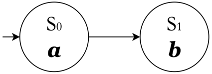


- No counterexample found.

Bounded Model Checking (2)

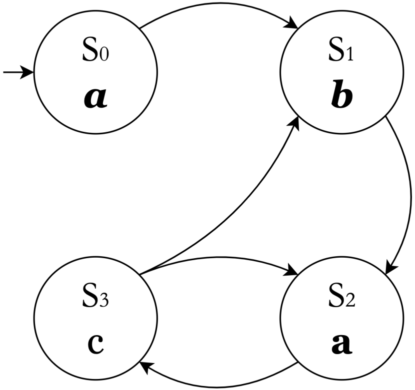


- $\varphi := \mathbf{G}\,(\,a \longrightarrow \mathbf{F}\,b\,)$
- $\neg\varphi := \mathbf{F}\,(\,a \wedge \mathbf{G}\,\neg b)$
- $k = 1$

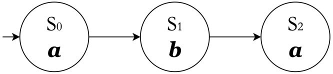


- No counterexample found.

Bounded Model Checking (3)

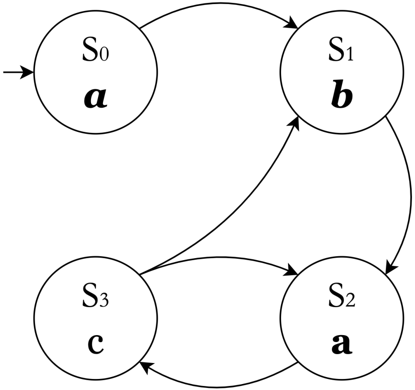


- $\varphi := \mathbf{G}\,(\,a \longrightarrow \mathbf{F}\,b\,)$
- $\neg\varphi := \mathbf{F}\,(\,a \wedge \mathbf{G}\,\neg b)$
- $k = 2$

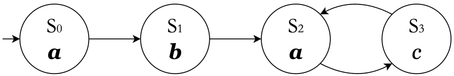


- No counterexample found.

Bounded Model Checking (4)



- $\varphi := \mathbf{G}\,(\,a \longrightarrow \mathbf{F}\,b\,)$
- $\neg\varphi := \mathbf{F}\,(\,a \wedge \mathbf{G}\,\neg b)$
- $k = 3$



- **Loopback** $\Rightarrow$ Counterexample!

SAT-based Bounded Model Checking

- Look for **counterexample paths** of increasing length k.
- For each k, build a Boolean formula that is satisfiable iff there is a *witness* of length k.
  - Formula construction is *not* subject to state explosion.
  - Linear in the number of variables and steps.
- Satisfiability of the Boolean formulas is checked using a **SAT solver**
  - Can manage complex formulae on several variables
  - Returns a **satisfying assignment** (i.e., a *counterexample*)

SAT encoding (1)

Ingredients:

- Kripke structure $\mathcal{M} := < \mathcal{S}, \mathcal{I}, \mathcal{R}, \mathcal{L} >$
- Upper Bound $k \geq 0$
- LTL property φ

Is there a a partial execution (trace) of k steps that satisfies φ?

- We want to find

$$\underbrace{[\![\mathcal{M}, \varphi]\!]_k}_{\text{SAT encoding}} \Rightarrow \underbrace{s^0, s^1, \ldots, s^k}_{\text{witness}}$$

SAT encoding (2)

$$[\![\mathcal{M}, \varphi]\!]_k := [\![\mathcal{M}]\!]_k \wedge [\![\varphi]\!]_k \tag{1}$$

- $[\![\mathcal{M}]\!]_k$ encodes the fact that the k-path is a **legal trace of** $\mathcal{M}$
- $[\![\varphi]\!]_k$ encodes the fact that the k-path **satisfies** φ

SAT legal trace encoding

$$[\![\mathcal{M}]\!]_k := \mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(s^i, s^{i+1}) \tag{2}$$

SAT property encoding (1)

$$[\![\varphi]\!]_k := \overbrace{\left(\neg \bigvee_{L=0}^{k} \mathcal{R}(s^k, s^L) \wedge [\![\varphi]\!]_k^0\right)}^{\psi_1} \vee \overbrace{\left(\bigvee_{L=0}^{k} \left(\mathcal{R}(s^k, s^L) \wedge {}_L[\![\varphi]\!]_k^0\right)\right)}^{\psi_2} \qquad (3)$$

SAT property encoding (2)

$$\psi_1 := \neg \bigvee_{L=0}^{k} \mathcal{R}(s^k, s^L) \wedge [\![\varphi]\!]_k^0 \tag{4}$$

- ψ_1 is the constraint needed to express a model with **without loopback**.
- $[\![\varphi]\!]_k^i$ with $i \in [0, k]$ encodes the fact that φ holds in s^i under the assumption that $s^0, \ldots, s^k$ is a no-loopback path.

SAT property encoding (3)

$$\psi_2 := \bigvee_{L=0}^{k} \left(\mathcal{R}(s^k, s^L) \wedge {}_L[\![\varphi]\!]_k^0 \right) \tag{5}$$

- ψ_2 is the constraint needed to express a **loopback** in the model to all possible points in the past.
- ${}_L[\![\varphi]\!]_k^i$ with $i \in [0, k]$ encodes the fact that φ holds in s^i under the assumption that $s^0, \ldots, s^k$ is a path *with* a loopback from s^k to s^L.

$[\![\varphi]\!]_k^i$ SAT encoding (1)

- If:

$$\varphi := \alpha$$

- Then:

$$[\![\varphi]\!]_k^i := \alpha^i$$

- If:

$$\varphi := \neg\alpha$$

- Then:

$$[\![\varphi]\!]_k^i := \neg\alpha^i$$

$[\![\varphi]\!]_k^i$ SAT encoding (2)

- If:

$$\varphi := \alpha \wedge \beta$$

- Then:

$$[\![\varphi]\!]_k^i := [\![\alpha]\!]_k^i \wedge [\![\beta]\!]_k^i$$

- If:

$$\varphi := \alpha \vee \beta$$

- Then:

$$[\![\varphi]\!]_k^i := [\![\alpha]\!]_k^i \vee [\![\beta]\!]_k^i$$

$[\![\varphi]\!]_k^i$ SAT encoding (3)

- If:

$$\varphi := \mathbf{X}\,\alpha$$

- Then:

$$[\![\varphi]\!]_k^i := \begin{cases} [\![\alpha]\!]_k^{i+1} & \text{if } i < k \\ \bot & \text{otherwise} \end{cases}$$

$[\![\varphi]\!]_k^i$ SAT encoding (4)

- If:

$$\varphi := \mathbf{G}\,\alpha$$

- Then:

$$[\![\varphi]\!]_k^i := \perp$$

$[\![\varphi]\!]_k^i$ SAT encoding (5)

- If:

$$\varphi := \mathbf{F}\, \alpha$$

- Then:

$$[\![\varphi]\!]_k^i := \bigvee_{j=i}^{k} [\![\alpha]\!]_k^j$$

$[\![\varphi]\!]_k^i$ SAT encoding (6)

- If:

$$\varphi := \alpha \, \mathbf{U} \, \beta$$

- Then:

$$[\![\varphi]\!]_k^i := \bigvee_{j=i}^{k} \left([\![\beta]\!]_k^j \wedge \bigwedge_{n=i}^{j-1} [\![\alpha]\!]_k^n \right)$$

${}_L[\![\varphi]\!]_k^i$ SAT encoding (1)

- If:

$$\varphi := \mathbf{X}\,\alpha$$

- Then:

$${}_L[\![\varphi]\!]_k^i := \begin{cases} {}_L[\![\alpha]\!]_k^{i+1} & \text{if } i < k \\ {}_L[\![\alpha]\!]_k^L & \text{otherwise} \end{cases}$$

${}_L[\![\varphi]\!]_k^i$ SAT encoding (2)

- If:

$$\varphi := \mathbf{G}\,\alpha$$

- Then:

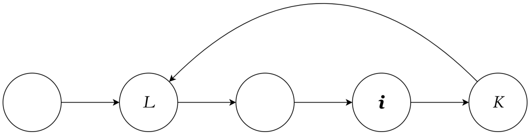
$${}_L[\![\varphi]\!]_k^i := \bigwedge_{j=min(i,L)}^{k} {}_L[\![\alpha]\!]_k^j$$

- If:

$$\varphi := \mathbf{F}\,\alpha$$

- Then:

$${}_L[\![\varphi]\!]_k^i := \bigvee_{j=min(i,L)}^{k} {}_L[\![\alpha]\!]_k^j$$



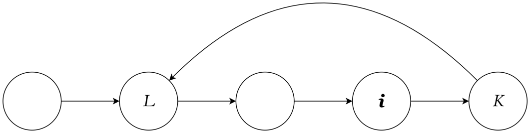
${}_L[\![\varphi]\!]_k^i$ SAT encoding (3)

- If:

$$\varphi := \alpha\, \mathbf{U}\, \beta$$

- Then:

$$ {}_L[\![\varphi]\!]_k^i := \bigvee_{j=i}^{k} \left({}_L[\![\beta]\!]_k^j \wedge \bigwedge_{n=i}^{j-1} {}_L[\![\alpha]\!]_k^n \right) \vee \bigvee_{j=L}^{i-1} \left({}_L[\![\beta]\!]_k^j \wedge \bigwedge_{n=i}^{k} {}_L[\![\alpha]\!]_k^n \wedge \bigwedge_{n=L}^{j-1} {}_L[\![\alpha]\!]_k^n \right)$$



Relevant Sub-cases

Reachability

- If:

$$\varphi := \mathbf{F}\,\alpha$$

- And we are gradually **increasing values of** k then, instead of:

$$[\![\mathcal{M},\varphi]\!]_k := \underbrace{\mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(s^i, s^{i+1})}_{[\![\mathcal{M}]\!]_k} \wedge \underbrace{\bigvee_{j=0}^{k} \alpha^j}_{[\![\varphi]\!]_k^0 = {}_L[\![\varphi]\!]_k^0}$$

- We can write:

$$[\![\mathcal{M},\varphi]\!]_k := \mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \left(\mathcal{R}(s^i, s^{i+1}) \wedge \neg\alpha^i\right) \wedge \alpha^k$$

Constant

- If:

$$\varphi := \mathbf{G}\,\alpha$$

- We will have:

$$[\![\mathcal{M},\varphi]\!]_k := \underbrace{\mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(s^i, s^{i+1})}_{[\![\mathcal{M}]\!]_k} \wedge \underbrace{\bigvee_{L=0}^{k} \mathcal{R}(s^k, s^L) \wedge \underbrace{\bigwedge_{j=0}^{k} \alpha^j}_{{}_L[\![\varphi]\!]_k^0}}_{[\![\varphi]\!]_k}$$

- There is no witness of the formula without a *loopback*.

Fairness

- If:

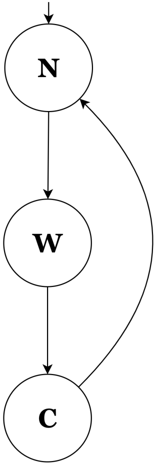
$$\varphi := \mathbf{GF}\,\alpha$$

- We will have:

$$[\![\mathcal{M},\varphi]\!]_k := \underbrace{\mathcal{I}(s^0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(s^i, s^{i+1})}_{[\![\mathcal{M}]\!]_k} \wedge \underbrace{\bigvee_{L=0}^{k} \left(\mathcal{R}(s^k, s^L) \wedge \bigvee_{j=L}^{k} \alpha^j \right)}_{[\![\varphi]\!]_k}$$

Mutex Example

Variables



$$\varphi := \mathbf{G}\,\mathbf{F}\,\mathrm{crit}$$

$$\neg\varphi := \mathbf{F}\,\mathbf{G}\,\neg\mathrm{crit}$$

$$\mathcal{S} = \{N, W, C\}$$

$$PL = \left\{ s_s^i | i \in [0, k], s \in \mathcal{S} \right\}$$

$$[\![\mathcal{M}, \neg\varphi]\!]_k := [\![\mathcal{M}]\!]_k \wedge [\![\neg\varphi]\!]_k$$

SAT encoding of $\mathcal{M}$

Initial Condition:

$$\mathcal{I} := s_N^0 \wedge \bigwedge_{j}^{\mathcal{S} \setminus N} \neg s_j^0 \tag{6}$$

Model Encoding:

$$[\![\mathcal{M}]\!]_k := \mathcal{I} \wedge \mathcal{R} \tag{7}$$

Transition Relation:

$$\mathcal{R} := \bigwedge_{i=0}^{k} (\neg s_N^i \vee s_W^{i+1}) \wedge (\neg s_W^i \vee s_C^{i+1}) \wedge (\neg s_C^i \vee s_N^{i+1}) \tag{8}$$

SAT encoding of $\neg\varphi$

$$[\![\mathbf{F}\,\mathbf{G}\,\neg\mathrm{crit}]\!]_k := \psi_1 \vee \psi_2$$

$$\psi_1 := \overbrace{\bigwedge_{L=0}^{k} \neg s_C^k \vee \neg s_W^L}^{\text{no loopback}} \wedge \overbrace{\bigwedge_{i=0}^{k} \bigvee_{j=i}^{k} \bot}^{[\![\neg\varphi]\!]_k^0}$$

$$\psi_2 := \bigvee_{L=0}^{k} \left[\overbrace{s_C^k \wedge s_N^L}^{\text{loopback}} \wedge \overbrace{\bigvee_{i=0}^{k} \bigwedge_{j=min(i,L)}^{k} \neg s_C^j}^{{}_L[\![\neg\varphi]\!]_k^0} \right]$$

nuXmv

nuXmv Language Enhancements

Backwards-compatible input language, but no processes

Functions, constant array literals, complex array access (**READ**, **WRITE**)

New unbounded data types **Integer** and **Real**, so *infinite-state systems*!

$\Longrightarrow$ Satisfiability Modulo Theories (SMT) instead of Boolean SAT

Timed Transition Systems (TTS), clock variables, LTL time extensions

Support for AIGER And-Inverter Graph input language used for hardware description

New finite-state MC algorithms for Invariant Checking

MiniSat resolution proofs

Interpolation-based algorithms

k-induction algorithms

IC3 algorithms (incremental inductive strengthening, abstraction refinement)

Guided Reachability using PSL SERE regular expressions

Infinite-state Model Checking using SMT

Use SMT instead of SAT solver to implement BMC, k-induction, interpolation, IC3

CEGAR: Counterexample-guided abstraction refinement

$\implies$ Solution to ALLSAT problem, using BDDs + SMT

Novel abstraction refinement algorithms for BMC and k-induction without quantifier elimination

Miscellaneous Improvements

NuSMV: exporting FSM in DOT format

nuXmv: exporting FSM in XMI format, can be read by UML viewer

Range Recoding as state-space reduction technique: only encode used bits of bit vectors

Fin.