A Machine-Checked Proof of a State-Space Construction Algorithm

Nestor Catano\textsuperscript{1}  
Radu I. Siminiceanu\textsuperscript{2}

\textsuperscript{1}University of Madeira, CMU-Portugal
\textsuperscript{2}National Institute of Aerospace, Hampton, Virginia, US

NFM 2010
Outline

1. Introduction
   - Saturation Algorithm
   - Saturation Properties

2. Proof of Correctness
   - Correctness Theorem
   - PVS Formalisation

3. Conclusion and Future Work
What is Saturation?

A symbolic algorithm for building state spaces of discrete systems

At-a-glance

Not your “typical” reachability algorithm
- not focused on states, but on (decision diagram) nodes
- “chaotic” fixed points
- other “non-standard” data structures and conventions
  - QOMDDs, Kronecker matrices
  - storage: index-based nodes, partitioned caches/unique tables, ...
  - first implemented in the SMART tool ⇒ Petri nets

- much faster and leaner than Breadth-First-Search (BFS)
The Saturation Algorithm

Brief History

Evolved from standard BDD algorithm for state-space construction

- **1999**: exploit *event locality*, MDDs
- **2000**: exploit *Kronecker decomposition*, chaining
- **2001**: new iteration strategy (named *Saturation*)
- **2002**: on-the-fly (a.k.a. “unbound”) version
- **2003**: build counterexamples (+ EVMDDs)
- **2004**: other extensions, generalizations
  - lift Kronecker requirements (i.e. back to diagrams for tr. rel.)
  - identity reduced MDDs
  - partial reachability (with EVMDDs)
- **2009**: C library
Saturation: Background and origins

**BFS for Discrete state systems:** $(S, S^0, \mathcal{N})$

- $S^0 \in S$: initial state(s)
- $\mathcal{N} : S \rightarrow 2^S$, the next-state function

**Standard iteration strategy:** $S = S^0 \cup \mathcal{N}(S^0) \cup \mathcal{N}^2(S^0) \cup \ldots$

**Weaknesses:**
- function $\mathcal{N}$ is **monolithic**
- applied to the **entire** current set of states
Kronecker Consistency

**How to improve?**

- **split** the transition function: \( \mathcal{N} = \bigcup_{e \in \mathcal{E}} \mathcal{N}_e \)
- further split events by subsystem (level): \( \mathcal{N}_e = \mathcal{N}_{e,K} \times \ldots \times \mathcal{N}_{e,1} \)
- \( \text{Top}(e) = I \)

**What it does?**

- **Speeds up algorithm**
  - allows one to operate on sub-states
  - fire below, then concatenate with above: perfectly “legal”
- **Makes prover’s life miserable**
Kronecker Consistency

**Origin**: Kronecker product of (transition probability) matrices:

\[
\sum_{e \in E} \bigotimes_{K \geq k \geq 1} \mathcal{N}_{e,k}
\]

**What is it?** independence of local effects of \(\mathcal{N}_{e,k}\) (by level \(k\))

\[
\mathcal{N}_e(s) = \mathcal{N}_{e,K}(s_K) \times \mathcal{N}_{e,K-1}(s_{K-1}) \times \ldots \times \mathcal{N}_{e,1}(s_1)
\]
Saturation: Description

A node $p$ at level $k$ is **saturated** if it encodes a fixed point w.r.t. events $e$ with $\text{Top}(e) \leq k$

$$\text{Below}(k, p) = \mathcal{N}_{\leq k}^*(\text{Below}(k, p))$$

Algorithm in a nutshell:

- build the MDD encoding of $S^0$
- for $k = 1$ upto $K$, **saturate** all nodes at level $k$:
  - $\Rightarrow$ **exhaustively fire** events $e$ with $\text{Top}(e) = k$ (transitive closure)
- if a **firing** creates nodes at levels below $k$:
  - $\Rightarrow$ **saturate** them immediately upon creation
- when the root node is saturated: **voilà**, the full state space

NB: mutual recursion between saturation and firing of events
The two core routines: \texttt{Fire()} and \texttt{Saturate()}

Simplified/sanitized pseudo-code:

\begin{verbatim}
Saturate(k,p)
  do
  foreach e : Top(e) = k do
    Fire(e,k,p)
  while new states discovered
\end{verbatim}

\begin{verbatim}
Fire(e,k,p)
  foreach local transition \( i_k \xrightarrow{e} j_k \)
  \( f = Fire(e, k - 1, \text{child}(k,p,i_k)) \)
  if \( f \neq \emptyset \)
  \( \text{Saturate}(k - 1,f) \)
  \( u = \text{Union}(k-1, f, \text{child}(k,p,j_k)) \)
  set \( \text{child}(k,p,j_k) \leftarrow u \)
\end{verbatim}

NB: Auxiliary structures (caches, unique tables) and other MDD node management routines are not captured here
The Correctness Theorem

Original statement (Simplified Version)

“Let $\langle k|p \rangle$ be a node with saturated children, and $\langle l|q \rangle$ be one of its children with $q \neq 0$ and $l = k - 1$;

- let $\mathcal{U}$ stand for $\text{Below}(l, q)$ before the call to $\text{Fire}(e, l, q)$, for some event $e$ with $l < \text{Top}(e)$

- let $\mathcal{V}$ represent $\text{Below}(l, f)$, where $f = \text{Fire}(e, l, q)$;

$\Rightarrow$ Then, $\mathcal{V} = \mathcal{N}^*_\mathcal{U}(\mathcal{N}_e(\mathcal{U}))$.”
PVS Formalisation

PVS

- PVS is a system for specifying and verifying properties of software and hardware systems.
- PVS’ logic is based on simply typed higher-order logic with functions, product types, records, and recursive definitions.
- PVS’ type systems is extended with sub-types.

Abuse(s) of notation

- $N_e(U)$ is an abuse: because $\text{Top}(e) = k > l$
- Semantics of $N$ is actually not next-state
- PVS discovered these “inconsistencies”

Challenge

- PVS does not directly provide support to mutually recursive functions definitions.
General Approach

- Formalize basic concepts related to Kronecker consistency.
  - Events, MDDs, states, local states, next-state functions, local next-state functions, etc.
- Use definitions to formalize `Saturate()` and `Fire()` to reflect in Logic invariant properties of Saturation algorithm
- Conduct proofs following the pencil-and-paper proof
PVS Formalization

Formalizing Basic Concepts

local_value?(m)(n) : bool = (m = 0 \land n = 0) \lor \\
(m > 0 \land n > 0 \land n \leq nk(m))

local_value(m) : type = (local_value?(m))

state(k) : type = \{ s : Seq(k) \mid \forall (m : upto(k)) : \\
(m = 0 \land s'\sq(m) = 0) \lor \\
(m > 0 \land s'\sq(m) > 0 \land s'\sq(m) \leq nk(m)) \} 

Formalizing Basic Concepts

event : type+

Top : [event \rightarrow posnat]
Formalizing Basic Concepts

\[\text{next}(k) : \text{type} = [\text{event} \rightarrow [\text{state}(k) \rightarrow \text{setof}[\text{state}(k)]]] \]

\[\text{Localnext}(k) : \text{type} = [\text{event} \rightarrow [\text{local}_\text{value}(k) \rightarrow \text{setof}[\text{local}_\text{value}(k)]]] \]

\[\text{Kronecker?}(k)(N)(fs) : \text{bool} = \]
\[\forall (e : \text{event}, x, y : \text{state}(k)) : \]
\[N(e)(x)(y) \iff \forall (m : \text{upto}(k)) : \text{fs}'\text{sq}(m)(e)(x'\text{sq}(m))(y'\text{sq}(m)) \]
Discussion

- **Definitions do not introduce inconsistencies.**
- The use of definitions may force PVS to generate additional proof obligations all over the theorems and lemmas using the definitions, cluttering the proofs.
- **Definitions** may require to reflect the implementation of the algorithm in Logic.
Axiomatic vs Definition Approach

**Discussion**

- **Definitions** do not introduce inconsistencies.
- The use of **definitions** may force PVS to generate additional proof obligations all over the theorems and lemmas using the definitions, cluttering the proofs.
- **Definitions** may require to reflect the implementation of the algorithm in Logic.
Axiomatic vs Definition Approach

Discussion

- **Axioms** might introduce inconsistencies.
- Axioms are more suitable than **definitions** when one is not interested in generating code.
Axiomatic vs Definition Approach

Discussion

- **Axioms** might introduce inconsistencies.
- **Axioms** are more suitable than **definitions** when one is not interested in generating code.
Axiomatic vs Definition Approach

Lessons learned from this exercise

- Non-axiomatic approach was attempted first
- Too many TCCs, mostly from subtyping conditions
- Forced to change definitions, add more (type-dependent) parameters, etc.
- This introduces more TCCs ...
- Proof becomes unmanageable
Formalizing \textit{Saturate()}

Is node $\langle k|p \rangle$ saturated?

$$Below(k, p) = \mathcal{N}_{\leq k}^*(Below(k, p))$$

Is node $\langle k|p \rangle$ saturated?

\[
\begin{align*}
\text{saturated?}(k: \text{upto}(L), p: \text{OMDD}) & \quad (N: \text{next}(k), \text{fs}: (\text{kronecker?}(k)(N)))(w: \text{nat}): \text{bool} = \\
\text{Below}(k, p) = \text{Apply}(k)(N, \text{fs})(w)(\text{Below}(k, p))
\end{align*}
\]

Reaching the Fixed-Point

- $w$ represents the number of iterations after which $\mathcal{N}_{\leq k}^*(\text{Below}(k, p))$ does not generate any new state.

- The existence of $w$ is guaranteed by the finiteness of $\text{Below}(k, p)$ and because the firing of any $\mathcal{N}_e \in \mathcal{N}_{\leq k}$ is an increasing function (the set of reached states gets larger)
Formalizing \textit{Fire()}

Recursion’s Base Case

\[
\text{fire}(l, e, N, fs, w, q) : \text{OMDD}
\]

\[
\text{fire_trivial} : \text{axiom}
\quad l < \text{Bottom}(e) \lor l = 0 \Rightarrow \text{fire}(l, e, N, fs, w, q) = q
\]
Formalizing \textit{Fire()}

**Recursive Case**

\textit{Fire()} is recursively called on their children \texttt{child}(q,i)

**Recursive Case**

\begin{verbatim}
fire_recursive: axiom
Below(l, fire(l, e, N, fs, w, q)) =
{ s : state(l) | ∃(i : local_value(l)) : fs'sq(l)(e)(i)(s'sq(l)) ∧
    Below(l-1, fire(l-1, e, N₁', fs₁', w, child(q, i)))(sₜ(l, s, l-1)) }
\end{verbatim}
Formalizing *Fire()*

**Mutual Recursion Fire() vs Saturate()**

*Fire()* is always invoked on a saturated node \(\langle l|q \rangle\) with \(l < Top(e)\) and *Saturate()* is invoked just before returning from *Fire()*.

**Mutual Recursion Fire() vs Saturate()**

\[
\text{fire\_saturated: axiom}
\]
\[
saturated?(l, fire(l, e, N, fs, w, q))(N, fs)(w)
\]
The PVS Correctness Proof

Auxiliary Result

\[ N^*_{\leq k-1}(N_e(B(k, p))) = \bigcup_{i \in S^k} N^k_e(i) \times N^*_{\leq k-1}(N_e(B(\langle k - 1|\langle k|p\rangle[i]\rangle))) \]

Auxiliary Result

**kronecker_apply: theorem**

\[
\text{Apply}(k,k-1)(N,fs)(w)(\text{Next}(k,ev)(N,fs)(\text{Below}(k,p)))(s) \\
\iff \\
(\exists (i:\text{local\_value}(k)):\ \\
fs'\text{sq}(k)(ev)(i)(s'sq(k)) \land \\
\text{Apply}(k-1)(N'_k,fs'_k)(w_1)(\text{Next}(k - 1, ev)(N'_k, fs'_k)(
\text{Below}(k-1, \text{child}(p, i)))))(s_t(k, s, k-1))
\]

General Idea

- Induction on \( w \)
- Kronecker Consistency
The PVS Correctness Proof

- let $U$ stand for $\text{Below}(l, q)$ before the call to $\text{Fire}(e, l, q)$, for some event $e$ with $l < \text{Top}(e)$
- let $V$ represent $\text{Below}(l, f)$, where $f = \text{Fire}(e, l, q)$;

  $\Rightarrow$ Then, $V = \mathcal{N}_{\leq l}^*(\mathcal{N}_e(U))$.

The PVS Correctness Proof

```
saturation_correctness: theorem
Below(l, fire(l, e, N, fs, w, q)) =
  Apply(l)(N, fs)(w)(Next(l, e)(N, fs)(Below(l, q)))
```

General Idea

- Kronecker consistency
- Finite domains, increasing functions
- Induction on $l$
Proof Statistics

- 10 Theories
- 145 Proofs
- 10 Lemmas
- 2 main Theorems

Code Generation

- Porting PVS theories to B Machines.
- Using refinement tools (e.g., AtelierB) to generate certified C code.
- Benchmarking (parts of) generated code with existing implementation in SMART.
Questions?