Refinement for Symbolic Trajectory Evaluation

Authors

Chalmers

Abstract. Model refinement such that it preserves symbolic trajectory evalutions.

Keywords: STE · Refinement · ?

1 Introduction to STE

1.1 Original STE

Symbolic trajectory evaluation [5] (STE) is a high-performance model checking technique based on *symbolic simulation* extended with a temporal *next-time* operator to describe circuit behaviour over time. In its simplest form, STE tests the validity of an *assertion* of the form $A \Rightarrow C$, where both the *antecedent* A and *consequent* C are formulas in the following logic:

$$f ::= p \mid f \wedge f \mid P \rightarrow f \mid \mathbf{N}\, f$$

Here, p is a simple predicate over "values" in a circuit and P is a Boolean propositional formula, and the operators $\wedge$, $\rightarrow$ and $\mathbf{N}$ are conjunction, domain restriction and the next-time operator, respectively.

If the circuit contains Boolean signals, p is typically drawn from the following two predicates: $n\,\mathbf{is}\,1$ and $n\,\mathbf{is}\,0$, where n ranges over the signals (or nodes) in a circuit. For example, suppose we have a unit-delayed, two-input AND-gate, then it is reasonable to assume that the assertion $(in_1\,\mathbf{is}\,1 \wedge in_2\,\mathbf{is}\,1) \Rightarrow \mathbf{N}(out\,\mathbf{is}\,1)$ is true. Indeed, STE efficiently validates such statements for us.

While the truth semantics of an assertion in STE is defined as the satisfaction of its "defining" trajectory (bounded sequence of states) relative to a model structure of the circuit, what the STE algorithm computes is exactly the solution of a data-flow equation [1] in the classic format [4]. ...

1.2 Lattice-theoretic STE

Consider an arbitrary, but fixed, digital circuit M operating in discrete time. A *configuration* of M, denoted by C, is non-empty and finite set that represents a snapshot of M at a discrete point in time. If the circuit M has m boolean signals, then its set of configurations is typically represented as a sequence $\mathbb{B}^m$, where $\mathbb{B} = \{0, 1\}$ is the set of boolean values.

Circuit Model A simple conceptual model of M is a *transition relation*, $M_R \subseteq C \times C$, where $(c, c') \in M_R$ means that M can move from c to c' in one step[1]. The power set of C, denoted by $\mathcal{P}(C)$, can be viewed as a the set of *predicates* on configurations, where $\cap$, $\cup$, and $\subseteq$ correspond to conjunction, disjunction and implication, respectively. We denote by $\cap Q$ and $\cup Q$ the intersection and union of all members of any $Q \subseteq \mathcal{P}(C)$.

M_R induces a *predicate transformer* $M_F \in \mathcal{P}(C) \to \mathcal{P}(C)$ using the relational image operation:

$$M_F(p) = \{c' \in C \mid \exists c \in p : (c, c') \in M_R\}$$

It is intuitively obvious that if M is in one of the configurations in $p \in \mathcal{P}(C)$, then in one time step it must be in one of the configurations in $M_F(p)$. We also see that M_F distributes over arbitrary unions:

$$M_F(\cup Q) = \cup\{M_F(q) \mid q \in Q\}$$

for all $Q \subseteq \mathcal{P}(C)$. Any M_F that satisfies this distributive property also defines a M_R through the equivalence $(c, c') \in M_R \Leftrightarrow c' \in M_F(\{c\})$, that is to say, there is no loss of information going from M_R to M_F or vice versa. It follows that M_F preserves the empty set of constraints, i.e. $M_F(\emptyset) = \emptyset$, and is monotonic, i.e. $p \subseteq q \Rightarrow M_F(p) \subseteq M_F(q)$ for all $p, q \in \mathcal{P}(C)$. We adopt this functional model of M and drop its subscript.

In practice, the predicates of C correspond to signals in M and divided into external, or "input" and "output", and internal signals. While an input signal is typically controlled by the external environment, and thus unconstrained by M itself, non-input signals are determined by the circuit topology and functionality. For example, supposed M is the earlier example of a unit-delayed two-input AND gate, we could then define its model $M \in \mathcal{P}(\mathbb{B}^3) \to \mathcal{P}(\mathbb{B}^3)$ as:

$$M(p) = \{\langle b_1, b_2, i_1 \wedge i_2\rangle \in \mathbb{B}^3 \mid \langle i_1, i_2, o\rangle \in p\}$$

Here i_1 and i_2 refer to the two inputs of the AND gate and o the ignored output; b_1 and b_2 are unconstrained inputs in the new configuration.

Ternary lattices Manipulating subsets of $\mathbb{B}^m$ is however impractical for even moderately large m, which leads us to one of the key insights of STE. Namely, instead of manipulating subsets of $\mathbb{B}^m$ directly, one can use sequences of ternary values $\mathbb{T} = \mathbb{B} \cup \{\mathrm{X}\}$ to approximate them, whose sizes are only linear in m. Here the 1 and 0 from $\mathbb{B}$ denotes specific, defined values whereas X denotes an "unknown" value that could be either 1 or 0. This intuition induces a partial order $\sqsubseteq$ on $\mathbb{T}$, where $0 \sqsubseteq \mathrm{X}$ and $1 \sqsubseteq \mathrm{X}$[2]. For any $m \in \mathbb{N}$, this ordering on $\mathbb{T}$ is lifted component-wise to $\mathbb{T}^m$.

[1] Mention how this affects circuits with zero-delays?

[2] We use the reverse ordering of what is originally used in STE.

Note that $\mathbb{T}^m$ does not quite form a complete lattice because it lacks a bottom: both $0 \sqsubseteq \mathrm{X}$ and $1 \sqsubseteq \mathrm{X}$ but 0 and 1 are equally defined. A special bottom element $\bot$ is therefore introduced, such that $\bot \sqsubseteq t$ and $\bot \neq t$ for all $t \in \mathbb{T}^m$. The extended $\mathbb{T}^m_\bot = \mathbb{T}^m \cup \{\bot\}$ then becomes a complete lattice. We denote the top element $\langle \mathrm{X}, \ldots, \mathrm{X} \rangle$ of $\mathbb{T}^m_\bot$ by $\top$.

Generalising from any specific domain, let $(\hat{P}, \sqsubseteq)$ be a finite, complete lattice of *abstract predicates* in which the meet $\sqcap$ and join $\sqcup$ of any subset $Q \subseteq \hat{P}$ exists. Similar to the previous set operations for power sets, $\sqcap$, $\sqcup$ and $\sqsubseteq$ correspond to conjunction, disjunction and implication for abstract predicates, respectively. Furthermore, for any $Q \subseteq \hat{P}$, we denote by $\sqcap Q$ and $\sqcup Q$ the meet and join of all members of Q.

Abstract circuit model Let there be a Galois connection relating "concrete" predicates $\mathcal{P}(C)$ and abstract predicates $\hat{P}$. The usual definition of a Galois connection is in terms of an *abstraction* $\alpha \in \mathcal{P}(C) \to \hat{P}$ and a *concretisation* $\gamma \in \hat{P} \to \mathcal{P}(C)$ function, such that $\alpha(p) \sqsubseteq \hat{p} \Leftrightarrow p \subseteq \gamma(\hat{p})$ for all $p \in \mathcal{P}(C)$ and $\hat{p} \in \hat{P}$. For example, a Galois connection from $\mathcal{P}(\mathbb{B}^m)$ to $\mathbb{T}^m_\bot$ for any $m \in \mathbb{N}$ can be defined in a natural way by its concretisation function $\gamma \in \mathbb{T}^m_\bot \to \mathcal{P}(\mathbb{B}^m)$:

$$\gamma(\langle t_0, \ldots, t_{m-1} \rangle) = \{\langle b_0, \ldots, b_{m-1} \rangle \in \mathbb{B}^m \mid \forall i < m : t_i \neq \mathrm{X} \Rightarrow b_i = t_i\}$$
$$\gamma(\bot) = \emptyset$$

Listing each concrete predicate approximated by a given abstract predicate. Its abstraction function $\alpha \in \mathcal{P}(\mathbb{B}^m) \to \mathbb{T}^m_\bot$ instead finds the most precise abstract predicate for a set of concrete predicates:

$$\alpha(p) = \sqcup\{\langle t_0, \ldots, t_{m-1} \rangle \in \mathbb{T}^m_\bot \mid \langle b_0, \ldots, b_{m-1} \rangle \in p, \forall i < m : b_i = t_i\}$$
$$\alpha(\emptyset) = \bot$$

We adopt a different interpretation of Galois connections that is given by means of a binary relation between $\mathcal{P}(C)$ and $\hat{P}$. Specifically, let $\ll \subseteq \mathcal{P}(C) \times \hat{P}$ be a binary relation, where $p \ll \hat{p}$ reads as "p can be approximated as $\hat{p}$", such that for all $Q \subseteq \mathcal{P}(C)$ and $\hat{Q} \subseteq \hat{P}$:

$$\forall p \in Q : \forall \hat{p} \in \hat{Q} : p \ll \hat{p} \Leftrightarrow \cup Q \ll \sqcap \hat{Q}$$

Intuitively, this relation is an extension of the partial order $\subseteq$ of $\mathcal{P}(C)$ and $\sqsubseteq$ of $\hat{P}$ to an ordering between $\mathcal{P}(C)$ and $\hat{P}$. The original abstraction and concretisation functions can be derived from the relation by: $\alpha(p) = \sqcap\{\hat{p} \in \hat{P} \mid p \ll \hat{p}\}$ and $\gamma(\hat{p}) = \cup\{p \in \mathcal{P}(C) \mid p \ll \hat{p}\}$. Conversely, the relation can be derived from α and γ as: $p \ll \hat{p} \Leftrightarrow \alpha(p) \sqsubseteq \hat{p}$ and $p \ll \hat{p} \Leftrightarrow p \subseteq \gamma(\hat{p})$.

An *abstract predicate transformer* $\hat{M} \in \hat{P} \to \hat{P}$ is an *abstract interpretation* [2] of $M \in \mathcal{P}(C) \to \mathcal{P}(C)$ iff: (1) $\hat{M}$ preserves $\bot$, i.e. $\hat{M}(\bot) = \bot$; (2) $\hat{M}$ is monotonic, i.e. $\hat{p} \sqsubseteq \hat{q} \Rightarrow \hat{M}(\hat{p}) \sqsubseteq \hat{M}(\hat{q})$ for all $\hat{p}, \hat{q} \in \hat{P}$; and (3) $\ll$ is a *simulation*

relation from $\mathcal{P}(C)$ to $\hat{P}$, i.e. $p \ll \hat{p} \Rightarrow M(p) \ll \hat{M}(\hat{p})$ for all $p \in \mathcal{P}(C)$ and $\hat{p} \in \hat{P}$. That $\ll$ is a simulation relation can also be stated in terms of its abstraction α and concretisation γ functions: $\alpha(M(p)) \sqsubseteq \hat{M}(\alpha(p))$ for all $p \in \mathcal{P}(C)$, and $M(\gamma(\hat{p})) \subseteq \gamma(\hat{M}(\hat{p}))$ for all $\hat{p} \in \hat{P}$.

Note that $\hat{M}$ does not distribute over arbitrary join in general because information is potentially discarded when joining two lattices. As an example, let the following $\hat{M}$ abstract the earlier unit-delayed AND gate:

$$\begin{array}{ll} \hat{M}(\langle 1,1,p_2\rangle) = \langle \mathrm{X},\mathrm{X},1\rangle & \hat{M}(\langle 0,0,p_2\rangle) = \langle \mathrm{X},\mathrm{X},0\rangle \\ \hat{M}(\langle 0,\mathrm{X},p_2\rangle) = \langle \mathrm{X},\mathrm{X},0\rangle & \hat{M}(\langle \mathrm{X},0,p_2\rangle) = \langle \mathrm{X},\mathrm{X},\mathrm{X}\rangle \\ \hat{M}(\langle p_0,p_1,p_2\rangle) = \langle \mathrm{X},\mathrm{X},\mathrm{X}\rangle & \end{array}$$

Where the last, most general matching is overlapped by the concrete ones. If we apply $\hat{M}$ to the join of $\langle 0,1,\mathrm{X}\rangle$ and $\langle 1,0,\mathrm{X}\rangle$, or if we apply $\hat{M}$ to them individually and then join, we get two different results:

$$\begin{array}{l} \hat{M}(\langle 0,1,\mathrm{X}\rangle \sqcup \langle 1,0,\mathrm{X}\rangle) = \hat{M}(\langle \mathrm{X},\mathrm{X},\mathrm{X}\rangle) = \langle \mathrm{X},\mathrm{X},\mathrm{X}\rangle \\ \hat{M}(\langle 0,1,\mathrm{X}\rangle) \sqcup \hat{M}(\langle 1,0,\mathrm{X}\rangle) = \langle \mathrm{X},\mathrm{X},0\rangle \sqcup \langle \mathrm{X},\mathrm{X},0\rangle = \langle \mathrm{X},\mathrm{X},0\rangle \end{array}$$

The inequality $\sqcup\{\hat{M}(\hat{q}) \mid \hat{q} \in \hat{Q}\} \sqsubseteq \hat{M}(\sqcup\hat{Q})$ for all $\hat{Q} \sqsubseteq \hat{P}$ does however hold, since it is implied by the monotonicity of $\hat{M}$.

Assertions and satisfaction A *trajectory assertion* for an abstact model $\hat{M}$ is a quintuple $\hat{A} = (S, s_0, R, \pi_a, \pi_c)$, where S is a finite set of *states*, $s_0 \in S$ is an *initial state*, $R \subseteq S \times S$ is a *transition relation*, $\pi_a \in S \to \hat{P}$ and $\pi_c \in S \to \hat{P}$ label each state s with an *antecedent* $\pi_a(s)$ and a *consequent* $\pi_c(s)$. Furthermore, we assume that $(s, s_0) \notin S$ for all $s \in S$ without any loss of generality.

For all functions $\Phi \in S \to \hat{P}$ and states $s \in S$, define $F \in S \to (\hat{P} \to \hat{P})$ and $\mathcal{F} \in (S \to \hat{P}) \to (S \to \hat{P})$ as follows:

$$F(s)(\hat{p}) = \hat{M}(\pi_a(s) \sqcap \hat{p}) \tag{1}$$

$$\mathcal{F}(\Phi)(s) = \textbf{if } (s = s_0) \textbf{ then } \top \textbf{ else } \sqcup\{F(s')(\Phi(s')) \mid (s', s) \in R\} \tag{2}$$

F preserves $\bot$, both F and $\mathcal{F}$ are monotonic, and two $\Phi, \Phi' \in S \to \hat{P}$ are ordered as $\Phi \sqsubseteq \Phi' \Leftrightarrow \forall s \in S : \Phi(s) \sqsubseteq \Phi'(s)$. Let $\Phi_* \in S \to \hat{P}$ be the least fixpoint of the equation $\Phi = \mathcal{F}(\Phi)$ [3]. Since both S and $\hat{P}$ are finite, Φ_* is given by $\lim \Phi_n(s)$ where Φ_n is defined as follows:

$$\Phi_n = \textbf{if } (n = 0) \textbf{ then } (\lambda s \in S : \bot) \textbf{ else } \mathcal{F}(\Phi_{n-1}) \tag{3}$$

$\hat{M}$ *satisfies* a trajectory assertion $\hat{A}$, denoted by $\hat{M} \models \hat{A}$, iff:

$$\forall s \in S : \Phi_*(s) \sqcap \pi_\alpha(s) \sqsubseteq \pi_c(s) \tag{4}$$

$\hat{M} \models \hat{A}$ implies that a concretisation of $\hat{A}$ can also be satisfied by the original, set-based model M [1].

2 System refinement

Consider another fixed, but arbitrary, circuit model $N \in \mathcal{P}(D) \to \mathcal{P}(D)$ where its set configurations D is non-empty and finite. Exactly what configurations such as D are, were not important previously. To reason about refinement of models, however, we make a distinction between their external and internal elements. The rationale is that refinement relate visible behaviours of circuit models, it should not matter whether their internal elements can be aligned or not.

Let the visible components visible elements of a configurations C be identified by two projection mappings, $\mathrm{out} \in \mathcal{P}(C) \to \mathcal{P}(C)$ and $\mathrm{in} \in \mathcal{P}(C) \to \mathcal{P}(C)$, identifying its "inputs" and "outputs", respectively. The set of all possible inputs is the image $\mathrm{in}[\mathcal{P}(C)] = \{\mathrm{in}(p) \mid p \in \mathcal{P}(C)\}$, which we denote by I. Note that, since M cannot control its input signals, the transitions of M in response to some input $i \in I$ are given by $M(p \cap i)$, for any initial state $p \in \mathcal{P}(C)$.

A *trajectory* of M is a non-empty sequence of configurations, $\tau \in C^+$, induced by its response to a *driver* $\delta \in I^+$. Specifically, $\tau_0 = C$ and $n \in \mathbb{N} : n < |\delta| \Rightarrow \tau_{n+1} = M(\tau_n \cap \delta_n)$; the trajectory of M induced by a δ is denoted by M_δ and is prefix-closed. As C is finite, the set of all possible trajectories of M fully capture its behaviour for starting in the arbitrary initial state.

A Appendices

Text.

References

1. Chou, C.T.: The mathematical foundation of symbolic trajectory evaluation. In: International Conference on Computer Aided Verification. pp. 196–207. Springer (1999)
2. Cousot, P.: Abstract interpretation. ACM Computing Surveys (CSUR) **28**(2), 324–328 (1996)
3. Davey, B.A., Priestley, H.A.: Introduction to lattices and order. Cambridge university press (2002)
4. Muchnick, S., et al.: Advanced compiler design implementation. Morgan kaufmann (1997)
5. Seger, C.J.H., Bryant, R.E.: Formal verification by symbolic evaluation of partially-ordered trajectories. Formal Methods in System Design **6**(2), 147–189 (1995)