The Mathematical Foundation of Symbolic Trajectory Evaluation

Ching-Tsun Chou
⟨ctchou@mipos2.intel.com⟩

Intel Corporation
3600 Juliette Lane, SC12-401
Santa Clara, CA 95052, U.S.A.

Abstract. In this paper we elucidate the mathematical foundation underlying both the *basic* and the *extended* forms of *symbolic trajectory evaluation* (STE), with emphasis on the latter. In addition, we make three contributions to the theory of STE which, we believe, are new. First, we provide a satisfactory answer to the question: what does it mean for a circuit to satisfy a trajectory assertion? Second, we make the observation that STE is a form of *data flow analysis* and, as a corollary, propose a conceptually simple algorithm for (extended) STE. Third, we show that the ternary model of circuits used by STE is an *abstract interpretation* of the ordinary boolean model via a *Galois connection*. We hope that our exposition will make STE, especially its extended form, less mysterious.

1 Introduction

In BDD-based formal verification, *symbolic trajectory evaluation* (STE) [10, 6] is the main alternative to *symbolic model checking* (SMC) [3]. Compared with SMC, STE has the advantage that it can be applied to very large circuits directly, without the need to abstract the circuits before verification. This is made possible by a pleasant property of STE: the number of BDD variables needed in an STE run depends only on the assertion being checked, *not* on the circuit under analysis. Thus one can use STE to verify a collection of assertions against the same circuit without having to invent a different abstraction of the circuit for each assertion, as one often has to do when doing SMC. On the other hand, what STE can verify is more restricted than what SMC can. In its basic form [10], STE can only verify assertions over bounded intervals of time, possibly iterated by non-nested loops. But in its extended form [6][1], STE can verify assertions expressed as arbitrary state-transition graphs, thus enabling STE to verify any safety properties. As far as we know, STE has not been generalized to reason about liveness properties.

Unfortunately, STE seems to be much less well-known than SMC, certainly less than it deserves to be. In the hope of generating more interests in STE, we elucidate in this paper the mathematical foundation underlying both the basic [10] and the extended [6] forms of STE, with emphasis on the latter. We hope that our exposition will fill enough

[1] According to Carl Seger, the basic ideas of *extended* (a.k.a. *generalized*) STE came out of an e-mail brainstorming session in 1994 among Derek Beatty, Randy Bryant, and Carl Seger on an (unpublished) note written by Beatty. The author learned of the basic ideas of extended STE from Alok Jain's Ph.D. thesis, which was supervised by Randy Bryant [6].

gaps and clarify enough obscurities in the existing literature to make STE, especially its extended form, less mysterious. In addition to this, we make three contributions to the theory of STE which, we believe, are new.

First, we would like to clarify the semantics of STE by providing a satisfactory answer to the following question:

> What does it mean for a circuit to satisfy a trajectory assertion?

More precisely, we propose to define the satisfaction relation for extended STE [6], in which trajectory assertions can have arbitrary state-transition graphs, as a universally quantified generalization of the form of basic STE [10] in which trajectory assertions are bounded sequences of states. Interestingly, this is not how the satisfaction relation for extended STE was originally defined in [6], which uses a partially existentially quantified generalization. To justify our definition, we show that it guarantees that a circuit satisfies a trajectory assertion iff (if and only if) the set-theoretic STE algorithm returns a positive answer, and that this is not the case for the definition in [6]. Another advantage of our definition is that it does not require us to make the distinction of whether a trajectory assertion is "oblivious" (which basically means "deterministic"), whereas the definition in [6] does.

Second, we would like to make the following observation:

> STE is a form of *data flow analysis* (DFA).

More precisely, we show that, when properly formulated, what an STE algorithm computes is exactly the solution of a data flow equation in the classic format [8]. Though perhaps obvious in retrospect, this point seems to have never been noticed before. Furthermore, as a corollary of this DFA formulation, we propose a BDD-based, completely implicit algorithm for STE that is very easy to understand and, we hope, can lead to simple implementations of STE. (Of course, this hope can be confirmed or disproved only through experimentation, which is beyond the scope of this paper.)

Third, we would like to propose an appropriate framework in which to address the following question:

> How is the ternary model of circuits that STE algorithms use related to the ordinary boolean model of circuits?

Specifically, we show that the ternary model is an *abstract interpretation* in the classic sense [9] of the boolean model via a *Galois connection* [4, 9]. We also point out a relationship between the two models (namely, the Galois connection should be a *simulation* from the boolean model to the ternary model) that seems to have never been articulated in the existing literature on STE [10, 6].

The rest of this paper is organized as follows. Section 2 presents STE from a "set-theoretic" viewpoint, in which circuits are modeled as functions operating on sets of boolean vectors. Section 3 presents STE from a "lattice-theoretic" viewpoint, in which circuits are modeled as functions operating on ternary vectors, which form a lattice. Section 4 concludes this paper by discussing some directions for future research. Due to space limitations, the proofs of all theorems and the reviews of some mathematical machineries are relegated to the appendices.

2 Set-Theoretic STE

Following [6], we start with set-theoretic STE, which manipulates sets of configurations of circuits. As will be seen, set-theoretic STE is impractical except on small circuits. But it provides an easy-to-understand semantic foundation by which STE can be related to symbolic model checking, which takes a set-theoretic view of circuits. Furthermore, the development of lattice-theoretic STE in the next section closely parallels that of set-theoretic STE.

2.1 Set-Theoretic Models of Circuits

Circuits as Relations. Consider a digital circuit M operating in discrete time. A *configuration* of M is an assignment of "values" to "signals" in M, representing a snapshot of M at a discrete point in time. In this section, exactly what "values" and "signals" are, is actually not important. We will only assume that the set of all configurations of M, denoted by C, is nonempty and *finite*.

The conceptually simplest model of M is a *transition relation*, $M_{\text{Rel}} \subseteq C \times C$, where $(c, c') \in M_{\text{Rel}}$ means that M can in one step move from configuration c to configuration c'. Note that since M cannot control its input signals, M_{Rel} is in general a relation rather than a function.

Circuits as Functions. The power set of C, denoted by $\mathcal{P}(C)$, can be viewed as the set of *predicates* on configurations, where $\cap$, $\cup$, and $\subseteq$ correspond to conjunction, disjunction, and implication, respectively. For any $Q \subseteq \mathcal{P}(C)$, we denote by $\bigcap Q$ and $\bigcup Q$ the intersection and union of all members of Q, respectively.

Using the relational image operation, the transition relation M_{Rel} induces a *predicate transformer* $M_{\text{Fun}} \in \mathcal{P}(C) \to \mathcal{P}(C)$ in a natural way:

$$M_{\text{Fun}}(p) \;=\; \{c' \in C \mid \exists\, c \in p : (c, c') \in M_{\text{Rel}}\} \tag{1}$$

for all $p \in \mathcal{P}(C)$. Intuitively, if M is in one of the configurations in p, then in one step it must be in one of the configurations in $M_{\text{Fun}}(p)$. It is easy to show from (1) that M_{Fun} distributes over arbitrary union:

$$M_{\text{Fun}}\Big(\bigcup Q\Big) \;=\; \bigcup\{M_{\text{Fun}}(q) \mid q \in Q\} \tag{2}$$

for all $Q \subseteq \mathcal{P}(C)$. Conversely, for any $M_{\text{Fun}} \in \mathcal{P}(C) \to \mathcal{P}(C)$ that satisfies (2), the equivalence: $(c, c') \in M_{\text{Rel}} \Leftrightarrow c' \in M_{\text{Fun}}(\{c\})$, where $c, c' \in C$, defines a $M_{\text{Rel}} \subseteq C \times C$ that satisfies (1). Thus there is no loss of information in going from M_{Rel} to M_{Fun} and vice versa.

In the remainder of this paper we will use the functional model of circuits exclusively and drop the subscript $_{\text{Fun}}$. Note that it follows from distributivity (2) that M $(= M_{\text{Fun}})$ both preserves $\emptyset$ and is monotonic:

$$M(\emptyset) \;=\; \emptyset \tag{3}$$

$$p \subseteq q \;\Rightarrow\; M(p) \subseteq M(q) \tag{4}$$

for all $p, q \in \mathcal{P}(C)$.

2.2 Set-Theoretic Trajectory Assertions

From now on we focus on a fixed, but arbitrary, circuit $M \in \mathcal{P}(C) \to \mathcal{P}(C)$, where C is nonempty and finite, such that (2) is true.

Definition of a Trajectory Assertion. A *trajectory assertion* for M is a quintuple $A = (S, s_0, R, \pi_a, \pi_c)$, 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*, and $\pi_a \in S \to \mathcal{P}(C)$ and $\pi_c \in S \to \mathcal{P}(C)$ label each state s with an *antecedent* $\pi_a(s)$ and a *consequent* $\pi_c(s)$, such that:

$$\forall\, s \in S : (s, s_0) \notin R \tag{5}$$

Assumption (5) is made for the technical reason that in formulating data flow algorithms, it is convenient to have a unique source node whose flow value never needs changing. No generality is lost by assuming (5).

Satisfaction of a Trajectory Assertion by a Circuit. What does it mean for the circuit M to satisfy the trajectory assertion $A = (S, s_0, R, \pi_a, \pi_c)$? Roughly speaking, it means that for every trajectory τ of M and every run ρ of A, as long as τ satisfies the antecedents in ρ, τ satisfies the consequents in ρ. To state this precisely, we have to introduce some terminologies. (Also, see Appendix A for notations about sequences.)

A *trajectory* of M is a nonempty sequence of configurations, $\tau \in C^+$, such that:

$$\forall\, i \in \mathbf{N} : 0 < i < |\tau| \;\Rightarrow\; \tau[i] \in M(\{\tau[i-1]\})$$

The set of trajectories of M is denoted by $\mathit{Traj}(M)$. A *run* of A is a nonempty sequence of states, $\rho \in S^+$, such that:

$$\rho[0] = s_0 \;\;\wedge\;\; \forall\, i \in \mathbf{N} : 0 < i < |\rho| \;\Rightarrow\; (\rho[i-1], \rho[i]) \in R$$

The set of runs of A is denoted by $\mathit{Runs}(A)$. Note that both $\mathit{Traj}(M)$ and $\mathit{Runs}(A)$ are *prefix-closed*:

$$\forall\, \tau, \tau' \in C^+ : \tau \preceq \tau' \;\Rightarrow\; (\,\tau' \in \mathit{Traj}(M) \;\Rightarrow\; \tau \in \mathit{Traj}(M)\,)$$
$$\forall\, \rho, \rho' \in S^+ : \rho \preceq \rho' \;\Rightarrow\; (\,\rho' \in \mathit{Runs}(A) \;\Rightarrow\; \rho \in \mathit{Runs}(A)\,)$$

For any $\tau \in \mathit{Traj}(M)$ and $\rho \in \mathit{Runs}(A)$ such that $|\tau| = |\rho|$, we say τ *a-satisfies* ρ, denoted by $\tau \models_a \rho$, iff every configuration in τ satisfies the corresponding antecedent in ρ:

$$\forall\, i \in \mathbf{N} : 0 \le i < |\rho| \;\Rightarrow\; \tau[i] \in \pi_a(\rho[i])$$

Similarly, we say τ *c-satisfies* ρ, denoted by $\tau \models_c \rho$, iff every configuration in τ satisfies the corresponding consequent in ρ:

$$\forall\, i \in \mathbf{N} : 0 \le i < |\rho| \;\Rightarrow\; \tau[i] \in \pi_c(\rho[i])$$

Finally, we say the circuit M *satisfies* the trajectory assertion A, denoted by $M \models A$, iff:

$$\forall\, \tau \in \mathit{Traj}(M) : \forall\, \rho \in \mathit{Runs}(A) :\; |\tau| = |\rho| \;\Rightarrow\; (\,\tau \models_a \rho \;\Rightarrow\; \tau \models_c \rho\,) \tag{6}$$

Comparison with Another Definition of Satisfaction. It is instructive to compare our definition of the satisfaction relation, (6), with the definition used in [6]:

$$\begin{array}{l} \forall\, \tau \in \mathit{Traj}(M) : \\ \quad (\,\exists\, \rho \in \mathit{Runs}(A) : |\tau| = |\rho| \;\wedge\; \tau \models_a \rho \;\wedge\; \tau \models_c \rho\,) \;\vee \\ \quad (\,\forall\, \tau' \preceq \tau : \forall\, \rho' \in \mathit{Runs}(A) : |\tau'| = |\rho'| \;\Rightarrow\; (\tau' \models_a \rho' \;\Rightarrow\; \tau' \models_c \rho')\,) \end{array} \tag{7}$$

Note that (6) implies (7), because $\mathit{Traj}(M)$ is prefix-closed. The converse is not true, but if its first disjunct were removed, (7) would indeed be equivalent to (6). That first disjunct, which contains an existential quantifier, makes (7) harder to implement than (6). Intuitively, the existential quantifier requires backtracking to implement. Formally, we will show in the next subsection that (6) holds iff the set-theoretic STE algorithm returns a positive answer, and that (7) lacks this nice property.

To get around this difficulty, [6] introduces the notion of oblivious trajectory assertions. A trajectory assertion $A = (S, s_0, R, \pi_a, \pi_c)$ is *oblivious* iff for any $s, s', s'' \in S$ such that $(s, s') \in R$ and $(s, s'') \in R$, it must be the case that $\pi_a(s') \cap \pi_a(s'') = \emptyset$. Consequently, given any trajectory τ, there is at most one run ρ of A such that τ a-satisfies ρ. It is not hard to see that for an oblivious trajectory assertion, (7) implies (6), which then implies that (7) is equivalent to the set-theoretic STE algorithm returning a positive answer. With our definition (6), there is no need to introduce the notion of obliviousness.

2.3 Set-Theoretic STE as DFA

In this subsection we show that the checking of $M \models A$ can be formulated as a DFA problem [8].

Define $F \in S \to (\mathcal{P}(C) \to \mathcal{P}(C))$ such that $F(s)(p) = M(\pi_a(s) \cap p)$ for all $s \in S$ and $p \in \mathcal{P}(C)$. It follows from (2) that:

$$F(s)(\emptyset) \;=\; \emptyset \tag{8}$$

$$p \subseteq q \;\Rightarrow\; F(s)(p) \subseteq F(s)(q) \tag{9}$$

$$F(s)(\bigcup Q) \;=\; \bigcup\{F(s)(q) \mid q \in Q\} \tag{10}$$

for all $s \in S$, $p, q \in \mathcal{P}(C)$, and $Q \subseteq \mathcal{P}(C)$.

Next, define $\mathcal{F} \in (S \to \mathcal{P}(C)) \to (S \to \mathcal{P}(C))$ such that:

$$\mathcal{F}(\Phi)(s) \;=\; \begin{cases} C & \text{if } s = s_0 \\ \bigcup\{\, F(s')(\Phi(s')) \mid (s', s) \in R \,\} & \text{otherwise} \end{cases}$$

for all $\Phi \in S \to \mathcal{P}(C)$ and $s \in S$. Since $F(s)$ is monotonic for all $s \in S$ (see (9) above), $\mathcal{F}$ is monotonic as well, where the function space $S \to \mathcal{P}(C)$ is ordered as follows: $\Phi \sqsubseteq \Phi' \Leftrightarrow \forall\, s \in S : \Phi(s) \subseteq \Phi'(s)$, for all $\Phi, \Phi' \in S \to \mathcal{P}(C)$. Hence, by Knaster-Tarski Fixpoint Theorem [4], the following fixpoint equation:

$$\Phi \;=\; \mathcal{F}(\Phi) \tag{11}$$

has a least solution $\Phi_* \in S \to \mathcal{P}(C)$. Furthermore, since both S and C are finite, Φ_* is the limit of the sequence $\langle \Phi_n \in S \to \mathcal{P}(C) \mid n \in \mathbf{N} \rangle$ defined by:

$$\Phi_n \;=\; \begin{cases} \lambda\, s \in S : \emptyset & \text{if } n = 0 \\ \mathcal{F}(\Phi_{n-1}) & \text{if } n > 0 \end{cases} \tag{12}$$

in the sense that there exists a sufficiently large $k \in \mathbf{N}$ such that $\Phi_n = \Phi_*$ for all $n \geq k$.

We say the circuit M *satisfies* the trajectory assertion A *by set-theoretic STE*, denoted by $M \models_{\text{Set}} A$, iff $\forall\, s \in S : \Phi_*(s) \cap \pi_a(s) \subseteq \pi_c(s)$. Now we are ready to state our first main result:

Theorem 1.

$$M \models_{\text{Set}} A \;\Leftrightarrow\; M \models A$$

Proof. See Appendix D.

Had we used the definition adopted in [6] for $M \models A$ (viz., (7) above), the $\Rightarrow$ direction of Theorem 1 would still be true (furthermore, notice that the obliviousness condition used in [6] is unnecessary for this part of the proof), but the $\Leftarrow$ direction would be false, as the following example shows. Consider a trivial circuit M with only one signal whose value is either 0 or 1 (i.e., $C = \{0, 1\}$), and this signal is the output of the constant source 1 (i.e., $M(p) = \{1\}$ for $\emptyset \neq p \subseteq \{0, 1\}$). Suppose that the trajectory assertion A has only three states $S = \{s_0, s_1, s_2\}$ and two transitions $R = \{(s_0, s_1), (s_0, s_2)\}$ such that $\pi_a(s_0) = \pi_a(s_1) = \pi_a(s_2) = \pi_c(s_0) = \{0, 1\}$ and $\pi_c(s_1) = \{1\}$ and $\pi_c(s_2) = \{0\}$. Then (7) is satisfied, because for any trajectory τ of M with $|\tau| = 2$, the run $\rho = \langle s_0, s_1 \rangle$ satisfies both $\tau \models_a \rho$ and $\tau \models_c \rho$. But $M \not\models_{\text{Set}} A$, since $\Phi_*(s_2) = \{1\}$, $\pi_a(s_2) = \{0, 1\}$, but $\pi_c(s_2) = \{0\}$.

3 Lattice-Theoretic STE

The definition (12) of the sequence $\langle \Phi_n \mid n \in \mathbf{N} \rangle$ above yields a simple method for computing the least fixpoint solution Φ_* of (11): just compute $\Phi_0, \Phi_1, \Phi_2, \ldots$ one by one until a fixpoint, which must be Φ_*, is reached. Then, by Theorem 1, $M \models A$ can be checked by checking $M \models_{\text{Set}} A$. Since all objects involved are finite, this scheme for checking $M \models A$, which we call the *set-theoretic STE algorithm*, is clearly effective.

Unfortunately, the set-theoretic STE algorithm is not practical except for small circuits. For, if the circuit M has m boolean signals, then its set of configurations is $\mathbf{B}^m$, where $\mathbf{B} = \{0, 1\}$ is the set of boolean values. Even with state-of-the-art BDD technologies, manipulating subsets of $\mathbf{B}^m$ is impractical for even moderately large m, say several hundred signals. But interesting circuits in the real world often contain thousands of (if not more!) signals, on which set-theoretic STE is powerless.

In this section we will describe what can be regarded as the key insight of the STE paradigm. Namely, instead of manipulating subsets of $\mathbf{B}^m$ directly, we approximate them with ternary vectors, whose sizes are only linear in m. But, to compensate for possible loss of information in the approximation process, we may have to complicate the trajectory assertion, or use a family of trajectory assertions, or both. Yet, in both cases, the number of BDD variables depends only on the trajectory assertion(s) and *not* on the circuit under analysis. This makes it possible to do STE on very large circuits without first abstracting them.

We will use many concepts and notations from the theory of partial orders and lattices [4]. In particular, the notions of *complete lattices* and *Galois connections* are reviewed in Appendices B and C, respectively.

3.1 Lattice-Theoretic Models of Circuits

Recall that $M \in \mathcal{P}(C) \to \mathcal{P}(C)$ represents a circuit (i.e., (2) is true), and that the set C of configurations of M is finite. What exactly C is, is not important until Section 3.4.

Let $(\hat{P}, \sqsubseteq)$ be a finite complete lattice of *abstract predicates* such that there is a Galois connection $\ll \subseteq \mathcal{P}(C) \times \hat{P}$. An *abstract predicate transformer* $\hat{M} \in \hat{P} \to \hat{P}$ is an *abstract interpretation* [9] of $M \in \mathcal{P}(C) \to \mathcal{P}(C)$ iff all of the following conditions hold:

$$\hat{M}(\hat{\bot}) \;=\; \hat{\bot} \tag{13}$$

$$\hat{p} \sqsubseteq \hat{q} \;\Rightarrow\; \hat{M}(\hat{p}) \sqsubseteq \hat{M}(\hat{q}) \tag{14}$$

$$p \ll \hat{p} \;\Rightarrow\; M(p) \ll \hat{M}(\hat{p}) \tag{15}$$

for all $\hat{p}, \hat{q} \in \hat{P}$ and $p \in \mathcal{P}(C)$. (13) says that $\hat{M}$ preserves the bottom $\hat{\bot}$ of $\hat{P}$, (14) that $\hat{M}$ is monotonic, and (15) that the Galois connection $\ll$ is a *simulation relation* between the two predicate lattices. Just as the Galois connection $\ll \subseteq \mathcal{P}(C) \times \hat{P}$ can be equivalently defined using an *abstraction* function $\alpha : \mathcal{P}(C) \to \hat{P}$ or a *concretization* function $\gamma : \hat{P} \to \mathcal{P}(C)$, (15) can be equivalently stated as one of the following [9]:

$$\alpha(M(p)) \;\sqsubseteq\; \hat{M}(\alpha(p)) \tag{16}$$

$$M(\gamma(\hat{p})) \;\subseteq\; \gamma(\hat{M}(\hat{p})) \tag{17}$$

for all $\hat{p} \in \hat{P}$ and $p \in \mathcal{P}(C)$. As far as we know, none of (15)–(17) has been explicitly stated in the existing literature on STE [10, 6]. It would be interesting to check whether actual implementations of STE satisfy this condition.

Note that we do *not* require of $\hat{M}$ the counterpart of (2):

$$\hat{M}(\sqcup\hat{Q}) \;=\; \sqcup\{\hat{M}(\hat{q}) \mid \hat{q} \in \hat{Q}\}$$

where $\hat{Q} \subseteq \hat{P}$, because it is not true in general. For example, suppose $\hat{M}$ abstracts a unit-delay two-input AND-gate using ternary values. Then it is reasonable to require:

$$\begin{aligned} \hat{M}(\langle 0,1,\mathsf{X}\rangle \sqcup \langle 1,0,\mathsf{X}\rangle) \;&=\; \hat{M}(\langle \mathsf{X},\mathsf{X},\mathsf{X}\rangle) \;=\; \langle \mathsf{X},\mathsf{X},\mathsf{X}\rangle \\ \hat{M}(\langle 0,1,\mathsf{X}\rangle) \sqcup \hat{M}(\langle 1,0,\mathsf{X}\rangle) \;&=\; \langle \mathsf{X},\mathsf{X},0\rangle \sqcup \langle \mathsf{X},\mathsf{X},0\rangle \;=\; \langle \mathsf{X},\mathsf{X},0\rangle \end{aligned}$$

where the first two vector components correspond to the two inputs and the last component the output. Intuitively, the join operation $\langle 0,1,\mathsf{X}\rangle \sqcup \langle 1,0,\mathsf{X}\rangle = \langle \mathsf{X},\mathsf{X},\mathsf{X}\rangle$ throws away the information that one of the inputs is 0, so $\hat{M}$ can no longer assign 0 to the output. Note, however, that the following inequality *does* hold:

$$\hat{M}(\sqcup\hat{Q}) \;\sqsupseteq\; \sqcup\{\hat{M}(\hat{q}) \mid \hat{q} \in \hat{Q}\}$$

for all $\hat{Q} \subseteq \hat{P}$. It follows from the monotonicity of $\hat{M}$.

3.2 Lattice-Theoretic Trajectory Assertions

A trajectory assertion for $\hat{M}$ is a quintuple $\hat{A} = (S, s_0, R, \hat{\pi}_a, \hat{\pi}_c)$, where the assumptions on S, s_0, and R are the same as in Section 2.2 (including (5)), and $\hat{\pi}_a \in S \to \hat{P}$ and $\hat{\pi}_c \in S \to \hat{P}$ are the antecedent and consequent labeling functions, respectively. Define $\gamma(\hat{A}) = (S, s_0, R, \gamma(\hat{\pi}_a), \gamma(\hat{\pi}_c))$, where $\gamma(\hat{\pi}_a) = \lambda\, s \in S : \gamma(\hat{\pi}_a(s))$ and $\gamma(\hat{\pi}_c) = \lambda\, s \in S : \gamma(\hat{\pi}_c(s))$. Note that $\gamma(\hat{A})$ is a trajectory assertion for M.

3.3 Lattice-Theoretic STE as DFA

Define $\hat{F} \in S \to (\hat{P} \to \hat{P})$ such that $\hat{F}(s)(\hat{p}) = \hat{M}(\hat{\pi}_a(s) \sqcap \hat{p})$ for all $s \in S$ and $\hat{p} \in \hat{P}$. Using (13) and (14), it is easy to verify that $\hat{F}(s) \in \hat{P} \to \hat{P}$ satisfies:

$$\hat{F}(s)(\hat{\bot}) \;=\; \hat{\bot} \tag{18}$$

$$\hat{p} \sqsubseteq \hat{q} \;\Rightarrow\; \hat{F}(s)(\hat{p}) \sqsubseteq \hat{F}(s)(\hat{q}) \tag{19}$$

for all $s \in S$ and $\hat{p}, \hat{q} \in \hat{P}$.

Next, define $\hat{\mathcal{F}} \in (S \to \hat{P}) \to (S \to \hat{P})$ such that:

$$\hat{\mathcal{F}}(\hat{\Phi})(s) \;=\; \begin{cases} \hat{\top} & \text{if } s = s_0 \\ \sqcup\{\, \hat{F}(s')(\hat{\Phi}(s')) \mid (s', s) \in R \,\} & \text{otherwise} \end{cases}$$

for all $\hat{\Phi} \in S \to \hat{P}$ and $s \in S$. Since $\hat{F}(s)$ is monotonic for all $s \in S$ (see (19) above), $\hat{\mathcal{F}}$ is monotonic as well, where the function space $S \to \hat{P}$ is ordered as follows: $\hat{\Phi} \sqsubseteq \hat{\Phi}' \Leftrightarrow \forall\, s \in S : \hat{\Phi}(s) \sqsubseteq \hat{\Phi}'(s)$, for all $\hat{\Phi}, \hat{\Phi}' \in S \to \hat{P}$. Hence, by Knaster-Tarski Fixpoint Theorem [4], the following fixpoint equation:

$$\hat{\Phi} \;=\; \hat{\mathcal{F}}(\hat{\Phi}) \tag{20}$$

has a least solution $\hat{\Phi}_* \in S \to \hat{P}$. Furthermore, since both S and $\hat{P}$ are finite, $\hat{\Phi}_*$ is the limit of the sequence $\langle \hat{\Phi}_n \in S \to \hat{P} \mid n \in \mathbf{N} \rangle$ defined by:

$$\hat{\Phi}_n \;=\; \begin{cases} \lambda\, s \in S : \hat{\bot} & \text{if } n = 0 \\ \hat{\mathcal{F}}(\hat{\Phi}_{n-1}) & \text{if } n > 0 \end{cases} \tag{21}$$

in the sense that there exists a sufficiently large $k \in \mathbf{N}$ such that $\hat{\Phi}_n = \hat{\Phi}_*$ for all $n \geq k$.

We say the abstract circuit $\hat{M}$ *satisfies* the abstract trajectory assertion $\hat{A}$ *by lattice-theoretic STE*, denoted by $\hat{M} \models_{\text{Lat}} \hat{A}$, iff $\forall\, s \in S : \hat{\Phi}_*(s) \sqcap \hat{\pi}_a(s) \sqsubseteq \hat{\pi}_c(s)$. Now we are ready to state our second main result:

Theorem 2. *If $\hat{M}$ is an abstract interpretation of M, then:*

$$\hat{M} \models_{\text{Lat}} \hat{A} \;\Rightarrow\; M \models_{\text{Set}} \gamma(\hat{A})$$

Proof. See Appendix E.

The converse of Theorem 2 is not true. For example, consider a circuit with five signals $\langle i_1, i_2, j_1, j_2, o \rangle$, where j_1 (resp., j_2) is i_1 (i_2) delayed by one unit of time and o is the unit-delayed AND of j_1 and j_2. Suppose the trajectory assertion has five states $\{s_0, s_1, s_1', s_2, s_3\}$ and five transitions $\{(s_0, s_1), (s_0, s_1'), (s_1, s_2), (s_1', s_2), (s_2, s_3)\}$ and the following labeling: $\pi_a(s_1) = \langle 0, 1, \mathsf{X}, \mathsf{X}, \mathsf{X} \rangle$, $\pi_a(s_1') = \langle 1, 0, \mathsf{X}, \mathsf{X}, \mathsf{X} \rangle$, $\pi_c(s_3) = \langle \mathsf{X}, \mathsf{X}, \mathsf{X}, \mathsf{X}, 0 \rangle$; all other labels are $\langle \mathsf{X}, \mathsf{X}, \mathsf{X}, \mathsf{X}, \mathsf{X} \rangle$. Intuitively, the antecedent at s_1 (resp., s_1') assumes that $i_1 = 0$ and $i_2 = 1$ (resp., $i_1 = 1$ and $i_2 = 0$) at time 1, and the consequent at s_3 checks that at time 3, $o = 0$ regardless of which assumption was used. It is easy to verify that for this example, $M \models_{\text{Set}} \gamma(\hat{A})$ but $\hat{M} \not\models_{\text{Lat}} \hat{A}$. And the reason is simple: at time 2, when the information from s_1 and s_1' is merged at s_2, we have:

$$\{\langle 0, 1 \rangle\} \cup \{\langle 1, 0 \rangle\} = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\} \quad \text{but} \quad \langle 0, 1 \rangle \sqcup \langle 1, 0 \rangle = \langle \mathsf{X}, \mathsf{X} \rangle$$

the latter of which loses information. Clearly, this merge could be avoided by duplicating s_2 and s_3, so that there is a separate copy of them to deal with each of the assumptions $\pi_a(s_1)$ and $\pi_a(s_1')$. But then the number of states in the trajectory assertion increases. This kind of trade-offs between complexity and precision is typical of STE.

3.4 A BDD-Based Algorithm for Lattice-Theoretic STE

Up to this point, except in a few examples, we have not needed to specify what exactly the set C of configurations is, except that C should be finite. This makes our theory more general. But in order to have a BDD-based implementation, we have to make up our mind now as to what C is. Thus, in this subsection, we shall assume that $C = \mathbf{B}^m$ for some $m \in \mathbf{N}$. In other words, M is a boolean circuit with m signals. Furthermore, we assume that the abstract circuit $\hat{M}$ operates on ternary vectors, i.e., $\hat{P} = \mathbf{T}_\perp^m$. How sets of boolean vectors can be approximated by ternary vectors is explained in Appendix C.3.

Similar to the set-theoretic case, the definition (21) of the sequence $\langle \hat{\Phi}_n \mid n \in \mathbf{N} \rangle$ yields a simple algorithm for checking $\hat{M} \models_{\text{Lat}} \hat{A}$: compute $\hat{\Phi}_0, \hat{\Phi}_1, \hat{\Phi}_2, \ldots$ one by one until a fixpoint, which must be $\hat{\Phi}_*$, is reached; then check $\hat{M} \models_{\text{Lat}} \hat{A}$ using its definition. Note that since the converse of Theorem 2 is not true, this algorithm, which we call the *lattice-theoretic STE algorithm*, can give falsely negative answers (i.e., when $\hat{M} \not\models_{\text{Lat}} \hat{A}$ but $M \models \gamma(\hat{A})$). But, by virtue of Theorems 1 and 2, it can never produce falsely positive answers (i.e., $\hat{M} \models_{\text{Lat}} \hat{A}$ does imply $M \models \gamma(\hat{A})$). We now argue that the lattice-theoretic STE algorithm can be implemented using BDDs in a straightforward manner.

First, notice that every ternaray value $t \in \mathbf{T}$ can be encoded with two boolean values: $B_0(t) = (0 \sqsubseteq t)$ and $B_1(t) = (1 \sqsubseteq t)$. With this encoding, join and meet are implemented by: $B_i(t \sqcup t') = B_i(t) \vee B_i(t')$ and $B_i(t \sqcap t') = B_i(t) \wedge B_i(t')$, where $i \in \mathbf{B}$. For any $m \in \mathbf{N}$, this encoding and the associated join and meet operations can be extended componentwise to $\mathbf{T}_\perp^m$. Note that $\perp$ has multiple encodings (viz., all m-tuples of boolean pairs in which at least one of the pairs is such that $B_0 = B_1 = 0$).

Without loss of generality, suppose the state space S of the trajectory assertion is $\mathbf{B}^k$, for some $k \in \mathbf{N}$. With the above encoding of ternary values, the objects manipulated by the lattice-theoretic STE algorithm have the following "types": $R \in \mathbf{B}^k \times \mathbf{B}^k \to \mathbf{B}$ and $\hat{\pi}_a, \hat{\pi}_c, \hat{\Phi}_n, \hat{\Phi}_* \in \mathbf{B}^k \to (\mathbf{B} \times \mathbf{B})^m$, for all $n \in \mathbf{N}$. It is not hard to see that these objects can all be represented by BDDs on at most $2k$ variables, and that $\hat{\mathcal{F}}$ and the checking of $\hat{M} \models_{\text{Lat}} \hat{A}$ can be implemented by BDD operations on these BDDs *provided that* the output of the abstract circuit $\hat{M} \in \mathbf{T}_\perp^m \to \mathbf{T}_\perp^m$ for any given input can be computed without ever having to represent $\hat{M}$ itself as BDDs (which would require $2m$ variables). Real-world STE implementations amply demonstrate that this proviso is practical.

We emphasize again that the maximum number of boolean variables needed by our algorithm, $2k$, depends only on the trajectory assertion and *not* on the circuit. Of course, this independence is somewhat illusory, since the possible loss of information in the approximation by ternary vectors may necessitate more complex state-transition structure in the trajectory assertion, which would increase k. Furthermore, note that our formulation so far has been "unparameterized" in the sense that the antecedents and consequents are simple ternary vectors without parameters. In fact, they can be parameterized by boolean variables, so that a single run of the parameterized algorithm is equivalent to multiple runs of the unparameterized algorithm, one for each truth assignment to the boolean parameters. Needless to say, such parameters increase further the total number of boolean variables.

4 Future Research

Despite its simplicity, the lattice-theoretic STE algorithm described above does not seem to have ever been implemented. Since we do not see any reason *a priori* why it should

not give rise to implementations as efficient as any current implementations of STE, it would be interesting to try to implement it and use it on real-world circuits.

For specifying properties of hardware, which are usually highly parallel, one would like to have powerful parallel programming constructs in order to express the finite-state machine part of the trajectory assertion in an elegant manner. Our past experience [2] shows that synchronous languages [1, 5] may provide such constructs. Furthermore, programs in synchronous languages can be automatically translated into finite-state machines in the form of circuits, which are then readily representable by BDDs [1, 2].

Given our observation that STE is a form of DFA, two natural questions arise. First, is there anything in the vast literature on DFA [8] that is useful to STE? Conversely, since most traditional DFA algorithms operate on bit vectors [8], could some forms of BDD-based algorithms (such as our STE algorithm) benefit traditional DFA? We hope that our observation about the connection between STE and DFA can lead to cross-fertilization of research ideas between these two fields.

Last but not least, it may be possible to generalize STE to express and reason about liveness properties. Doing so would make STE even more useful than it already is now. Unfortunately, we do not have anything concrete to report on this problem.

Acknowledgments

The author is grateful to Pascalin Amagbegnon, John Mark Bouler, Pei-Hsin Ho, Marten van Hulst, and Carl Seger for comments and encouragements, and especially to Victor Konrad for giving him time to work on this paper.

References

1. Gérard Berry, "The Foundations of Esterel", in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling and M. Tofte, editors, MIT Press, 1998.
2. Ching-Tsun Chou, Jiun-Lang Huang, and Masahiro Fujita, "A High-Level Language for Programming Complex Temporal Behaviors and Its Translation into Synchronous Circuits", poster presentation, *IFIP Conference on Hardware Description Languages*, Apr. 1997.
3. E.M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.
4. B.A. Davey and H.A. Priestley, *Introduction to Lattices and Order*, Cambridge University Press, 1990.
5. N. Halbwachs, *Synchronous Programming of Reactive Systems*, Kluwer Academic Publishers, 1993.
6. Alok Jain, "Formal Hardware Verification by Symbolic Trajectory Evaluation", Ph.D. Dissertation supervised by Randal E. Bryant, Carnegie-Mellon University, July 1997.
7. G.A. Kildall, "A Unified Approach to Global Program Optimization", pp.194–206 of *Conf. Rec. of 1st ACM Symp. on Principles of Programming Languages (POPL'73)*, Oct. 1973.
8. Steven S. Muchnick, *Advanced Compiler Design and Implementation*, Morgan Kaufmann Publishers, 1997.
9. D.A. Schmidt and B. Steffen, "Data-Flow Analysis as Model Checking of Abstract Interpretations", invited tutorial paper, *Proc. 5th Static Analysis Symposium*, G. Levi (ed.), Pisa, Sep. 1998, Springer LNCS 1503.
10. Carl-Johan H. Seger and Randal E. Bryant, "Formal Verification by Symbolic Evaluation of Partially-Ordered Trajectories", *Formal Methods in System Designs*, Vol. 6, No. 2, pp. 147–189, March 1995.

A Sequences

Let $\mathbf{N} = \{0, 1, 2, \cdots\}$ be the set of natural numbers. For any set V and any $n \in \mathbf{N}$, V^n (resp., V^+ and V^*) denotes the set of all finite sequences of length n (resp., positive and nonnegative lengths) over V.

Let $\sigma, \tau \in V^*$. The length of σ is denoted by $|\sigma|$, the concatenation of σ followed by τ by $\sigma^\frown\tau$, and σ being a prefix of τ by $\sigma \preceq \tau$. For any $i \in \mathbf{N}$ with $0 \leq i < |\sigma|$, the i-th element of σ is denoted by $\sigma[i]$. (Note that we index sequence elements starting from 0 instead of 1.) The last element of σ is denoted by $\mathit{last}(\sigma)$, i.e., $\mathit{last}(\sigma) = \sigma[\,|\sigma| - 1\,]$. The empty sequence (i.e., the sequence whose length is 0) is denoted by $\langle\rangle$. A sequence consisting of elements $v_0, v_1, v_2, \cdots, v_{n-1} \in V$ (in that order) is written as $\langle v_0, v_1, v_2, \cdots, v_{n-1}\rangle$.

We use the terms "sequences" and "vectors" interchangeably; the elements of vectors are sometimes referred to as "components".

B Complete Lattices

A *complete lattice* is a poset $(P, \sqsubseteq)$ in which the meet and join of elements of any subset $Q \subseteq P$, denoted by $\sqcap Q$ and $\sqcup Q$ respectively, always exist. Intuitively, we think of the elements of a complete lattice as "predicates", so that $\sqcap$, $\sqcup$, and $\sqsubseteq$ corresponds to "conjunction", "disjunction", and "implication", respectively.

For any set V, its power set $\mathcal{P}(V)$, ordered by set inclusion $\subseteq$, forms a complete lattice. Here $\sqcap$, $\sqcup$, and $\sqsubseteq$ are $\cap$, $\cup$, and $\subseteq$, respectively.

Let $\mathbf{T} = \{0, 1, \mathsf{X}\}$ be the set of *ternary values*, where X denotes an unknown value. Intuitively, X signifies a lack of information: it could be 0, it could be 1; we simply don't know. We partially order $\mathbf{T}$ as follows:[2] $0 \sqsubseteq \mathsf{X}$ and $1 \sqsubseteq \mathsf{X}$. For any $m \in \mathbf{N}$, this order on $\mathbf{T}$ can be extended component-wise to $\mathbf{T}^m$:

$$\langle t_0, \cdots, t_{m-1}\rangle \sqsubseteq \langle t'_0, \cdots, t'_{m-1}\rangle \;\Leftrightarrow\; t_0 \sqsubseteq t'_0 \wedge \cdots \wedge t_{m-1} \sqsubseteq t'_{m-1}$$

But $(\mathbf{T}^m, \sqsubseteq)$ is not a complete lattice, because it lacks a bottom. We can fix this by introducing a special bottom element, $\bot$, such that $\bot \sqsubseteq t$ and $\bot \neq t$ for all $t \in \mathbf{T}^m$. Now $\mathbf{T}^m_\bot = \mathbf{T}^m \cup \{\bot\}$, ordered by $\sqsubseteq$, is indeed a complete lattice. We denote the top element $\langle \mathsf{X}, \cdots, \mathsf{X}\rangle$ of $\mathbf{T}^m_\bot$ by $\top$.

C Galois Connections

C.1 Galois Connections as Relations

Let $(P^\flat, \sqsubseteq^\flat)$ and $(P^\sharp, \sqsubseteq^\sharp)$ be complete lattices[3] of "concrete predicates" and "abstract predicates", respectively. In the sequel we will drop the superscripts "$\flat$" and "$\sharp$" from the partial orders $\sqsubseteq^\flat$ and $\sqsubseteq^\sharp$ and the meet and join operations they induce, since they will always be clear from the context.

[2] Note that our ordering of $\mathbf{T}$ is the reverse of that used in [10, 6]. We do so because we want to make clear that the ordering of $\mathbf{T}$ is an abstraction of set inclusion.

[3] The theory of Galois connections can in fact be developed for general posets, but doing so requires the introduction of many inconvenient side-conditions. In any case, we only need Galois connections between complete lattices in this paper.

A *Galois connection* [4, 9][4] from $P^\flat$ to $P^\sharp$ is a binary relation $\ll \subseteq P^\flat \times P^\sharp$, where $p^\flat \ll p^\sharp$ reads: "$p^\flat$ can be approximated by $p^\sharp$", such that for all $Q^\flat \subseteq P^\flat$ and $Q^\sharp \subseteq P^\sharp$:

$$Q^\flat \ll Q^\sharp \;\Leftrightarrow\; \sqcup Q^\flat \;\ll\; \sqcap Q^\sharp \tag{22}$$

where we define: $Q^\flat \ll Q^\sharp \Leftrightarrow \forall\, p^\flat \in Q^\flat : \forall\, p^\sharp \in Q^\sharp : p^\flat \ll p^\sharp$. Intuitively, (22) says that the approximation relation $\ll$ is an "extension" of the partial orders *inside* $P^\flat$ and $P^\sharp$ to *between* $P^\flat$ and $P^\sharp$.

C.2 Galois Connections as Functions

The usual definitions of Galois connections in the literature [4, 9] are in terms of an *abstraction* function $\alpha : P^\flat \to P^\sharp$ and a *concretization* function $\gamma : P^\sharp \to P^\flat$, which in our framework can be derived from $\ll$ as follows:

$$\alpha(p^\flat) \;=\; \sqcap\{p^\sharp \in P^\sharp \mid p^\flat \ll p^\sharp\} \tag{23}$$
$$\gamma(p^\sharp) \;=\; \sqcup\{p^\flat \in P^\flat \mid p^\flat \ll p^\sharp\} \tag{24}$$

for all $p^\flat \in P^\flat$ and $p^\sharp \in P^\sharp$. Intuitively, $\alpha(p^\flat)$ (respectively, $\gamma(p^\sharp)$) is the "most precise approximation" of $p^\flat$ ($p^\sharp$) in $P^\sharp$ ($P^\flat$).

Conversely, the relation $\ll$ can be derived from α or γ as follows:

$$p^\flat \ll p^\sharp \;\Leftrightarrow\; \alpha(p^\flat) \sqsubseteq p^\sharp \tag{25}$$
$$p^\flat \ll p^\sharp \;\Leftrightarrow\; p^\flat \sqsubseteq \gamma(p^\sharp) \tag{26}$$

for all $p^\flat \in P^\flat$ and $p^\sharp \in P^\sharp$. It is easy to see from (23)–(26) how α and γ can be derived from each other.

It is not hard to show that α has the following properties:

$$\alpha(\bot^\flat) \;=\; \bot^\sharp \tag{27}$$
$$p^\flat \sqsubseteq q^\flat \;\Rightarrow\; \alpha(p^\flat) \sqsubseteq \alpha(q^\flat) \tag{28}$$
$$\alpha(\sqcup Q^\flat) \;=\; \sqcup\{\alpha(q^\flat) \in P^\sharp \mid q^\flat \in Q^\flat\} \tag{29}$$

for all $p^\flat, q^\flat \in P^\flat$ and $Q^\flat \subseteq P^\flat$. (In fact, (29) implies both (27) and (28).) Similarly, γ has the following properties:

$$\gamma(\top^\sharp) \;=\; \top^\flat \tag{30}$$
$$p^\sharp \sqsubseteq q^\sharp \;\Rightarrow\; \gamma(p^\sharp) \sqsubseteq \gamma(q^\sharp) \tag{31}$$
$$\gamma(\sqcap Q^\sharp) \;=\; \sqcap\{\gamma(q^\sharp) \in P^\flat \mid q^\sharp \in Q^\sharp\} \tag{32}$$

for all $p^\sharp, q^\sharp \in P^\sharp$ and $Q^\sharp \subseteq P^\sharp$. (And (32) implies both (30) and (31).)

C.3 Galois Connection from $\mathcal{P}(\mathbf{B}^m)$ to $\mathbf{T}^m_\bot$

For any $m \in \mathbf{N}$, there is a natural Galois connection $\ll$ from $\mathcal{P}(\mathbf{B}^m)$ to $\mathbf{T}^m_\bot$, which is most conveniently defined by specifying its concretization function $\gamma : \mathbf{T}^m_\bot \to \mathcal{P}(\mathbf{B}^m)$:

$$\begin{aligned}\gamma(\langle t_0, \cdots, t_{m-1}\rangle) \;=\; \{\; &\langle b_0, \cdots, b_{m-1}\rangle \in \mathbf{B}^m \mid \\ &\forall i \in \mathbf{N} : 0 \le i < m \wedge t_i \neq \mathsf{X} \;\Rightarrow\; b_i = t_i \;\} \\ \gamma(\bot) \;=\; &\emptyset\end{aligned}$$

[4] We should point out that in the formulation in [4], the partial order on $P^\sharp$ is reversed.

for all $\langle t_0, \cdots, t_{m-1} \rangle \in \mathbf{T}^m$. In other words, for any ternary vector $t \in \mathbf{T}^m$, $\gamma(t)$ is the set of all boolean vectors $\in \mathbf{B}^m$ that agree with t on all non-X components (so X's can be thought of as "wild cards"), and $\gamma(\bot)$ is the empty set. Note that γ is in fact a bijection from $\mathbf{T}^m_\bot$ to those subsets of $\mathbf{B}^m$ that are (hyper)cubes.

From γ, the Galois connection $\ll \subseteq \mathcal{P}(\mathbf{B}^m) \times \mathbf{T}^m_\bot$ and the abstraction function $\alpha : \mathcal{P}(\mathbf{B}^m) \to \mathbf{T}^m_\bot$ can be easily derived:

$$\begin{aligned} b \ll t &\Leftrightarrow b \subseteq \gamma(t) \\ \alpha(b) &= \sqcap \{ t \in \mathbf{T}^m_\bot \mid b \subseteq \gamma(t) \} \end{aligned}$$

for all $b \in \mathcal{P}(\mathbf{B}^m)$ and $t \in \mathbf{T}^m_\bot$. In other words, $b \ll t$ iff the cube corresponding to t contains b, and $\alpha(b)$ is the element of $\mathbf{T}^m_\bot$ that corresponds to the smallest cube in $\mathbf{B}^m$ that contains b.

D Proof of Theorem 1

We prove the two directions of $\Leftrightarrow$ separately.

The $\Rightarrow$ direction: Suppose this is not true, i.e., $M \models_{\text{Set}} A$ but $M \not\models A$. Then $M \not\models A$ implies that there exist $\tau \in \mathit{Traj}(M)$ and $\rho \in \mathit{Runs}(A)$ such that $|\tau| = |\rho|$, $\tau \models_a \rho$, and $\tau' \models_c \rho'$, but $\mathit{last}(\tau) \notin \pi_c(\mathit{last}(\rho))$, where τ' and ρ' are the prefixes of τ and ρ respectively such that $|\tau'| = |\tau| - 1$ and $|\rho'| = |\rho| - 1$. We claim that for all $i \in \mathbf{N}$ with $0 \leq i < |\tau|$, $\tau[i] \in \Phi_*(\rho[i])$. This is proved by induction on i. The base case $i = 0$ is trivial, since $\Phi_*(s_0) = C$. For the induction step, assume the claim is true for $i < |\tau| - 1$. Then $\tau[i] \in \Phi_*(\rho[i]) \cap \pi_a(\rho[i])$, since $\tau \models_a \rho$. So $\tau[i+1] \in F(\rho[i])(\Phi_*(\rho[i]))$, since $\tau \in \mathit{Traj}(M)$. But, since Φ_* is a solution of (11), $F(\rho[i])(\Phi_*(\rho[i])) \subseteq \Phi_*(\rho[i+1])$. This completes the induction step, so the claim is true. But the claim implies that $\mathit{last}(\tau) \in \Phi_*(\mathit{last}(\rho)) \cap \pi_a(\mathit{last}(\rho))$, which implies $\mathit{last}(\tau) \in \pi_c(\mathit{last}(\rho))$ because $M \models_{\text{Set}} A$. So $\mathit{last}(\tau) \in \pi_c(\mathit{last}(\rho))$ and $\mathit{last}(\tau) \notin \pi_c(\mathit{last}(\rho))$, a contradiction.

The $\Leftarrow$ direction: Since $F(s)$ is distributive over arbitrary union for all $s \in S$ (see (10) above), a well-known result from DFA [7] states that the least fixpoint solution Φ_* of (11) is in fact the same as the *union-over-all-runs* solution of (11). More precisely, Φ_* satisfies the following equation:

$$\Phi_*(s) = \begin{cases} C & \text{if } s = s_0 \\ \bigcup \{ G(\rho') \mid \rho' \in \mathit{Runs}(A) \wedge (\mathit{last}(\rho'), s) \in R \} & \text{otherwise} \end{cases} \tag{33}$$

for all $s \in S$, where $G : \mathit{Runs}(A) \cup \{\langle\rangle\} \to \mathcal{P}(C)$ is defined inductively by:

$$\begin{aligned} G(\langle\rangle) &= C \\ G(\rho^\frown \langle s \rangle) &= F(s)(G(\rho)) \end{aligned}$$

Let $c \in C$ and $s \in S$. Using the definitions of G, F, and M, (33) can be rephrased as:

$$\begin{aligned} c \in \Phi_*(s) \Leftrightarrow\ & \exists\, \tau \in \mathit{Traj}(M) : \exists\, \rho \in \mathit{Runs}(A) : \\ & |\tau| = |\rho| \wedge \mathit{last}(\tau) = c \wedge \mathit{last}(\rho) = s \wedge \\ & \forall\, i \in \mathbf{N} : 0 \leq i < |\tau| - 1 \Rightarrow \tau[i] \in \pi_a(\rho[i]) \end{aligned}$$

Conjoining $c \in \pi_a(s)$ to both sides, we get:

$$\begin{aligned} c \in \Phi_*(s) \cap \pi_a(s) \;\Leftrightarrow\; & \exists\, \tau \in \mathit{Traj}(M) : \exists\, \rho \in \mathit{Runs}(A) : \\ & |\tau| = |\rho| \;\wedge\; \mathit{last}(\tau) = c \;\wedge\; \mathit{last}(\rho) = s \;\wedge \\ & \tau \models_a \rho \end{aligned}$$

Now the definition of $M \models A$ shows that the $\Leftarrow$ direction is indeed true.

E Proof of Theorem 2

Throughout this proof, we will freely use the equivalence (26) that $p \ll \hat{p} \;\Leftrightarrow\; p \subseteq \gamma(\hat{p})$ for all $p \in \mathcal{P}(C)$ and $\hat{p} \in \hat{P}$. For set-theoretic STE, the notations are exactly the same as in Section 2.3 and Appendix D, except that the (concrete) trajectory assertion is $\gamma(\hat{A}) = (S, s_0, R, \gamma(\hat{\pi}_a), \gamma(\hat{\pi}_c))$ instead of A.

First, we claim that:

$$p \ll \hat{p} \;\Rightarrow\; F(s)(p) \ll \hat{F}(s)(\hat{p}) \tag{34}$$

for all $p \in \mathcal{P}(C)$, $\hat{p} \in \hat{P}$, and $s \in S$. This is proved as follows:

$$\begin{array}{rcll} \gamma(\hat{F}(s)(\hat{p})) & = & \gamma(\hat{M}(\hat{\pi}_a(s) \sqcap \hat{p})) & \{\text{ Definition of } \hat{F} \} \\ & \supseteq & M(\gamma(\hat{\pi}_a(s) \sqcap \hat{p})) & \{\text{ (17) }\} \\ & = & M(\gamma(\hat{\pi}_a(s)) \cap \gamma(\hat{p})) & \{\text{ (32) }\} \\ & \supseteq & M(\gamma(\hat{\pi}_a(s)) \cap p) & \{\, p \ll \hat{p} \text{ and (4) }\} \\ & = & F(s)(p) & \{\text{ Definition of } F \} \end{array}$$

where $\{\cdots\}$'s on the right give the justifications of the steps.

Second, we claim that:

$$\Phi_*(s) \;\ll\; \hat{\Phi}_*(s) \tag{35}$$

for all $s \in S$. Since $\Phi_*(s) = \lim \Phi_n(s)$ and $\hat{\Phi}_*(s) = \lim \hat{\Phi}_n(s)$, it suffices to prove that $\Phi_n(s) \ll \hat{\Phi}_n(s)$ for all $s \in S$ and $n \in \mathbf{N}$. This is proved by induction on n. The base case is trivial, since a Galois connection always relates the two bottoms. For the induction step, assume $\Phi_n(s) \ll \hat{\Phi}_n(s)$ for all $s \in S$. That $\Phi_{n+1}(s_0) \ll \hat{\Phi}_{n+1}(s_0)$ is also trivial, since a Galois connection always relates the two tops. For any $s_0 \neq s \in S$, we have:

$$\begin{array}{rcll} \gamma(\hat{\Phi}_{n+1}(s)) & = & \gamma(\sqcup\{\hat{F}(s')(\hat{\Phi}_n(s')) \mid (s', s) \in R\}) & \{\text{ (21) }\} \\ & \supseteq & \cup\{\gamma(\hat{F}(s')(\hat{\Phi}_n(s'))) \mid (s', s) \in R\} & \{\text{ (31) }\} \\ & \supseteq & \cup\{F(s')(\Phi_n(s')) \mid (s', s) \in R\} & \{\, \Phi_n(s) \ll \hat{\Phi}_n(s) \text{ and (34) }\} \\ & = & \Phi_{n+1}(s) & \{\text{ (12) }\} \end{array}$$

This completes the induction step, so the claim is true.

Finally, suppose $\hat{M} \models_{\text{Lat}} \hat{A}$. Then, for all $s \in S$, we have:

$$\begin{array}{rcll} \gamma(\hat{\pi}_c(s)) & \supseteq & \gamma(\hat{\Phi}_*(s) \sqcap \hat{\pi}_a(s)) & \{\, \hat{M} \models_{\text{Lat}} \hat{A} \text{ and (31) }\} \\ & = & \gamma(\hat{\Phi}_*(s)) \cap \gamma(\hat{\pi}_a(s)) & \{\text{ (32) }\} \\ & \supseteq & \Phi_*(s) \cap \gamma(\hat{\pi}_a(s)) & \{\text{ (35) }\} \end{array}$$

Therefore, $M \models_{\text{Set}} \gamma(\hat{A})$.