中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-07-002

Automatic Derivation of Compositional Rules in Automated Compositional Reasoning

Bow-Yaw Wang

January 15, 2007 || Technical Report No. TR-IIS-07-002

http://www.iis.sinica.edu.tw/LIB/TechReport/tr2007/tr07.html

AUTOMATIC DERIVATION OF COMPOSITIONAL RULES IN AUTOMATED COMPOSITIONAL REASONING

BOW-YAW WANG

Abstract. Soundness of compositional reasoning rules depends on computational models and sometimes is rather involved. Since it is tedious to establish new rules, verifiers are forced to mould verification problems into a handful of proof rules available to them. In this paper, a syntactic approach to establishing soundness of proof rules in automated compositional reasoning is shown. Not only can our work justify all proof rules known to us automatically, but also derive new circular rules by intuitionistic reasoning without human intervention. Practitioners can now develop their own rules in automated compositional reasoning through learning rather easily.

1. Introduction

One of the most effective techniques to alleviate the state-explosion problem in formal verification is compositional reasoning. The technique divides compositions and conquers the verification problem by parts. The decomposition however cannot be done naively. Oftentimes, components function correctly only in specific contexts; they may not work separately. Assume-guarantee reasoning circumvents the problem by introducing environmental assumptions. Components are not checked against arbitrary environment, but verified under certain assumptions. Nevertheless, making proper environmental assumptions requires clairvoyance. It is so tedious a task that one would like to do without.

In [4], the problem is solved by a novel application of the L^* learning algorithm. Consider, for example, the following assume-guarantee rule where $M \models P$ denotes that the system M satisfies the property P.

$$\frac{M_0 \| A \models P \qquad M_1 \models A}{M_0 \| M_1 \models P}$$

To apply the rule, the new paradigm constructs an assumption A satisfying all premises by automated supervised learning. By the completeness of the proof rule and the L^* algorithm, it is guaranteed that a proper assumption A will be construed if the composition does satisfy the property.

But few proof rules have been established in automated compositional reasoning. Since proofs of their soundness are essentially tedious case analysis, verifiers may be reluctant to develop new rules lest introducing flaws in the paradigm. Moreover, existing proof rules for other computational models may not apply because their proofs of soundness often depend on different assumptions. Subsequently, all verification tasks must be moulded into a handful of proof rules available in automated compositional reasoning. The effectiveness and applicability of the new paradigm are therefore impeded.

In this paper, a proof-theoretic technique for establishing soundness of proof rules in automated compositional reasoning is developed. We begin with the simple observation that regular languages are closed under Boolean operations and form a Boolean algebra. The proof system LK for classical logic can hence be used to deduce relations among regular sets syntactically.

Nonetheless classical logic has its limitation. Consider the following circular compositional rule in [2].

$$\frac{M_0 \| P_1 \models P_0 \qquad P_0 \| M_1 \models P_1}{M_0 \| M_1 \models P_0 \| P_1}$$

If we think compositions as conjunctions and satisfactions as implications, it is easy to see that the rules is not sound even in the Boolean domain. Hence the circular compositional rule cannot be derived in any sound proof system for Boolean algebra.

Following Abadi and Plotkin's work in [1], we show that non-empty, prefix-closed regular languages form a Heyting algebra. Hence the proof system LJ for intuitionistic logic can be used to deduce relations among them. Moreover, a circular inference rule in [1] is shown to be sound in the Heyting algebra. After adding it in the system LJ, we are able to derive the soundness of the aforementioned circular compositional proof rule syntactically.

With the help of modern proof assistants, we can in fact justify compositional rules automatically. For the classical interpretation, the proof assistant Isabelle [9] is used to establish the soundness of all proof rules in [4, 3]. The proof assistant Coq [8] proves the soundness of a circular compositional rule and variants of assume-guarantee rules in [4, 3] by intuitionistic reasoning. The proof search engines in both tools are able to justify all rules without human intervention. Verifiers are hence liberated from tedious case analysis in proofs of soundness and can establish their own rules effortlessly.

Many compositional reasoning rules have been proposed in literature (for a comprehensive introduction, see [5]). The present work focuses on the rules used in automated compositional reasoning via learning [4, 3]. Instead of proposing new rules for the paradigm, a systematic way to establishing compositional rules is developed. Since it is impossible to enumerate all rules for various scenarios, we feel our work could be more useful to practitioners.

Although we are motivated by the advent of automated compositional reasoning, our techniques borrow extensively from Abadi and Plotkin [1]. There are, nonetheless, a couple of essential differences. The computational model in [1] is very abstract where a property is a set of sequences consisting of alternating states and agents. Ours is, on the other hand, very specific where all system behaviors and properties are but regular languages. Secondly, all our constructions are shown to preserve regularity. They therefore fit perfectly in the context of automated compositional reasoning.

The paper is organized as follows. After the preliminaries in Section 2, a classical interpretation of propositional logic over regular languages and its limitation is presented in Section 3. The intuitionistic interpretation is then followed in Section 4. Applications are illustrated in Section 5. We briefly discuss the completeness issues in Section 6. Finally, we conclude the paper in Section 7.

2. Preliminaries

The systems LK for classical logic and LJ for intuitionistic logic are briefly described in this section. It is known that the system LK is sound and complete for Boolean algebra while the system LJ for Heyting algebra. We begin the definitions of algebraic models and their properties. They are followed by the descriptions of proof systems. Elementary results in finite automata theory are also recalled. For more detailed exposition, please refer to [7, 10, 6].

A *partially ordered set* $\mathcal{P} = (P, \leq)$ consists of a set P and a reflexive, anti-symmetric, and transitive binary relation $\leq$ over P. Given a set $A \subseteq P$, an element u is an *upper bound* of A if $a \leq u$ for all $a \in A$. The element u is a *least upper bound* if u is an upper bound of A and $u \leq v$ for any upper bound v of A. Lower bounds and greatest lower bounds of A can be defined symmetrically. Since $\leq$ is anti-symmetric, it is straightforward to verify that least upper bounds and greatest lower bounds for a fixed set are unique.

Definition 1. *A* lattice $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$ *is a partially ordered set where the least upper bound* $(a \sqcup b)$ *and the greatest lower bound* $(a \sqcap b)$ *exist for any* $\{a, b\}$ *with* $a, b \in L$.

It is easy to verify that $a \leq b$ if and only if $a \sqcup b = b$ and $a \sqcap b = a$ in any lattice.

Lemma 1. *Let* $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$ *be a lattice. Then* $a \leq b$ *if and only if* $a \sqcup b = b$ *and* $a \sqcap b = a$ *for* $a, b \in L$.

Proof. Suppose $a \leq b$. We have $a \sqcap b \leq a$ and $b \leq a \sqcup b$ by definition of $a \sqcap b$ and $a \sqcup b$. Since a and b are respectively a lower bound and an upper bound of $\{a, b\}$, $a \leq a \sqcap b$ and $a \sqcup b \leq b$ as well.

$a = a \sqcap b \leq b$ and $a \leq a \sqcup b = b$ by definition. □

Given a lattice $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$, we say it is *distributive* if $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$ and $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$ for $a, b, c \in L$. The lattice $\mathcal{L}$ is *bounded* if it has a *unit* $1 \in L$ and a *zero* $0 \in L$ such that $0 \leq a$ and $a \leq 1$ for all $a \in L$.

In a bounded lattice $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$, b is a *complement* of a if $a \sqcup b = 1$ and $a \sqcap b = 0$. A bounded lattice $\mathcal{L}$ is *complemented* if each element has a complement. It can be shown that complements are unique in any bounded distributive lattice.

Lemma 2. *Let* $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$ *be a bounded distributive lattice. If* a' *and* a'' *are complements of* a*, then* $a' = a''$.

Proof. Recall that $a \sqcup a' = 1$, $a \sqcap a' = 0$, $a \sqcup a'' = 1$, and $a \sqcap a'' = 0$. Then $a' = a' \sqcap 1 = a' \sqcap (a \sqcup a'') = (a' \sqcap a) \sqcup (a' \sqcap a'') = a' \sqcap a''$. Hence $a' \leq a''$. Symmetrically, $a'' \leq a'$. Therefore $a' = a''$. □

A Boolean algebra is but a complemented distributive lattice. Since complements, zero, and unit are unique, we give them distinct notations in the following definition.

Definition 2. *A* Boolean algebra $\mathcal{B} = (B, \leq, \sqcup, \sqcap, -, 0, 1)$ *is a complemented distributive lattice where*

- $a \sqcup b$ *and* $a \sqcap b$ *are the least upper bound and the greatest lower bound of* a *and* b *respectively;*
- $-a$ *is the complement of* a*; and*
- 0 *and* 1 *are its zero and unit respectively.*

The complement of a can be viewed as the greatest element incompatible with a (that is, the greatest c such that $a \sqcap c = 0$). The view can be generalized to define complements relative to arbitrary elements as follows.

Definition 3. *Let $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$ be a lattice. For any a and b in L, a* pseudo-complement *of a* relative *to b is an element p in L such that*

$$\textit{for all } c, c \leq p \textit{ if and only if } a \sqcap c \leq b.$$

Since a lattice is also a partial ordered set, pseudo-complements of a relative to b are in fact unique. We hence write $a \Rightarrow b$ for the pseudo-complement of a relative to b. A lattice $\mathcal{L}$ is *relatively pseudo-complemented* if the pseudo-complement of a relative to b exists for all a and b. It can be shown that the unit exists in any relatively pseudo-complemented lattice.

A Heyting algebra can now be defined formally as a relative pseudo-complemented lattice with a zero.

Definition 4. *A* Heyting algebra $\mathcal{H} = (H, \leq, \sqcup, \sqcap, \Rightarrow, 0, 1)$ *is a relatively pseudo-complemented lattice with a zero where*

- *$a \sqcup b$ and $a \sqcap b$ are the least upper bound and the greatest lower bound of a and b respectively;*
- *$a \Rightarrow b$ is the pseudo-complement of a relative to b; and*
- *0 and 1 are its zero and unit respectively.*

The following lemma relates pseudo-complements with the partial order in a lattice. It is very useful when the syntactic deduction and semantic interpretation are linked together later in our exposition.

Lemma 3. *Let $\mathcal{L} = (L, \leq, \sqcup, \sqcap)$ be a relatively pseudo-complemented lattice. Then $a \Rightarrow b = 1$ if and only if $a \leq b$.*

Proof. Recall that for any $c \in L$, $a \sqcap c \leq b$ iff $c \leq a \Rightarrow b$.

Suppose $a \Rightarrow b = 1$. Since $c \leq 1$ for all $c \in L$, we have $a \sqcap c \leq b$ for all $c \in L$. Particularly, $a \sqcap 1 = a \leq b$.

On the other hand, suppose $a \leq b$. Hence $a \sqcap 1 = a \leq b$. Thus, $1 \leq a \Rightarrow b$. Since $a \Rightarrow b \leq 1$, we have $a \Rightarrow b = 1$ as required. □

Lemma 4. *Let $\mathcal{B} = (B, \leq, \sqcup, \sqcap, -, 0, 1)$ be a Boolean algebra. Then $\mathcal{B}$ is relatively pseudo-complemented.*

Proof. Consider any $a, b \in B$. Define $a \Rightarrow b$ to be $-a \sqcup b$. We will show $c \leq a \Rightarrow b$ if and only if $a \sqcap c \leq b$ for all c.

Suppose $c \leq a \Rightarrow b = -a \sqcup b$. Then

$$a \sqcap c \leq a \sqcap (-a \sqcup b) = (a \sqcap -a) \sqcup (a \sqcap b) = a \sqcap b \leq b.$$

On the other hand, suppose $a \sqcap c \leq b$. Then

$$c = c \sqcap 1 = c \sqcap (-a \sqcup a) = (c \sqcap -a) \sqcup (c \sqcap a) \leq -a \sqcup (c \sqcap a) \leq -a \sqcup b.$$

□

By Lemma 4, a Boolean algebra is also a Heyting algebra.

We will consider both classical and intuitionistic propositional logics in this work. Given a set PV of *propositional variables*, the syntax of *propositional formulae* is defined as follows.

$$\varphi \quad = \quad PV \;\big|\; \bot \;\big|\; \varphi \vee \varphi \;\big|\; \varphi \wedge \varphi \;\big|\; \varphi \rightarrow \varphi$$

$$\text{Ax}\ \frac{}{P, \Gamma \vdash_K \Delta, P}\ P \text{ atomic} \qquad \text{L}\bot\ \frac{}{\bot, \Gamma \vdash_K}$$

$$\text{LW}\ \frac{\Gamma \vdash_K \Delta}{\varphi, \Gamma \vdash_K \Delta} \qquad \text{RW}\ \frac{\Gamma \vdash_K \Delta}{\Gamma \vdash_K \Delta, \varphi}$$

$$\text{L}\wedge\ \frac{\varphi, \varphi', \Gamma \vdash_K \Delta}{\varphi \wedge \varphi', \Gamma \vdash_K \Delta} \qquad \text{R}\wedge\ \frac{\Gamma \vdash_K \Delta, \varphi \qquad \Gamma \vdash_K \Delta, \varphi'}{\Gamma \vdash_K \Delta, \varphi \wedge \varphi'}$$

$$\text{L}\vee\ \frac{\varphi, \Gamma \vdash_K \Delta \qquad \varphi', \Gamma \vdash_K \Delta}{\varphi \vee \varphi', \Gamma \vdash_K \Delta} \qquad \text{R}\vee\ \frac{\Gamma \vdash_K \Delta, \varphi, \varphi'}{\Gamma \vdash_K \Delta, \varphi \vee \varphi'}$$

$$\text{L}\rightarrow\ \frac{\Gamma \vdash_K \Delta, \varphi \qquad \varphi', \Gamma \vdash_K \Delta}{\varphi \rightarrow \varphi', \Gamma \vdash_K \Delta} \qquad \text{R}\rightarrow\ \frac{\varphi, \Gamma \vdash_K \Delta, \varphi'}{\Gamma \vdash_K \Delta, \varphi \rightarrow \varphi'}$$

FIGURE 1. The System LK_0

We will use φ, φ', ψ to range over propositional formulae and abbreviate $\neg\varphi$ and $\varphi \leftrightarrow \varphi'$ for $\varphi \rightarrow \bot$ and $(\varphi \rightarrow \varphi') \wedge (\varphi' \rightarrow \varphi)$ respectively.

Let Γ and Δ be finite sets of propositional formulae. A *sequent* is of the form $\Gamma \vdash_\bullet \Delta$. For simplicity, we write $\varphi, \Gamma \vdash_\bullet \Delta, \varphi'$ for $\{\varphi\} \cup \Gamma \vdash_\bullet \Delta \cup \{\varphi'\}$. An *inference rule* in a proof system is denoted by

$$\ell\ \frac{\Gamma_0 \vdash_\bullet \Delta_0 \qquad \cdots \qquad \Gamma_n \vdash_\bullet \Delta_n}{\Gamma \vdash_\bullet \Delta}$$

where ℓ is the label of the rule, $\Gamma_0 \vdash_\bullet \Delta_0, \ldots, \Gamma_n \vdash_\bullet \Delta_n$ its *premises*, and $\Gamma \vdash_\bullet \Delta$ its *conclusion*. A *proof tree* is a tree-like structure constructed according to inference rules in a proof system. Proof systems offer a syntactic way to derive valid formulae. Gentzen defines the proof system LK for classical first-order logic. The system LK_0 for its propositional fragment is presented in Figure 1.[1] A proof tree in system LK_0 can be found in Figure 3 (a).

Let $\mathcal{B} = (B, \leq, \sqcup, \sqcap, -, 0, 1)$ be a Boolean algebra. Define a *valuation* ρ in $\mathcal{B}$ to be a mapping from PV to B. The valuation $[\![\varphi]\!]_K^\rho$ of a propositional formula φ is defined as follows.

$$\begin{array}{rcl@{\qquad}rcl}
[\![V]\!]_K^\rho & = & \rho(V) \text{ for } V \in PV & [\![\bot]\!]_K^\rho & = & 0 \\
[\![\varphi \vee \varphi']\!]_K^\rho & = & [\![\varphi]\!]_K^\rho \sqcup [\![\varphi']\!]_K^\rho & [\![\varphi \wedge \varphi']\!]_K^\rho & = & [\![\varphi]\!]_K^\rho \sqcap [\![\varphi']\!]_K^\rho \\
[\![\varphi \rightarrow \varphi']\!]_K^\rho & = & -[\![\varphi]\!]_K^\rho \sqcup [\![\varphi']\!]_K^\rho & & &
\end{array}$$

Given a Boolean algebra $\mathcal{B} = (B, \leq, \sqcup, \sqcap, -, 0, 1)$, a valuation ρ in $\mathcal{B}$, a propositional formula φ, and a set of propositional formulae Γ, we define $\mathcal{B}, \rho \models_K \varphi$ if $[\![\varphi]\!]_K^\rho = 1$ and $\mathcal{B}, \rho \models_K \Gamma$ if $\mathcal{B}, \rho \models_K \varphi$ for all $\varphi \in \Gamma$. Finally, $\Gamma \models_K \varphi$ if $\mathcal{B}, \rho \models_K \Gamma$ implies $\mathcal{B}, \rho \models_K \varphi$ for all $\mathcal{B}, \rho$.

The following theorem states that the system LK_0 is both sound and complete with respect to Boolean algebra.

Theorem 1. *Let Γ be a set of propositional formulae and φ a propositional formula. $\Gamma \vdash_K \varphi$ if and only if $\Gamma \models_K \varphi$.*

In contrast to classical logic, intuitionistic logic does not admit the law of excluded middle ($\varphi \vee \neg\varphi$). Philosophically, intuitionistic logic is closely related to constructivism. Its proof system, however, can be obtained from the system LK with a simple restriction: all sequents have exactly one formula at their right-hand

[1] Figure 1 is in fact a variant of the system LK, see [10].

$$\text{Ax}\ \frac{}{P, \Gamma \vdash_J P}\ P \text{ atomic} \qquad \text{L}\bot\ \frac{}{\bot, \Gamma \vdash_J \psi}$$

$$\text{LW}\ \frac{\Gamma \vdash_J \psi}{\varphi, \Gamma \vdash_J \psi}$$

$$\text{L}\wedge\ \frac{\varphi, \varphi', \Gamma \vdash_J \psi}{\varphi \wedge \varphi', \Gamma \vdash_J \psi} \qquad \text{R}\wedge\ \frac{\Gamma \vdash_J \psi \qquad \Gamma \vdash_J \psi'}{\Gamma \vdash_J \psi \wedge \psi'}$$

$$\text{L}\vee\ \frac{\varphi, \Gamma \vdash_J \psi \qquad \varphi', \Gamma \vdash_J \psi}{\varphi \vee \varphi', \Gamma \vdash_J \psi} \qquad \text{R}\vee\ \frac{\Gamma \vdash_J \psi_i}{\Gamma \vdash_J \psi_0 \vee \psi_1}\ (i = 0, 1)$$

$$\text{L}\to\ \frac{\Gamma \vdash_J \varphi \qquad \varphi', \Gamma \vdash_J \psi}{\varphi \to \varphi', \Gamma \vdash_J \psi} \qquad \text{R}\to\ \frac{\varphi, \Gamma \vdash_J \psi}{\Gamma \vdash_J \varphi \to \psi}$$

FIGURE 2. The System LJ_0

side. Figure 2 shows the propositional fragment of Gentzen's system LJ for intuitionistic logic. A sample proof tree in system LJ_0 is shown in Figure 3 (b).

Let $\mathcal{H} = (H, \leq, \sqcup, \sqcap, \Rightarrow, 0, 1)$ be a Heyting algebra. A *valuation* η in $\mathcal{H}$ is a mapping from PV to H. Similarly, we define the valuation $[\![\bullet]\!]^\eta_J$ over propositional formulae as follows.

$$\begin{array}{rcl} [\![V]\!]^\eta_J & = & \eta(V) \text{ for } V \in PV \\ [\![\varphi \vee \varphi']\!]^\eta_J & = & [\![\varphi]\!]^\eta_J \sqcup [\![\varphi']\!]^\eta_J \\ [\![\varphi \to \varphi']\!]^\eta_J & = & [\![\varphi]\!]^\eta_J \Rightarrow [\![\varphi']\!]^\eta_J \end{array} \qquad \begin{array}{rcl} [\![\bot]\!]^\eta_J & = & 0 \\ [\![\varphi \wedge \varphi']\!]^\eta_J & = & [\![\varphi]\!]^\eta_J \sqcap [\![\varphi']\!]^\eta_J \end{array}$$

Let $\mathcal{H} = (H, \leq, \sqcup, \sqcap, \Rightarrow, 0, 1)$ be a Heyting algebra, η a valuation, φ a propositional formula, and Γ a set of propositional formulae. The following satisfaction relations are defined similarly: $\mathcal{H}, \rho \models_J \varphi$ if $[\![\varphi]\!]_\rho = 1$, $\mathcal{H}, \rho \models_J \Gamma$ if $\mathcal{H}, \rho \models_J \varphi$ for all $\varphi \in \Gamma$, and $\Gamma \models_J \varphi$ if $\mathcal{H}, \rho \models_J \Gamma$ implies $\mathcal{H}, \rho \models_J \varphi$ for all $\mathcal{H}, \rho$. The system LJ_0 is both sound and complete with respect to Heyting algebra.

Theorem 2. *Let Γ be a set of propositional formulae and φ a propositional formula. $\Gamma \vdash_J \varphi$ if and only if $\Gamma \models_J \varphi$.*

Fix a set Σ of *alphabets*. A *string* is a finite sequence $a_1 a_2 \cdots a_n$ such that $a_i \in \Sigma$ for $1 \leq i \leq n$. The set of strings over Σ is denoted by Σ^*. Given a string $w = a_1 a_2 \cdots a_n$, its *length* (denoted by $|w|$) is n. The *empty string* ϵ is the string of length 0. Moreover, the *i-prefix* of $w = a_1 a_2 \cdots a_{|w|}$, denoted by $w \downarrow_i$, is the substring $a_1 a_2 \cdots a_i$. We define w_0 to be ϵ for any $w \in \Sigma^*$. A *language* over Σ is a subset of Σ^*. We say a language $L \subseteq \Sigma^*$ is *prefix-closed* if for any string $w \in L$, $w \downarrow_i \in L$ for all $0 \leq i \leq |w|$.

Definition 5. *A* finite state automaton M *is a tuple* $(Q, q_0, \longrightarrow, F)$ *where*

- Q *is a non-empty finite set of* states*;*
- $q_0 \in Q$ *is its* initial state*;*
- $\longrightarrow \subseteq Q \times \Sigma \times Q$ *is the total* transition relation*; and*
- $F \subseteq Q$ *is the* accepting states*.*

We say a finite state automaton is *deterministic* if $\longrightarrow$ is a total function from $Q \times \Sigma$ to Q. It is known that determinism does not change the expressiveness of finite automata [7]. For clarity, we write $q \xrightarrow{a} q'$ for $(q, a, q') \in \longrightarrow$. A *run* of a string $w = a_1 a_2 \cdots a_n$ in M is a finite alternating sequence $q_0 a_1 q_1 a_2 \cdots q_{n-1} a_n q_n$ such that $q_i \xrightarrow{a_{i+1}} q_{i+1}$ for $0 \leq i < n$; it is *accepting* if $q_n \in F$. We say a string w is *accepted* by M if there is an accepting run of w in M. The set of all strings accepted by M is called the *language accepted by* M and denoted by $L(M)$. A language $L \subseteq \Sigma^*$ is *regular* if there is a finite state automaton M such that $L = L(M)$.

Let $L \subseteq \Sigma^*$ be regular. Define $\overline{L} = \Sigma^* \setminus L$. It is known that regular languages are closed under Boolean operations [7].

Theorem 3. *Let L and L' be regular. Then $L \cup L'$, $L \cap L'$, and $\overline{L}$ are regular.*

3. Classical Interpretation

It is trivial to see that regular languages form a Boolean algebra. More formally, consider the set $R = \{L \subseteq \Sigma^* : L \text{ is regular }\}$. The following theorem follows directly from the closure property of regular sets.

Theorem 4. *Let $\mathcal{R} = (R, \subseteq, \cup, \cap, \overline{\bullet}, \emptyset, \Sigma^*)$. $\mathcal{R}$ is a Boolean algebra.*

Proof. Let $L, L' \in R$. Then $L \cup L'$, $L \cap L'$, and $\overline{L}$ are in R by Theorem 3. Both $\emptyset$ and Σ^* are trivially regular. Furthermore, we have $L \cup \overline{L} = \Sigma^*$ and $L \cap \overline{L} = \emptyset$. Finally, $\emptyset \subseteq L$ and $L \subseteq \Sigma^*$ for any $L \in R$. □

Theorem 1 immediately gives the following corollary.

Corollary 1. *Let ρ be a valuation in $\mathcal{R}$ and φ a propositional formula. If $\Gamma \vdash_K \varphi$, then $\mathcal{R}, \rho \models_K \Gamma$ implies $\mathcal{R}, \rho \models_K \varphi$.*

To illustrate the significance of Corollary 1, let us consider the following scenario. Suppose we are given five regular languages M_0, M_1, A_0, A_1, and P. Further, assume $M_0 \cap A_0 \subseteq P$, $M_1 \cap A_1 \subseteq P$, and $\overline{A}_0 \cap \overline{A}_1 \subseteq P$. We can deduce $M_0 \cap M_1 \subseteq P$ as follows. First, consider the valuation ρ that assigns propositional variables to regular languages of the same name. Suppose there is a proof tree for the following sequent.

$$M_0 \wedge A_0 \to P, M_1 \wedge A_1 \to P, \neg A_0 \wedge \neg A_1 \to P \vdash_K M_0 \wedge M_1 \to P.$$

Corollary 1 asserts that if $\mathcal{R}, \rho \models_K M_0 \wedge A_0 \to P$, $\mathcal{R}, \rho \models_K M_1 \wedge A_1 \to P$, and $\mathcal{R}, \rho \models_K \neg A_0 \wedge \neg A_1 \to P$, then $\mathcal{R}, \rho \models_K M_0 \wedge M_1 \to P$. Lemma 3 gives exactly what we want in $\mathcal{R}$. Hence the proof tree in Figure 3 (a) suffices to prove $M_0 \cap M_1 \subseteq P$. Note that we do not make semantic arguments in the analysis. Instead, Corollary 1 allows us to derive semantic property ($M_0 \cap M_1 \subseteq P$) by manipulating sequents syntactically.

3.1. Limitation of Classical Interpretation.

Consider the following circular inference rule proposed in [1].

$$\frac{}{\vdash [(E \to M) \wedge (M \to E)] \to M}$$

It is easy to see that the valuation of the formula is 0 by taking $E = M = 0$ in any Boolean algebra. Since the system LK_0 is sound for Boolean algebra, we conclude that the rule cannot be derived by the proof system. But it does not imply that the rule is not sound in other algebraic semantics. To give insights to the intuitionistic interpretation, it is instructive to see how the rule fails in non-trivial cases.

Consider the automata in Figure 4. Let M and E denote the automaton shown in Figure 4 (a) and (b) respectively. Let the valuation ρ be that $\rho(E) = L(E)$ and $\rho(M) = L(M)$. Observe that the string $bd \notin L(M)$. Hence $bd \in \rho(M \to E) = \overline{L(M)} \cup L(E)$. Similarly, $bd \in \rho(E \to M)$. But $bd \notin L(M)$. Thus $\rho(M \to E) \cap \rho(E \to M) \not\subseteq \rho(M)$. Hence $\nvdash [(E \to M) \wedge (M \to E)] \to M$ by Lemma 3.

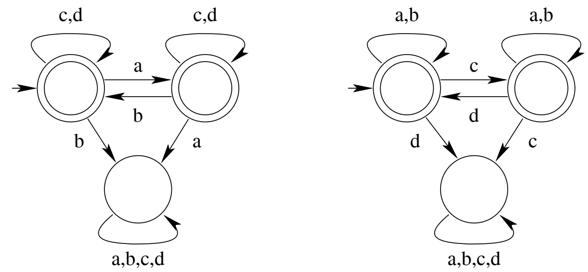
$$
\begin{prooftree}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$M_0, A_0 \vdash_K M_0$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$M_0, A_0 \vdash_K A_0$}
\LeftLabel{R$\wedge$}
\BinaryInfC{$M_0, A_0 \vdash_K M_0 \wedge A_0$}
\LeftLabel{RW}
\UnaryInfC{$M_0, A_0 \vdash_K P, M_0 \wedge A_0$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$M_0, A_0, P \vdash_K P$}
\LeftLabel{L$\rightarrow$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, M_0, A_0 \vdash_K P$}
\LeftLabel{LW}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_0, M_1, A_0 \vdash_K P$}
\LeftLabel{LW}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, M_0, M_1, A_0 \vdash_K P$}
\LeftLabel{RW}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, M_0, M_1, A_0 \vdash_K \bot, P$}
\LeftLabel{R$\rightarrow$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, M_0, M_1 \vdash_K \neg A_0, P$}
\AxiomC{$\Pi_{3,0}$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, M_0, M_1 \vdash_K \neg A_1, P$}
\LeftLabel{R$\wedge$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, M_0, M_1 \vdash_K \neg A_0 \wedge \neg A_1, P$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, P, M_0, M_1 \vdash_K P$}
\LeftLabel{L$\rightarrow$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, \neg A_0 \wedge \neg A_1 \rightarrow P, M_0, M_1 \vdash_K P$}
\LeftLabel{L$\wedge$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, \neg A_0 \wedge \neg A_1 \rightarrow P, M_0 \wedge M_1 \vdash_K P$}
\LeftLabel{R$\rightarrow$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, \neg A_0 \wedge \neg A_1 \rightarrow P \vdash_K M_0 \wedge M_1 \rightarrow P$}
\end{prooftree}
$$

(a) A Proof Tree in LK_0

$$
\begin{prooftree}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$A_0, M_0 \vdash M_0$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$A_0, M_0 \vdash A_0$}
\LeftLabel{R$\wedge$}
\BinaryInfC{$A_0, M_0 \vdash M_0 \wedge A_0$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$P, A_0, M_0 \vdash P$}
\LeftLabel{L$\rightarrow$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, A_0, M_0 \vdash P$}
\LeftLabel{LW}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, A_0, M_0, M_1 \vdash P$}
\LeftLabel{LW}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_0, M_0, M_1 \vdash P$}
\AxiomC{$\Pi_{3,1}$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_1, M_0, M_1 \vdash P$}
\LeftLabel{L$\vee$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_0 \vee A_1, M_0, M_1 \vdash P$}
\AxiomC{}
\LeftLabel{Ax}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, P, M_0, M_1 \vdash P$}
\LeftLabel{L$\vee$}
\BinaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_0 \vee A_1 \vee P, M_0, M_1 \vdash P$}
\LeftLabel{L$\wedge$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_0 \vee A_1 \vee P, M_0 \wedge M_1 \vdash P$}
\LeftLabel{R$\rightarrow$}
\UnaryInfC{$M_0 \wedge A_0 \rightarrow P, M_1 \wedge A_1 \rightarrow P, A_0 \vee A_1 \vee P \vdash M_0 \wedge M_1 \rightarrow P$}
\end{prooftree}
$$

(b) A Proof Tree in LJ_0

FIGURE 3. Sample Proof Trees of (a) LK_0 and (b) LJ_0



(a) The automaton M (b) The automaton E

Figure 4. Limitation of Classical Interpretation

Note that both $L(M)$ and $L(E)$ are non-empty and prefix-closed. The argument is still valid should we restrict to non-empty, prefix-closed regular languages.

In classical interpretation, the valuation of $E \rightarrow M$ is defined as $\overline{\rho(E)} \cup \rho(M)$. Any string not in $\rho(E)$ belongs to $\rho(E \rightarrow M)$, even though it may not belong to $\rho(M)$. The problem arises exactly when $\overline{\rho(E)} \cap \overline{\rho(M)}$ is not empty. One may suspect that the valuation of $E \rightarrow M$ is defined too liberally in classical interpretation and resort to a more conservative version. This is indeed the approach taken by Abadi and Plotkin and followed in this work.

4. Interpretation à la Abadi and Plotkin

In order to admit circular compositional rules, an interpretation inspired by [1] is proposed here. Mimicking the definition of relative pseudo-complements in [1], we show that non-empty, prefix-closed regular languages form a Heyting algebra. Hence the system LJ_0 can be used to derive compositional rules in the new interpretation. Moreover, we will prove the soundness of the circular inference rule in Section 3.1. It allows us to derive other circular compositional rules.

We use the regular set $\{\epsilon\}$ as the zero element. The following lemma shows that our choice is indeed justified.

Lemma 5. *Let L be a non-empty, prefix-closed language. Then $\epsilon \in L$ if and only if $L \neq \emptyset$.*

Proof. If $\epsilon \in L$, $L \neq \emptyset$ trivially. Now suppose $w \in L$. Since L is prefix-closed, $w\downarrow_n \in L$ for $0 \leq n \leq |w|$. Particularly, $w\downarrow_0 = \epsilon \in L$. $\square$

Similarly, we would like to know whether two non-empty, prefix-closed regular languages are closed under intersection and union. This is shown in the following lemma.

Lemma 6. *Let K and L be non-empty, prefix-closed regular languages. Then $K \cap L$ and $K \cup L$ are non-empty, prefix-closed regular languages.*

Proof. Since regular languages are closed under Boolean operators. $K \cap L$ and $K \cup L$ are regular. Further, $\epsilon \in K$ and $\epsilon \in L$ by Lemma 5. Hence both $K \cap L$ and $K \cup L$ are non-empty. It remains to show both are prefix-closed.

Consider any string $w \in K \cap L$. Hence $w \in K$. Thus all prefixes of w are in K since K is prefix-closed. Similarly, all prefixes of w are in L. Thus all prefixes of w are in $K \cap L$.

The case for $K \cup L$ is proved similarly. □

For relative pseudo-complements, we follow the definition in [1]. Note that the following definition does not allude to closure properties. In order to define a Heyting algebra, one must show that non-empty, prefix-closed regular languages are closed under relative pseudo-complementation.

Definition 6. *Let K and L be prefix-closed languages. Define*

$$K \to L = \{w : w\downarrow_n \in K \to w\downarrow_n \in L \textit{ for } 0 \leq n \leq |w|\}.$$

We first show that the language $K \to L$ defined above is indeed the pseudo-complement of K relative to L.

Proposition 1. *Let K, L, and M be prefix-closed languages. Then $K \cap M \subseteq L$ if and only if $M \subseteq K \to L$.*

Proof. Assume $K \cap M \subseteq L$ and $w \in M$, but $w \notin K \to L$. Then there is an n such that $w\downarrow_n \in K$ but $w\downarrow_n \notin L$. Since $w \in M$ and M is prefix-closed, $w\downarrow_n \in M$. Hence $w\downarrow_n \in K \cap M$. Thus $w\downarrow_n \in L$, a contradiction.

On the other hand, assume $M \subseteq K \to L$ and $w \in K \cap M$. Since $w \in M$ and $M \subseteq K \to L$, $w \in K \to L$. Particularly, $w \in K$ implies $w \in L$. But $w \in K$ by assumption. Hence $w \in L$ as required. □

Next, we show that $K \to L$ is non-empty and prefix-closed if both K and L are non-empty and prefix-closed.

Lemma 7. *Let K and L be non-empty, prefix-closed languages. Then $K \to L$ is non-empty and prefix-closed.*

Proof. Since both K and L are non-empty and prefix-closed, $\epsilon \in K$ and $\epsilon \in L$. Hence $\epsilon \in K \to L$ by definition.

Now suppose $w \in K \to L$. We have $w\downarrow_n \in K \to w\downarrow_n \in L$ for $0 \leq n \leq |w|$. Consider any prefix $w\downarrow_i$, we have $w\downarrow_j \in K \to w\downarrow_j \in L$ for $0 \leq j \leq i$. That is, $w\downarrow_i \in K \to L$. □

It remains to show that regularity is preserved by Definition 6. Given two deterministic finite state automata M and N, we construct a new automaton $M \to N$ such that $L(M \to N) = L(M) \to L(N)$. Our idea is to use an extra bit to accumulate information while simulating both automata. Let $\mathbb{B} = \{\ \mathsf{false}, \mathsf{true}\ \}$ be the Boolean domain. The following definition gives the construction of $M \to N$.

Definition 7. *Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be deterministic finite state automata accepting prefix-closed languages. Define the finite state automaton $M \to N = (P \times Q \times \mathbb{B}, (p_0, q_0, b_0), \longrightarrow, F)$ as follows.*

- $b_0 = \begin{cases} \mathsf{true} & \textit{if } p_0 \in F_M \to q_0 \in F_N \\ \mathsf{false} & \textit{otherwise} \end{cases}$
- $(p, q, b) \xrightarrow{a} (p', q', b')$ *if*
  - $p \xrightarrow{a}_M p'$;
  - $q \xrightarrow{a}_N q'$; *and*
  - $b' = \begin{cases} \mathsf{true} & \textit{if } b = \mathsf{true} \textit{ and } p' \in F_M \to q' \in F_N \\ \mathsf{false} & \textit{otherwise} \end{cases}$
- $F = \{(p, q, \mathsf{true}) : p \in P, q \in Q\}$.

The automaton $M \to E$ of the automata M and E in Figure 4 is shown in Figure 5. Note that the bold states are the products of the unaccepting states in

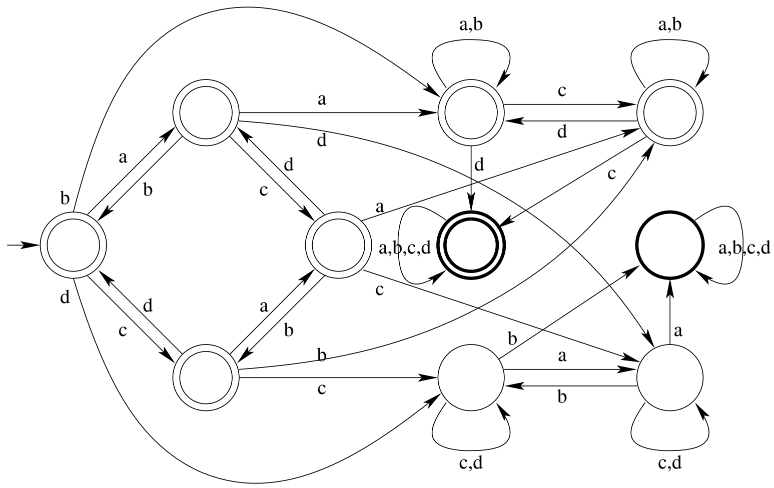


FIGURE 5. The automaton $M \to E$

Figure 4. Any string prefixed by strings in $\overline{L(M)}$ and followed by those in $\overline{L(E)}$ is accepted in the bold accepting state in $M \to E$. On the other hand, strings prefixed by $\overline{L(E)}$ and followed by $\overline{L(M)}$ reach the other bold state and are not accepted.

To show that $L(M \to N) = L(M) \to L(N)$, we use the following lemma.

Lemma 8. *Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be deterministic finite state automata accepting non-empty, prefix-closed languages. Consider any $w \in \Sigma^*$ and $(p_0, q_0, b_0) \xrightarrow{w\downarrow_n} (p_n, q_n, b_n)$ in $M \to N$ for $0 \leq n \leq |w|$. Then $b_{|w|}$ is true if and only if $p_n \in F_M \to q_n \in F_N$ for $0 \leq n \leq |w|$.*

Proof. ($\Rightarrow$) We prove $p_n \in F_M \to q_n \in F_N$ and $b_n = \mathsf{true}$ by induction on $|w| - n$. The basis, $n = |w|$, follows by the assumption $b_{|w|}$ is true and the definition of $b_{|w|}$. Now suppose $p_n \in F_M \to q_n \in F_N$ and $b_n = \mathsf{true}$. Observe that $b_{n-1} = \mathsf{true}$ for $b_n = \mathsf{true}$. Hence $p_{n-1} \in F_M \to q_{n-1} \in F_N$ follows by $b_{n-1} = \mathsf{true}$.

($\Leftarrow$) We prove $b_n = \mathsf{true}$ by induction on n. Since $p_0 \in F_M \to q_0 \in F_N$, $b_0 = \mathsf{true}$ by definition. Suppose $b_n = \mathsf{true}$. Then $b_{n+1} = \mathsf{true}$ for $b_n = \mathsf{true}$ and $p_{n+1} \in F_M \to q_{n+1} \in F_N$. □

Now we establish $L(M \to N) = L(M) \to L(N)$ in the following proposition.

Proposition 2. *Let M and N be deterministic finite state automata accepting non-empty, prefix-closed languages. Then $L(M) \to L(N) = L(M \to N)$.*

Proof. Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be two deterministic finite state automata accepting non-empty, prefix-closed regular languages. Consider any $w \in \Sigma^*$. Define $(p_0, q_0, b_0) \xrightarrow{w\downarrow_n} (p_n, q_n, b_n)$ as in Lemma 8.

Suppose $w \in L(M \to N)$. It suffices to show that $w\downarrow_n \in L(M) \to w\downarrow_n \in L(N)$ for $0 \leq n \leq |w|$. From Lemma 8, $b_n = \mathsf{true}$ for $0 \leq n \leq |w|$. By the definition of $M \to N$, we have

- $p_0 \xrightarrow{w\downarrow_n}_M p_n$;
- $q_0 \xrightarrow{w\downarrow_n}_N q_n$; and
- $p_n \in F_M \to q_n \in F_N$.

That is, $w\downarrow_n \in L(M) \to w\downarrow_n \in L(N)$ for $0 \le n \le |w|$. Hence $w \in L(M) \to L(N)$.

On the other hand, suppose $w \in L(M) \to L(N)$. Thus, $w\downarrow_n \in L(M) \to w\downarrow_n \in L(N)$ for $0 \le n \le |w|$. Consider $(p_0, q_0, b_0) \xrightarrow{w\downarrow_n} (p_n, q_n, b_n)$ in $M \to N$. We show $b_{|w|} = \mathsf{true}$ by induction on n. When $n = 0$, $w\downarrow_n = \epsilon \in L(M) \to L(N)$. Hence $\epsilon \in L(M) \to \epsilon \in L(N)$. Therefore $p_0 \in F_M \to q_0 \in F_N$. That is, $b_0 = \mathsf{true}$.

Suppose $b_n = \mathsf{true}$. Consider $w\downarrow_{n+1}$. We have $w\downarrow_{n+1} \in L(M) \to w\downarrow_{n+1} \in L(N)$. Hence $p_{n+1} \in F_M \to q_{n+1} \in F_N$. Therefore $b_{n+1} = \mathsf{true}$ as required. ☐

Since any regular language is accepted by a deterministic finite state automaton, the following proposition is only a simple application of Proposition 2.

Proposition 3. *Let K and L be non-empty, prefix-closed regular languages. Then $K \to L$ is a non-empty, prefix-closed regular language.*

Proof. Let $M = (P, \Sigma, p_0, \longrightarrow_M, F_M)$ and $N = (Q, \Sigma, q_0, \longrightarrow_N, F_N)$ be deterministic finite automata such that $L(M) = K$ and $L(N) = L$ respectively. By Proposition 2, $K \to L = L(M \to N)$. Hence $K \to L$ is regular. By Lemma 7, $K \to L$ is both non-empty and prefix-closed as required. ☐

After establishing closure properties of various operations, we can define our new interpretation. More formally, define $R^+ = \{L \subseteq \Sigma^* : L$ is non-empty, prefix-closed, and regular $\}$. The following theorem states that non-empty, prefix-closed regular languages form a Heyting algebra.

Theorem 5. *Let $\mathcal{R}^+ = (R^+, \subseteq, \cup, \cap, \to, \{\epsilon\}, \Sigma^*)$. $\mathcal{R}^+$ is a Heyting algebra.*

Proof. Lemma 6 and Proposition 3 show that R^+ is closed under $\cup$, $\cap$, and $\to$. Lemma 5 shows that $\{\epsilon\} \subseteq L$ for any non-empty, prefix-closed language L. Finally, Proposition 1 shows $K \to L$ is the pseudo-complement of K relative to L. ☐

Similar to our classical interpretation, Theorem 2 immediately gives the following corollary.

Corollary 2. *Let η be a valuation in $\mathcal{R}^+$ and φ a propositional formula. If $\Gamma \vdash_J \varphi$, then $\mathcal{R}^+, \eta \models_J \Gamma$ implies $\mathcal{R}^+, \eta \models_J \varphi$.*

Let us consider an application of Corollary 2. Suppose five non-empty prefix-closed regular languages M_0, M_1, A_0, A_1, and P are given. Assume $M_0 \cap A_0 \subseteq P$, $M_1 \cap A_1 \subseteq P$, and $A_0 \cup A_1 \cup P = \Sigma^*$. We can deduce $M_0 \cap M_1 \subseteq P$ by the proof tree in Figure 3 (b).

We now turn our attention to circular compositional rules. A modified version of non-interference in [1] is used in our setting.

Definition 8. *Let L be a non-empty, prefix-closed language in Σ^* and $\Xi \subseteq \Sigma$. We say L* constrains *Ξ, write $L \triangleright \Xi$, if for any $w \in L$, $wa \in L$ for any $a \notin \Xi$.*

Exploiting non-interference, we show the circular inference rule presented in Section 3.1 is sound by induction on the length of strings.

Theorem 6. *Let K and L be non-empty, prefix-closed languages such that $K \triangleright \Xi_K$, $L \triangleright \Xi_L$, and $\Xi_K \cap \Xi_L = \emptyset$. Then $(K \to L) \cap (L \to K) \subseteq K$.*

Proof. Consider any $w \in (K \to L) \cap (L \to K)$. We will show $w\downarrow_n \in K$ for all n. Firstly, $w\downarrow_0 = \epsilon \in K$ for K is non-empty. Now suppose $w\downarrow_{i+1} = w\downarrow_i a_i$ and $w\downarrow_i \in K$. Consider the following two cases.

(1) $a_i \notin \Xi_K$. Then $w\downarrow_{i+1} \in K$ for $K \triangleright \Xi_K$.

(2) $a_i \in \Xi_K$. Since $w\downarrow_i \in K$ and $w\downarrow_i \in K \to L$, $w\downarrow_i \in L$. Hence $w\downarrow_{i+1} \in L$ for $a_i \in \Xi_K$ and $\Xi_K \cap \Xi_L = \emptyset$. Finally, $w\downarrow_{i+1} \in K$ follows from $w\downarrow_{i+1} \in L \to K$. □

Theorem 6 admits an inference rule in the Heyting algebra $\mathcal{R}^+$. More formally, we have the following corollary.

Corollary 3. *Let P and P' be propositional variables, Ξ and Ξ' subsets of Σ, η a valuation in $\mathcal{R}^+$. Then we have the following inference rule with respect to $\mathcal{R}^+$, provided $\rho(P) \triangleright \Xi$, $\rho(P') \triangleright \Xi'$, $\Xi \cap \Xi' = \emptyset$.*

$$CP \frac{}{\vdash [(P \to P') \wedge (P' \to P)] \to P}$$

We can in fact characterize the non-empty prefix-closed language satisfying the condition in Theorem 6. Note that one direction can be obtained by syntactic deduction. It is the other direction where semantic analysis is needed.

Theorem 7. *Let K and L be non-empty, prefix-closed languages such that $K \triangleright \Xi_K$, $L \triangleright \Xi_L$, and $\Xi_K \cap \Xi_L = \emptyset$. Then $(K \to L) \cap (L \to K) = K \cap L$.*

Proof. By Theorem 6, $(K \to L) \cap (L \to K) \subseteq K$. Symmetrically, we have $(K \to L) \cap (L \to K) \subseteq L$.

Conversely, consider any $w \in K \cap L$. Then $w\downarrow_n \in K$ and $w\downarrow_n \in L$ for $0 \leq n \leq |w|$. Hence $w\downarrow_n \in K \to w\downarrow_n \in L$ for $0 \leq n \leq |w|$. That is, $w \in K \to L$. Similarly, $w \in L \to K$. Therefore $K \cap L \subseteq (K \to L) \cap (L \to K)$. □

We can similarly summarize Theorem 7 as follows.

Corollary 4. *Let P and P' be propositional variables, Ξ and Ξ' subsets of Σ, η a valuation in $\mathcal{R}^+$. Then we have the following inference rule with respect to $\mathcal{R}^+$, provided $\rho(P) \triangleright \Xi$, $\rho(P') \triangleright \Xi'$, $\Xi \cap \Xi' = \emptyset$.*

$$CP' \frac{}{\vdash [(P \to P') \wedge (P' \to P)] \leftrightarrow (P \wedge P')}$$

Consider again the examples in Figure 4. Observe that both $L(M)$ and $L(E)$ are non-empty and prefix-closed. Let the valuation η in $\mathcal{R}^+$ be that $\eta(M) = L(M)$ and $\eta(E) = L(E)$. Since $b \in \eta(E)$ but $b \notin \eta(M)$, $b \notin \eta(E \to M)$. Thus $bd \notin \eta(E \to M)$. Also note that $\eta(M) \triangleright \{a, b\}$ and $\eta(E) \triangleright \{c, d\}$. Hence $\eta(M \to E) \cap \eta(E \to M) = \eta(M) \cap \eta(E)$ by Theorem 7.

5. Applications

A subclass of finite state automata called labeled transition systems (LTS) is used for system modeling in automated compositional reasoning [3, 4]. Intuitively, an LTS models observable actions by its alphabets. When shared observable actions are performed in compositions of two LTS's, they are synchronized. On the other hand, an LTS stutters when the other LTS performs its local observable actions in their composition. In this section, we apply proof-theoretic techniques to derive soundness of compositional rules for LTS's. It is noted that our formulation is equivalent to those in [4, 3].

Definition 9. *Let $\Sigma_M \subseteq \Sigma$. A deterministic finite state automaton $M = (Q, q_0, \longrightarrow, F)$ is a* labeled transition system (LTS) over Σ_M *if the following conditions are satisfied.*

(1) *$q \xrightarrow{a} q$ for any $q \in Q$ and $a \in \Sigma \setminus \Sigma_M$; and*

(2) *for any* $n \geq 0$, *if* $q_i \xrightarrow{a_{i+1}} q_{i+1}$ *for* $0 \leq i < n$ *and* $q_n \in F$, *then* $q_i \in F$ *for* $0 \leq i \leq n$.

With our formulation, it is possible to define compositions of two LTS's by product automata. Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be two LTS's over Σ_M and Σ_N respectively. Define the finite state automaton $M \| N = (P \times Q, (p_0, q_0), \longrightarrow, F)$ as follows.

- $(p, q) \xrightarrow{a} (p', q')$ if $p \xrightarrow{a}_M p'$ and $q \xrightarrow{a}_N q'$; and
- $F = F_M \times F_N$.

The following lemma shows that product automata are indeed LTS's.

Lemma 9. *Let* $M = (P, p_0, \longrightarrow_M, F_M)$ *and* $N = (Q, q_0, \longrightarrow_N, F_N)$ *be two LTS's over* Σ_M *and* Σ_N *respectively. Then* $M \| N$ *is an LTS over* $\Sigma_M \cup \Sigma_N$.

Proof. If $(p, q) \xrightarrow{a} (p', q')$ and $(p, q) \xrightarrow{a} (p'', q'')$, then $p \xrightarrow{a}_M p'$ and $p \xrightarrow{a}_M p''$. Thus $p' = p''$ for the LTS M is a deterministic finite state automaton. Similarly, $q' = q''$. Hence $(p', q') = (p'', q'')$, $M \| N$ is deterministic.

Suppose $a \notin \Sigma_M \cup \Sigma_N$. We have $p \xrightarrow{a}_M p$ and $q \xrightarrow{a}_N q$ for M and N are LTS's. Hence $(p, q) \xrightarrow{a} (p, q)$ as required.

For any n, consider $(p_i, q_i) \xrightarrow{a_{i+1}} (p_{i+1}, q_{i+2})$ for $0 \leq i < n$ in $M \| N$. If $(p_n, q_n) \in F_M \times F_N$, $p_n \in F_M$ and $q_n \in F_N$. But then $p_i \in F_M$ and $q_i \in F_N$ and hence $(p_i, q_i) \in F_N$ for $0 \leq i \leq n$. □

We now recall that the language accepted by the product of two automata is nothing more than the intersection of the languages accepted by these automata.

Proposition 4. *Let* $M = (P, p_0, \longrightarrow_M, F_M)$ *and* $N = (Q, q_0, \longrightarrow_N, F_N)$ *be two LTS's over* Σ_M *and* Σ_N *respectively. Then* $L(M \| N) = L(M) \cap L(N)$.

Proof. Let $w = a_1 \cdots a_n \in L(M_0 \| M_1)$. Then there is a run

$$(p_0, q_0) \xrightarrow{a_1} (p_1, q_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (p_n, q_n).$$

By the definition of the transition relation of $M \| N$, we have two runs

$$p_0 \xrightarrow{a_1}_M p_1 \xrightarrow{a_2}_M \cdots \xrightarrow{a_n}_M p_n \text{ in } M$$

and

$$q_0 \xrightarrow{a_1}_N q_1 \xrightarrow{a_2}_N \cdots \xrightarrow{a_n}_N q_n \text{ in } N$$

respectively. Hence $w \in L(M)$ and $w \in L(N)$.

The other direction is similar. □

Symmetrically, define the finite state automaton $M + N = (P \times Q, (p_0, q_0), \longrightarrow, F)$ as follows.

- $(p, q) \xrightarrow{a} (p', q')$ if $p \xrightarrow{a}_M p'$ and $q \xrightarrow{a}_N q'$; and
- $F = (F_M \times Q) \cup (P \times F_N)$.

Lemma 10. *Let* $M = (P, p_0, \longrightarrow_M, F_M)$ *and* $N = (Q, q_0, \longrightarrow_N, F_N)$ *be two LTS's over* Σ_M *and* Σ_N *respectively. Then* $M + N$ *is an LTS over* $\Sigma_M \cup \Sigma_N$.

Proof. $M + N$ is deterministic for M and N are both deterministic.

If $a \notin \Sigma_M \cup \Sigma_N$, $(p, q) \xrightarrow{a} (p, q)$ for $p \xrightarrow{a}_M p$ and $q \xrightarrow{a}_N q$.

For any n, let $(p_i, q_i) \xrightarrow{a_{i+1}} (p_{i+1}, q_{i+1})$ for $0 \leq i < n$ in $M + N$. If $(p_n, q_n) \in F$, let $p_n \in F_M$ without loss of generality. Since M is an LTS, $p_i \in F_M$ for $0 \leq i \leq n$. That is, $(p_i, q_i) \in F$ for $0 \leq i \leq n$. □

It is as easy to show that $M + N$ accepts the union of the languages accepted by M and N.

Proposition 5. *Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be two LTS's over Σ_M and Σ_N respectively. Then $L(M + N) = L(M) \cup L(N)$.*

Proof. Consider an accepting run for $w = a_1 \cdots a_n \in L(M + N)$ as follows.

$$(p_0, q_0) \xrightarrow{a_1} (p_1, q_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (p_n, q_n).$$

Suppose $p_n \in F_M$ without loss of generality. The following run is accepting in M.

$$p_0 \xrightarrow{a_1}_M p_1 \xrightarrow{a_2}_M \cdots \xrightarrow{a_n}_M p_n$$

Hence $w \in L(M)$.

The other direction is similar. □

The automaton $M \to N$ is also an LTS if M and N both are. The alphabets constrained by $M \to N$ is also characterized in the following lemma.

Lemma 11. *Let $M = (P, p_0, \longrightarrow_M, F_M)$ and $N = (Q, q_0, \longrightarrow_N, F_N)$ be two LTS's over Σ_M and Σ_N respectively. Then $M \to N$ is an LTS over $\Sigma_M \cup \Sigma_N$.*

Proof. Let $M \to N = (P \times Q \times \mathbb{B}, (p_0, q_0, b_0), \longrightarrow, F)$.

Suppose $(p, q, b) \xrightarrow{a} (p', q', b')$ and $(p, q, b) \xrightarrow{a} (p'', q'', b'')$ in $M \to N$. Since M and N are deterministic, $p' = p''$ and $q' = q''$. But then $b'' = p'' \in F_M \to q'' \in F_N = p' \in F_M \to q' \in F_N = b'$. $M \to N$ is deterministic.

For any (p, q, b) and $a \notin \Sigma_M \cup \Sigma_N$, we have $p \xrightarrow{a}_M p$ and $q \xrightarrow{a}_N q$ for M and N are LTS's. Hence $(p, q, b) \xrightarrow{a} (p, q, b)$.

Finally, suppose $(p_i, q_i, b_i) \xrightarrow{a_{i+1}} (p_{i+1}, q_{i+1}, b_{i+1})$ for $0 \leq i < n$ and $b_n = \mathsf{true}$. Define $w = a_1 \cdots a_n$. We have $b_i = \mathsf{true}$ for $0 \leq i \leq n$ by Lemma 8. Hence $(p_i, q_i, b_i) \in F$ for $0 \leq i \leq n$. □

Suppose two LTS's M and P are given as specifications of the system and the property respectively. We would like to know whether all observable action sequences of the system M conform to the property P, namely, $L(M) \subseteq L(P)$. If so, we say M *satisfies* P and denote it by $M \models P$.

Example 1. *Consider the following assume-guarantee rule where M_0, M_1, A_0, A_1, P are LTS's, and $\overline{M}$ denotes the complement automaton of M. Note that $\overline{M}$ is not necessary an LTS* [3].

$$\frac{M_0 \| A_0 \models P \qquad M_1 \| A_1 \models P \qquad \overline{A}_0 \| \overline{A}_1 \models P}{M_0 \| M_1 \models P}$$

By Lemma 3, Proposition 4, and Theorem 1, we can establish the soundness of the rule by finding a proof tree of the following sequent.

$$M_0 \wedge A_0 \to P, M_1 \wedge A_1 \to P, \neg A_0 \wedge \neg A_1 \to P \vdash_K M_0 \wedge M_1 \to P$$

The proof tree is shown in Figure 3 (a). □

Using existing proof assistants, we can in fact prove all rules in [3] and [4] automatically.

Example 2. *Suppose $M_0, M_1, A_0, A_1, P_0, P_1$ are LTS's. Consider the following assume-guarantee rule in* [3].

$$\frac{M_0\|A_0 \models P_0 \quad M_1\|A_1 \models P_1 \quad M_0\|A_0 \models A_1 \quad M_1\|A_1 \models A_0 \quad L(\overline{A}_0\|\overline{A}_1) = \emptyset}{M_0\|M_1 \models P_0\|P_1}$$

It suffices to find a proof for the following sequent.

$$\begin{array}{l} M_0 \wedge A_0 \to P_0, \quad M_1 \wedge A_1 \to P_1, \\ M_0 \wedge A_0 \to A_1, \quad M_1 \wedge A_1 \to A_0, \;\; \vdash_K M_0 \wedge M_1 \to P_0 \wedge P_1 \\ \qquad\quad \neg A_0 \wedge \neg A_1 \leftrightarrow \mathsf{false} \end{array}$$

We can use the proof assistant Isabelle to search the proof tree. The following transcript shows that the tactic auto () is able to find a proof automatically.[2]

```
> Goal "ALL M0 M1 A0 A1 P0 P1 . (M0 & A0 → P0) & (M1 & A1 → P1) &
(M0 & A0 → A1) & (M1 & A1 → A0) & (( ¬ A0 & ¬ A1) = False) ⇒
(M0 & M1 → P0 & P1)";
Level 0 (1 subgoal)
...
> auto ();
Level 1
ALL M0 M1 A0 A1 P0 P1.
   (M0 & A0 → P0) &
   (M1 & A1 → P1) &
   (M0 & A0 → A1) & (M1 & A1 → A0) & ( ¬ A0 & ¬ A1) = False
⇒M0 & M1 → P0 & P1
No subgoals!
val it = () : unit
```
□

To establish circular compositional rules in our framework, the intuitionistic interpretation in Section 4 is needed. The following lemma characterizes the languages accepted by LTS's and the alphabets constrained by them.

Lemma 12. *Let $M = (Q, q_0, \longrightarrow, F)$ be an LTS over Σ_M. Then $L(M)$ is non-empty, prefix-closed, and $L(M) \triangleright \Sigma_M$.*

Proof. The condition (2) in Definition 9 implies $q_0 \in F$ by taking $n = 0$. Hence $\epsilon \in L(M)$, $L(M)$ is not empty.

Consider any string $w = a_1 a_2 \cdots a_n \in L(M)$. We have $q_i \xrightarrow{a_{i+1}} q_{i+1}$ for $0 \leq i < n$. Since $w \in L(M)$, $q_n \in F$. Hence $q_i \in F$ for $0 \leq i \leq n$ by condition (2). That is, $w\downarrow_i \in L(M)$ for $0 \leq i \leq n$. $L(M)$ is prefix-closed.

Suppose $b \notin \Sigma_M$. Hence $b \in \Sigma \setminus \Sigma_M$. Consider the string $wa = a_1 a_2 \cdots a_n b$. Since $q_n \xrightarrow{b} q_n$ by the first condition, $wa \in F$ if and only if $w \in F$. Thus for any $w \in F$, $wb \in F$. Therefore $L(M) \triangleright \Sigma_M$. □

As in classical interpretation, we can apply the techniques developed in Section 4 to establish the soundness of various compositional rules syntactically.

Example 3. *Let M_0, M_1, A_0, A_1 and P be LTS's. Consider the following assume-guarantee rule.*

$$\frac{M_0\|A_0 \models P \qquad M_1\|A_1 \models P \qquad \models A_0 + A_1 + P}{M_0\|M_1 \models P}$$

[2]Strictly speaking, Isabelle uses natural deduction instead of Gentzen's system LK. Both proof systems are equivalent [9].

By Lemma 3, Proposition 4, Proposition 5, and Theorem 2, we can establish the soundness of the rule by finding a proof tree of the following sequent.

$$M_0 \wedge A_0 \to P, M_1 \wedge A_1 \to P, A_0 \vee A_1 \vee P \vdash_J M_0 \wedge M_1 \to P$$

The proof tree is shown in Figure 3 (b). □

It is as easy to establish compositional rules by existing proof assistants automatically. The following example demonstrates how Coq may be used to automate our proof search.

Example 4. *Let M_0, M_1, A_0, A_1, and P be LTS's. Consider the following assume-guarantee rule.*

$$\frac{M_0 \| A_0 \models P \qquad M_1 \| A_1 \models P \qquad M_0 \models A_0 + A_1}{M_0 \| M_1 \models P}$$

We will search a proof tree for the following sequent.

$$M_0 \wedge A_0 \to P, M_1 \wedge A_1 \to P, M_0 \to A_0 \vee A_1 \vdash_J M_0 \wedge M_1 \to P$$

The following transcript demonstrates that the proof search engine in Coq is able to find a proof tree by the tactic intuition.[3]

```
Coq < Goal forall M0 M1 A0 A1 P : Prop, (M0 ∧ A0 → P) ∧ (M1 ∧ A1 → P) ∧
(M0 → A0 ∨ A1) → (M0 ∧ M1 → P) .
1 subgoal

  ============================
  forall M0 M1 A0 A1 P : Prop,
      (M0 ∧ A0 → P) ∧ (M1 ∧ A1 → P) ∧ (M0 → A0 ∨ A1) → M0 ∧ M1 → P

Unnamed_thm < intuition .
Proof completed.
```
□

We now prove a circular compositional rule in the following example.

Example 5. *Let M_0, M_1, P_0, and P_1 be LTS's. Further, assume P_0 and P_1 are over Σ_0 and Σ_1 respectively, and $\Sigma_0 \cap \Sigma_1 = \emptyset$. Consider the following circular compositional rule* [2].

$$\frac{M_0 \| P_1 \models P_0 \qquad P_0 \| M_1 \models P_1}{M_0 \| M_1 \models P_0 \| P_1}$$

By Theorem 7, we have

$$\vdash (P_0 \to P_1) \wedge (P_1 \to P_0) \to (P_0 \wedge P_1).$$

Hence the soundness of the given circular compositional rule can be established by finding a proof tree for the following sequent.

$$\begin{array}{c}(P_0 \to P_1) \wedge (P_1 \to P_0) \to (P_0 \wedge P_1), \\ M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1\end{array} \vdash_J M_0 \wedge M_1 \to P_0 \wedge P_1$$

Figure 6 shows the desired proof tree.

It is also possible to automate our proof search in Coq. The following transcript shows how Coq helps us complete our proof.

[3] Again, Coq uses a natural deduction system equivalent to LJ [8].

$$
\text{R}{\to}\,\dfrac{\text{L}{\wedge}\,\dfrac{\text{L}{\to}\,\dfrac{\text{R}{\wedge}\,\dfrac{\text{R}{\to}\,\dfrac{\text{LW}\,\dfrac{\text{LW}\,\dfrac{\text{L}{\to}\,\dfrac{\text{R}{\wedge}\,\dfrac{\text{Ax}\,\dfrac{}{M_1, P_0 \vdash_J P_0} \quad \text{Ax}\,\dfrac{}{M_1, P_0 \vdash_J M_1}}{M_1, P_0 \vdash_J P_0 \wedge M_1} \quad \text{Ax}\,\dfrac{}{P_1, M_1, P_0 \;\; \vdash_J P_1}}{P_0 \wedge M_1 \to P_1, M_1, P_0 \vdash_J P_1}}{P_0 \wedge M_1 \to P_1, M_0, M_1, P_0 \vdash_J P_1}}{M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1, M_0, M_1, P_0 \vdash_J P_1}}{M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1, M_0, M_1 \vdash_J P_0 \to P_1} \quad \dfrac{\Pi_6}{\begin{array}{c} M_0 \wedge P_1 \to P_0, \\ P_0 \wedge M_1 \to P_1, \\ M_0, M_1 \end{array} \vdash_J P_1 \to P_0}}{M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1, M_0, M_1 \vdash_J (P_0 \to P_1) \wedge (P_1 \to P_0)} \quad \text{Ax}\,\dfrac{}{\begin{array}{c} P_0 \wedge P_1, \\ M_0 \wedge P_1 \to P_0, \\ P_0 \wedge M_1 \to P_1, \\ M_0, M_1 \end{array} \vdash_J P_0 \wedge P_1}}{(P_0 \to P_1) \wedge (P_1 \to P_0) \to P_0 \wedge P_1, M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1, M_0, M_1 \vdash_J P_0 \wedge P_1}}{(P_0 \to P_1) \wedge (P_1 \to P_0) \to P_0 \wedge P_1, M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1, M_0 \wedge M_1 \vdash_J P_0 \wedge P_1}}{(P_0 \to P_1) \wedge (P_1 \to P_0) \to P_0 \wedge P_1, M_0 \wedge P_1 \to P_0, P_0 \wedge M_1 \to P_1 \vdash_J M_0 \wedge M_1 \to P_0 \wedge P_1}
$$

FIGURE 6. Proof of Circular Rule

Coq < Goal forall M0 M1 P0 P1 : Prop, ((P0 → P1) ∧ (P1 → P0) → (P0 ∧ P1)) ∧ (M0 ∧ P0 → P1) ∧ (M1 ∧ P1 → P0) → (M0 ∧ M1 → P0 ∧ P1) .
1 subgoal

```
============================
forall M0 M1 P0 P1 : Prop,
  ((P0 → P1) ∧ (P1 → P0) → P0 ∧ P1) ∧
  (M0 ∧ P0 → P1) ∧ (M1 ∧ P1 → P0) → M0 ∧ M1 → P0 ∧ P1
```

Unnamed_thm < intuition .
Proof completed. □

6. On Completeness

In automated compositional reasoning, the completeness of assume-guarantee rules is required for its termination. We say an assume-guarantee rule is *complete* if it is always possible to satisfy its premises when its conclusion holds. Thanks to Lemma 3, we can formulate the completeness of assume-guarantee rules as a proof search problem. However, it requires the proof systems LK and LJ for classical and intuitionistic first-order logics. We only give an example and leave technical details in another exposition.

Example 6. *Let M_0, M_1, A_0, A_1, P_0, and P_1 be LTS's. Consider again the following assume-guarantee rule.*

$$\frac{M_0 \| A_0 \models P_0 \quad M_1 \| A_1 \models P_1 \quad M_0 \| A_0 \models A_1 \quad M_1 \| A_1 \models A_0 \quad L(\overline{A}_0 \| \overline{A}_1) = \emptyset}{M_0 \| M_1 \models P_0 \| P_1}$$

To show the rule is complete, it suffices to find a proof tree for the following sequent in system LK.

$$M_0 \wedge M_1 \to P_0 \wedge P_1 \vdash_K \exists A_0 A_1. \begin{array}{ccc} (M_0 \wedge A_0 \to P_0) & \wedge & (M_1 \wedge A_1 \to P_1) \quad \wedge \\ (M_0 \wedge A_0 \to A_1) & \wedge & (M_1 \wedge A_1 \to A_0) \quad \wedge \\ & (\neg A_0 \wedge \neg A1) \leftrightarrow \mathsf{false} & \end{array}$$

Isabelle can in fact find a proof automatically.

```
> Goal "ALL M0 M1 P0 P1 . (M0 & M1 → P0 & P1) ⇒ EX A0 A1 .
(M0 & A0 → P0) & (M1 & A1 → P1) & (M0 & A0 → A1)
& (M1 & A1 → A0) & ((¬ A0 & ¬ A1) = False)";
Level 0 (1 subgoal)
ALL M0 M1 P0 P1. M0 & M1 → P0 & P1
⇒ EX A0 A1.
        (M0 & A0 → P0) & (M1 & A1 → P1) & (M0 & A0 → A1) &
        (M1 & A1 → A0) & (¬ A0 & ¬ A1) = False
1. ALL M0 M1 P0 P1. M0 & M1 → P0 & P1
    ⇒ EX A0 A1.
        (M0 & A0 → P0) & (M1 & A1 → P1) & (M0 & A0 → A1) &
        (M1 & A1 → A0) & (¬ A0 & ¬ A1) = False
val it = [] : Thm.thm list
> auto ();
Level 1
ALL M0 M1 P0 P1. M0 & M1 → P0 & P1
⇒ EX A0 A1.
        (M0 & A0 → P0) & (M1 & A1 → P1) & (M0 & A0 → A1) &
        (M1 & A1 → A0) & (¬ A0 & ¬ A1) = False
No subgoals!
```
□

7. Conclusions

Soundness theorems for compositional reasoning rules depend on underlying computational models and can be very involved. Since it is tedious to develop new compositional rules, few such rules are available for each computational model. The limitation may impede the usability of automated compositional reasoning because verifiers are forced to mould their problems in a handful of compositional rules available to them. In this paper, we apply proof theory and develop a syntactic approach to analyze compositional rules for automated compositional reasoning. With publicly available proof assistants, we are able to establish compositional rules automatically. Our work may improve the usability of automated compositional reasoning by automatic derivation of its compositional rules.

Although all compositional rules known to us have been established automatically, it is unclear whether these proof systems are complete with respect to regular languages. It would also be of great interest if one could generate compositional rules to fit different circumstances automatically. Moreover, it is unclear whether our techniques can be applied to ω-regular sets. These questions are currently under investigation.

For the past years, research topics which combine both model checking and theorem proving are not unusual. This work may be viewed as another attempt to integrate both technologies. In contrast to previous attempts, we prefer a white-boxed approach where both techniques are coupled more tightly. Our presentation gives a rather detailed exposition in the hope to reveal theoretical foundations of both technologies.

Acknowledgment. The author is very grateful for the discussion with Edmund Clarke, Orna Grumberg, Dale Miller, Nishant Sinha, and Yih-Kuen Tsay. This work would not be possible without their insightful comments.

References

[1] Martín Abadi and Gordon D. Plotkin. A logical view of composition and refinement. In *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages, Orlando*, pages 323–332. ACM, January 1991.

[2] R. Alur and T.A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999. Invited submission to FLoC'96 special isuue. A preliminary version appears in *Proc. 11th LICS, 1996*.

[3] Howard Barringer, Dimitra Giannakopoulou, and Corina S. Păsăreanu. Proof rules for automated compositional verification through learning. In *Workshop on Specification and Verification of Component-Based Systems*, pages 14–21, 2003.

[4] Jamieson M. Cobleigh, Dimitra Giannakopoulou, and Corina S. Păsăreanu. Learning assumptions for compositional verification. In Hubert Garavel and John Hatcliff, editors, *TACAS*, volume 2619 of *LNCS*, pages 331–346. Springer-Verlag, 2003.

[5] Willem-Paul de Roever, Frank de Boer, Ulrich Hanneman, Jozef Hooman, Yassine Lakhnech, Mannes Poel, and Job Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Number 54 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[6] Robert Goldblatt. *Topoi: The Categorial Analysis of Logic*. Dover Publications, revised edition, 2006.

[7] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[8] Gérard Huet, Gilles Kahn, and Paulin-Mohring. The Coq proof assistant: a tutorial: version 6.1. Technical Report 204, Institut National de Recherche en Informatique et en Automatique, 1997.

[9] Lawrence C. Paulson and Tobias Nipkow. Isabelle tutorial and user's manual. Technical Report TR-189, Computer Laboratory, University of Cambridge, 1990.

[10] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000.

Institute of Information Science, Academia Sinica, Taiwan
E-mail address: bywang@iis.sinica.edu.tw