Machine-Checked Compositional Specification and Proofs for Embedded Systems

Karl Palmskog[1(✉)], Mattias Nyberg[2], and Dilian Gurov[1]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{palmskog,dilian}@kth.se
[2] Scania CV AB, Södertälje, Sweden
mattias.nyberg@scania.com

Abstract. The effort of formal verification of large heterogeneous systems needs to scale linearly with the number of interacting components, to be feasible in industrial practice. This is made possible by compositional specification methods and proof systems. In this paper, we demonstrate how trustworthy verified decomposition can be performed for an industry-relevant embedded system: a fuel level display. We first formalize the underlying theory in the HOL4 theorem prover, and augment this theory to allow specifications using Metric Interval Temporal Logic (MITL). We then state a top-level specification for our system using MITL and decompose it down to the system components. Our HOL4 formalization provides a corrected and extended restatement of a general specification language and proof system from previous work and showcases its usefulness for verified decomposition of systems.

Keywords: Compositional proof · formal verification · embedded systems · HOL4

1 Introduction

A fundamental problem in applying formal verification to large-scale, heterogeneous industrial systems is the complexity of performing reasoning across many independent interacting components [11]. If the complexity grows with the size of the global state space, e.g., as defined by a product of many automata, then such verification quickly becomes unfeasible. The promise of *compositional* specification methods and proof systems is that the complexity of reasoning may instead grow only linearly with the number of components in the system, thanks to contract-based reasoning at each individual component, e.g., in the assume-guarantee style.

To this end, Nyberg et al. proposed a theory for compositional specification and proof, based on first-order logic [9]. Although motivated by industrial requirements, the theory is general and abstract, even eliding commitment to

P. Rümmer and Z. Wu (Eds.): TASE 2025, LNCS 15841, pp. 67–82, 2026.
https://doi.org/10.1007/978-3-031-98208-8_5

logical formulas for describing the behavior of individual components. However, this also means that there is a significant gap between the theory as initially presented and its application to specify and verify concrete systems relevant to industrial practice. Moreover, the theory is only demonstrated on small running examples, which may be insufficient to guide application to real systems.

In this paper we consider decomposition of formal verification for an industry-relevant embedded system: a vehicle fuel-level display (FLD). To enable trustworthy decomposition of a correctness proof, we first formalize the theory of Nyberg et al. in the HOL4 theorem prover [13] and instantiate it with a notion of *timed words* in place of its abstract set of (unstructured) system *runs*. This enables us to provide a natural top-level specification of the system using Metric Interval Temporal Logic (MITL) [1,8] that can be soundly decomposed. Besides paving the way for more automated tools for compositional specification and verification, our HOL4 formalization corrects several issues in the original account of the specification and proof theory and puts its connection to first-order logic on firm ground.

In summary, we make the following *contributions*:

(i) We formalize the abstract compositional specification theory and proof system of Nyberg et al. in the HOL4 theorem prover, including its interpretation in (sorted) first-order logic and soundness of the proof system.
(ii) We instantiate the theory and proof system for timed words, and augment specifications to allow occurrences of MITL formulas.
(iii) We further instantiate the theory of timed words for an embedded fuel level display system, and demonstrate how its verification can be soundly decomposed. The system design and structure is derived from a real industrial system implemented in Scania trucks.

We provide the code that comprises the HOL4 formalization (around 6000 lines) as supplementary material to the paper [10].

2 Fuel Level Display System Description

We apply the theory of Nyberg et al. [9] for compositional specification and verification of a cyber-physical embedded system that measures, processes, and displays the fuel level in a vehicle, as illustrated in Fig. 1.

The system consists of a fuel *tank* equipped with a sensor (left), a digital *controller* (top), and an analog fuel level *meter* (right). The tank sensor has a slider connected to a "floater" that measures the fuel (sensed) level Sl, trailing the (actual) level Al in the tank. The position of the slider maps to an analog voltage Av_{in}. This voltage is converted to a digital representation Dv_{in} by an Analog-to-Digital Converter (ADC) inside the controller. The digital voltage is processed by the *program* in the controller, which consists of *infrastructure software* and *application software*. Periodically, Dv_{in} is transformed by the program into Dv_{out}, which is then converted to an analog voltage Av_{out} by a Digital-to-Analog Converter (DAC). This analog voltage then maps to a (displayed) level Dl, shown by the meter.

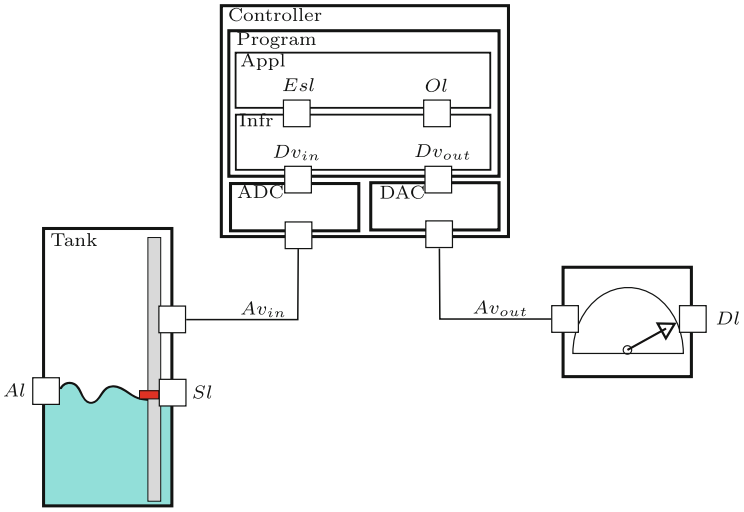

Fig. 1. Fuel level display (FLD) system architecture, diagram by Jonas Westman

In summary, the relevant variables of the fuel level display (FLD) system and their ranges and kinds are as follows:

Al, actual level, range 0–100, percentage.
Sl, sensed level (position of floater), range 0–100, percentage.
Av_{in}, analog voltage (w.r.t. ground), range $\geq$0, volts.
Dv_{in}, digital voltage, non-negative decimal number.
Esl, estimated sensed level, range 0–100, percentage.
Ol, output level, range 0–100, percentage.
Dv_{out}, digital voltage, non-negative decimal number.
Av_{out}, analog voltage out (w.r.t. ground), range $\geq$0, volts.
Dl, displayed level, range 0–100, percentage.

3 Specification Language and Proof System

In this section, we informally present our reformulation of the compositional specification language and proof system of Nyberg et al. [9], which we have formalized in HOL4, while highlighting some changes compared to the original presentation.

3.1 Syntax

The specification language syntax is divided into *components* c, *specifications* S, and *predicates* P, as shown in Fig. 2. To enable translation to a first-order logic theory (signature), components and specifications may be represented using both constants and variables. Intuitively, constants and their meaning depend on the system and context we are trying to describe, while variables are used when predicates about components and specifications contain quantification.

Component and Specification Syntax. A component is either a constant name $\mathbf{c}$, a variable q, or a composition $c_1 \times c_2$ of two components c_1 and c_2. A specification is either a constant $\mathbb{S}$, a variable V, or a combination of two specifications S_1 and S_2 in the form of a conjunction $S_1 \sqcap S_2$, an assume-guarantee pair (S_1, S_2), also referred to as *contract*, or a parallel composition $S_1 \,||\, S_2$. Special specification constants include $©$ for "compatibility" and $T_{||}$ for "top", whose meaning is elaborated below. Finally, $\mathsf{Assertional}(S)$ is meant to capture when the implementation of S is monotonic with respect to component composition.

Predicate Syntax. The predicate syntax expresses assertions about components and specifications. A component c may implement ("satisfy") a specification S, written $c : S$, and a specification S_1 may refine another specification S_2, written $S_1 \sqsubseteq S_2$. In contrast to the initial presentation [9], we distinguish quantification over components from quantification over specifications by writing $\forall_{\mathcal{C}}$ and $\forall_{\mathcal{S}}$, respectively. Moreover, we explicitly allow predicates expressing equality between two components ($=_{\mathcal{C}}$) or two specifications ($=_{\mathcal{S}}$).

$\mathbf{c}$: component constant name	$\mathbf{S}$: specification constant name
q: component variable	V: specification variable

$c ::= \mathbf{c} \mid c \times c \mid q$	component
$\mathbb{S} ::= \mathbf{S} \mid © \mid T_{\|\|}$	specification constant
$S ::= \mathbb{S} \mid S \sqcap S \mid (S, S) \mid S \,\|\|\, S \mid V$	specification
$P ::= c : S \mid S \sqsubseteq S \mid \mathsf{Assertional}(S) \mid \forall_{\mathcal{C}}\, q.\, P$	predicate
$\quad \mid \forall_{\mathcal{S}}\, V.\, P \mid P \wedge P \mid \neg P \mid c =_{\mathcal{C}} c \mid S =_{\mathcal{S}} S$	

Fig. 2. Component, specification, and predicate syntax

Example 1. Consider the predicate $c_1 \times c_2 : (A \sqcap ©, G \sqcap ©)$ in the specification language. This predicate expresses that the composition of the components c_1 and c_2 implements an assume-guarantee pair, where $A \sqcap ©$ is the assumption and $G \sqcap ©$ is the corresponding guarantee. The occurrences of the compatibility specification $©$ ensure, intuitively, that the contract is not vacuously satisfied, i.e., that there exist corresponding system runs that are consistent with the (allowed) behaviors of both components.

3.2 Semantics

We define the semantics of the specification language from Fig. 2 in terms of an abstract set Ω, whose elements are not given an explicit structure, but intuitively represent *runs* (or *executions*) of the system we are considering. For example, the elements of Ω may be (finite or infinite) sequences of system states, where

each adjacent pair is connected by a label describing a state transition, or *timed words*, as described in Sect. 4. To give meaning to constants, we use models $\mathcal{M}$ that map *component* constants to subsets of Ω, and map *specification* constants to sets of subsets of Ω. Analogously, we use substitutions σ to assign meaning to variables.

Semantics of Components. Following our definition of models, we define the semantics of components by mapping them to subsets of Ω, as shown in Fig. 3. Intuitively, the set of runs of a component describes its possible runtime behaviors. As hinted at in Example 1, the set may be empty for two composed components when they are incompatible.

$$\begin{aligned} [\![\mathbf{c}]\!]^{\sigma}_{\mathcal{M}} &= \mathcal{M}(\mathbf{c}) \\ [\![c_1 \times c_2]\!]^{\sigma}_{\mathcal{M}} &= [\![c_1]\!]^{\sigma}_{\mathcal{M}} \cap [\![c_2]\!]^{\sigma}_{\mathcal{M}} \\ [\![q]\!]^{\sigma}_{\mathcal{M}} &= \sigma(q) \end{aligned}$$

Fig. 3. Semantics of components

Semantics of Specifications. We define the semantics of specifications by mapping them to subsets of the powerset $\mathcal{P}(\Omega)$ of Ω, as shown in Fig. 4. Intuitively, describing specifications as sets of sets of runs (as opposed to simply sets of runs) means that we can capture relational properties over runs, e.g., security properties such as noninterference. We use the auxiliary definition below for the parallel operator $||$.

Definition 1. *The* double intersection *of the two sets of sets s_1 and s_2, written $s_1 \Cap s_2$, is the set $\{a \cap b \mid a \in s_1, b \in s_2\}$.*

$$\begin{aligned} [\![\mathbf{S}]\!]^{\sigma}_{\mathcal{M}} &= \mathcal{M}(\mathbf{S}) \\ [\![\copyright]\!]^{\sigma}_{\mathcal{M}} &= \{B \in \Omega \mid B \neq \emptyset\} \\ [\![T_{||}]\!]^{\sigma}_{\mathcal{M}} &= \{\Omega\} \\ [\![S_1 \sqcap S_2]\!]^{\sigma}_{\mathcal{M}} &= [\![S_1]\!]^{\sigma}_{\mathcal{M}} \cap [\![S_2]\!]^{\sigma}_{\mathcal{M}} \\ [\![S_1 || S_2]\!]^{\sigma}_{\mathcal{M}} &= [\![S_1]\!]^{\sigma}_{\mathcal{M}} \Cap [\![S_2]\!]^{\sigma}_{\mathcal{M}} \\ [\![(S_1, S_2)]\!]^{\sigma}_{\mathcal{M}} &= \{B \mid \forall B' \in [\![S_1]\!]^{\sigma}_{\mathcal{M}}.\, B \cap B' \in [\![S_2]\!]^{\sigma}_{\mathcal{M}}\} \\ [\![V]\!]^{\sigma}_{\mathcal{M}} &= \sigma(V) \end{aligned}$$

Fig. 4. Semantics of specifications

Semantics of Predicates. We define the semantics of predicates in Fig. 5 by mapping them to propositions in the metalanguage (`bool` in HOL4). We use the notation $\sigma[x \mapsto s]$ for the substitution σ updated with a mapping from the variable x to the set s. The translation uses the following auxiliary definition to define predicates of the form $\mathsf{Assertional}(S)$.

Definition 2. *A set s is* downward closed *when for every $e \in s$ and e', if $e' \subseteq e$, then $e' \in s$.*

From the basic semantic definitions, we can straightforwardly define derived predicate operators such as disjunction ($\vee$), implication ($\rightarrow$), and existential quantification ($\exists_{\mathcal{C}}$ and $\exists_{\mathcal{S}}$), which is deferred to the supplementary material.

$$
\begin{aligned}
[\![c : S]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![c]\!]^{\sigma}_{\mathcal{M}} \in [\![S]\!]^{\sigma}_{\mathcal{M}} \\
[\![S_1 \sqsubseteq S_2]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![S_1]\!]^{\sigma}_{\mathcal{M}} \subseteq [\![S_2]\!]^{\sigma}_{\mathcal{M}} \\
[\![\mathsf{Assertional}\,(S)]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![S]\!]^{\sigma}_{\mathcal{M}} \text{ is downward closed} \\
[\![\forall_{\mathcal{C}}\, q.\, P]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow \text{for all subsets } s \text{ of } \Omega,\ [\![P]\!]^{\sigma[q \mapsto s]}_{\mathcal{M}} \\
[\![\forall_{\mathcal{S}}\, V.\, P]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow \text{for all subsets } s \text{ of } \mathcal{P}(\Omega),\ [\![P]\!]^{\sigma[V \mapsto s]}_{\mathcal{M}} \\
[\![P_1 \wedge P_2]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![P_1]\!]^{\sigma}_{\mathcal{M}} \text{ and } [\![P_2]\!]^{\sigma}_{\mathcal{M}} \\
[\![\neg P]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow \text{ not } [\![P]\!]^{\sigma}_{\mathcal{M}} \\
[\![c_1 =_{\mathcal{C}} c_2]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![c_1]\!]^{\sigma}_{\mathcal{M}} = [\![c_2]\!]^{\sigma}_{\mathcal{M}} \\
[\![S_1 =_{\mathcal{S}} S_2]\!]^{\sigma}_{\mathcal{M}} &\Leftrightarrow [\![S_1]\!]^{\sigma}_{\mathcal{M}} = [\![S_2]\!]^{\sigma}_{\mathcal{M}}
\end{aligned}
$$

Fig. 5. Semantics of predicates.

Example 2. The predicate $c_1 \times c_2 : (A \sqcap \copyright, G \sqcap \copyright)$ from Example 1 is true precisely when, for every behavior $B \neq \emptyset$ that is allowed by A, the intersection of B and the behaviors of c_1 and c_2 are allowed by nonempty behaviors in G. Or more formally, the predicate is true for $\mathcal{M}$ and σ when for every $B \in [\![A]\!]^{\sigma}_{\mathcal{M}}$ such that $B \neq \emptyset$, it holds that $[\![c_1]\!]^{\sigma}_{\mathcal{M}} \cap [\![c_2]\!]^{\sigma}_{\mathcal{M}} \cap B \in [\![G \sqcap \copyright]\!]^{\sigma}_{\mathcal{M}}$.

3.3 Translation to First-Order Logic

Our translation of the specification language to first-order logic (FOL) is intuitively based on first interpreting predicates as formulas in two-sorted FOL and then leveraging the standard translation of sorts to obtain unsorted formulas [4]. Our first sort is $\mathcal{C}$, the sort of components, whose domain is $\mathcal{P}(\Omega)$, and the second sort is $\mathcal{S}$, the sort of specifications, whose domain is $\mathcal{P}(\mathcal{P}(\Omega))$. The translation from specification language predicates to unsorted FOL formulas was not explicitly defined in the work of Nyberg et al. [9]; here, we briefly sketch the translation and its correctness, and defer the full details to the supplementary material, where we use Harrison's formalized theory of FOL inside HOL4 [5].

Given a specification language model $\mathcal{M}$ and substitution σ, we construct a corresponding FOL model $L(\mathcal{M})$ and FOL substitution $L(\sigma)$, whose domain is the disjoint union of $\mathcal{P}(\Omega)$ and $\mathcal{P}(\mathcal{P}(\Omega))$. In this construction, we consider component and specification syntax as FOL function symbols, and predicate syntax as FOL predicate symbols, defining functions *c2t* and *S2t* that convert components and specifications to FOL terms, and a function *P2f* (using *c2t* and *S2t*) that translates predicates to FOL formulas. Quantification in predicates is handled by adding an implication using the additional predicate symbols *isc* and *isS*, which express that a term is a component or specification, respectively. Using this translation, Theorem 1 expresses how specification language predicates are interpreted in FOL.

Theorem 1 (Soundness of predicate translation to FOL). *For every model $\mathcal{M}$, substitution σ, and specification language predicate P, $[\![P]\!]^{\sigma}_{\mathcal{M}}$ holds precisely when $L(\mathcal{M}), L(\sigma) \models P2f(P)$, where $\models$ is the first order satisfaction relation.*

The proof is deferred to supplementary material [10], so we give an example that highlights the general idea of the translation.

Example 3. Consider a quantified version of the predicate from Example 1:

$$\forall_{\mathcal{C}}\, q.\, \forall_{\mathcal{S}}\, V_1.\, \forall_{\mathcal{S}}\, V_2.\, q : (V_1 \sqcap \copyright, V_2 \sqcap \copyright)$$

The corresponding unsorted FOL formula obtained by *P2f* is

$$\forall\, q.\, isc(q) \rightarrow \forall\, V_1.\, isS(V_1) \rightarrow \forall\, V_2.\, isS(V_2) \rightarrow \\ impl(q, ag(conj(V_1, compat), conj(V_2, compat)))$$

where *impl* is a predicate symbol corresponding to the implementation operator ":", and *ag*, *conj*, and *compat* are function symbols corresponding to the assume-guarantee, conjunction, and compatibility specification operators, respectively.

3.4 Specification Language Metatheory

In this section, we state some key formalized meta-theoretic results about the specification language, which are mostly reformulations of results from the work of Nyberg et al. [9]. The first three properties are useful for rewriting when performing proofs in the system in Sect. 3.5.

Property 1. Component composition is idempotent, associative, and commutative. That is, for all $\mathcal{M}$ and σ, $[\![c \times c =_{\mathcal{C}} c]\!]^{\sigma}_{\mathcal{M}}$, $[\![c_1 \times (c_2 \times c_3) =_{\mathcal{C}} (c_1 \times c_2) \times c_3]\!]^{\sigma}_{\mathcal{M}}$, and $[\![c_1 \times c_2 =_{\mathcal{C}} c_2 \times c_1]\!]^{\sigma}_{\mathcal{M}}$.

Property 2. Specification conjunction is idempotent, associative, and commutative. That is, for all $\mathcal{M}$ and σ, we have $[\![S \sqcap S =_{\mathcal{S}} S]\!]^{\sigma}_{\mathcal{M}}$, $[\![S_1 \sqcap (S_2 \sqcap S_3) =_{\mathcal{S}} (S_1 \sqcap S_2) \sqcap S_3]\!]^{\sigma}_{\mathcal{M}}$, and $[\![S_1 \sqcap S_2 =_{\mathcal{S}} S_2 \sqcap S_1]\!]^{\sigma}_{\mathcal{M}}$.

Property 3. Parallel composition of specifications is associative and commutative. That is, for all $\mathcal{M}$ and σ, it holds that $[\![S_1 \parallel (S_2 \parallel S_3) =_{\mathcal{S}} (S_1 \parallel S_2) \parallel S_3]\!]^{\sigma}_{\mathcal{M}}$ and $[\![S_1 \parallel S_2 =_{\mathcal{S}} S_2 \parallel S_1]\!]^{\sigma}_{\mathcal{M}}$. Idempotency does not hold.

The following five properties are the basis (in the form of introduction and elimination rules) for the proof system in Sect. 3.5.

*Property 4 (Refinement/*REF *introduction/elimination).* For all $\mathcal{M}$ and σ, if $[\![\forall_{\mathcal{C}}\, q.\, q : S_1 \to q : S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![S_1 \sqsubseteq S_2]\!]^{\sigma}_{\mathcal{M}}$ (intro); if $[\![c : S_1]\!]^{\sigma}_{\mathcal{M}}$ and $[\![S_1 \sqsubseteq S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c : S_2]\!]^{\sigma}_{\mathcal{M}}$ (elim).

*Property 5 (Assertional/*ASSN *introduction/elimination).* For all $\mathcal{M}$ and σ, if $q_1 \neq q_2$ and $[\![\forall_{\mathcal{C}}\, q_1.\, \forall_{\mathcal{C}}\, q_2.\, q_1 : S \to q_1 \times q_2 : S]\!]^{\sigma}_{\mathcal{M}}$, then $[\![\mathsf{Assertional}(S)]\!]^{\sigma}_{\mathcal{M}}$ (intro); if $[\![\mathsf{Assertional}(S)]\!]^{\sigma}_{\mathcal{M}}$ and $[\![c_1 : S]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c_1 \times c_2 : S]\!]^{\sigma}_{\mathcal{M}}$ (elim).

*Property 6 (Conjunction/*CONJ *introduction/elimination).* For all $\mathcal{M}$ and σ, if $[\![c : S_1]\!]^{\sigma}_{\mathcal{M}}$ and $[\![c : S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c : S_1 \sqcap S_2]\!]^{\sigma}_{\mathcal{M}}$ (intro); if $[\![c : S_1 \sqcap S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c : S_1]\!]^{\sigma}_{\mathcal{M}}$ and $[\![c : S_2]\!]^{\sigma}_{\mathcal{M}}$ (elim).

*Property 7 (Parallel/*PAR *introduction/elimination).* For all $\mathcal{M}$ and σ, if $[\![c_1 : S_1]\!]^{\sigma}_{\mathcal{M}}$ and $[\![c_2 : S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c_1 \times c_2 : S_1 \parallel S_2]\!]^{\sigma}_{\mathcal{M}}$ (intro); if variables q_1, q_2 do not occur in c for $q_1 \neq q_2$, and $[\![c : S_1 \parallel S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![\exists_{\mathcal{C}}\, q_1.\, \exists_{\mathcal{C}}\, q_2.\, q_1 : S_1 \wedge q_2 : S_2 \wedge c =_{\mathcal{C}} q_1 \times q_2]\!]^{\sigma}_{\mathcal{M}}$ (elim).

*Property 8 (Contract/*CONT *introduction/elimination).* For all $\mathcal{M}$ and σ, if q_1 does not occur in c_2 and $[\![\forall_{\mathcal{C}}\, q_1.\, q_1 : S_1 \to q_1 \times c_2 : S_2]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c_2 : (S_1, S_2)]\!]^{\sigma}_{\mathcal{M}}$ (intro); if $[\![c_1 : S_1]\!]^{\sigma}_{\mathcal{M}}$ and $[\![c_2 : (S_1, S_2)]\!]^{\sigma}_{\mathcal{M}}$, then $[\![c_1 \times c_2 : S_2]\!]^{\sigma}_{\mathcal{M}}$ (elim).

Finally, the two properties below shed some light on specifications, in particular conjunction, the top specification and assertional contracts.

Property 9 (Conjunction vs. parallel composition). For all $\mathcal{M}$ and σ, $[\![S_1 \sqcap S_2 \sqsubseteq S_1 \parallel S_2]\!]^{\sigma}_{\mathcal{M}}$; if $[\![\mathsf{Assertional}(S_1)]\!]^{\sigma}_{\mathcal{M}}$ and $[\![\mathsf{Assertional}(S_2)]\!]^{\sigma}_{\mathcal{M}}$, then $[\![S_1 \parallel S_2 \sqsubseteq S_1 \sqcap S_2]\!]^{\sigma}_{\mathcal{M}}$.

Property 10 (Generality, assertional contracts). For all $\mathcal{M}$ and σ, $[\![S =_{\mathcal{S}} (T_{||}, S)]\!]^{\sigma}_{\mathcal{M}}$ holds, and if $[\![\mathsf{Assertional}(S_2)]\!]^{\sigma}_{\mathcal{M}}$, then $[\![\mathsf{Assertional}((S_1, S_2))]\!]^{\sigma}_{\mathcal{M}}$.

3.5 Proof System

We define a proof system for the specification language as an inductive relation $\Gamma \vdash P$, where Γ is a set of predicates used as premises. The proof system rules are shown in Fig. 6, where we use the notation $vars(c)$ for the set of variables in a component c, and $P[c/q]$ for the capture-avoiding substitution of the component c for the variable q in P (and $P[S/V]$ analogously for specifications).

Theorem 2 (Proof system soundness). *The proof system in Fig. 6 is sound with respect to the predicate semantics in Fig. 4. That is, whenever $\Gamma \vdash P$, then for all $\mathcal{M}$ and σ, if $[\![P']\!]^{\sigma}_{\mathcal{M}}$ for all $P' \in \Gamma$, then $[\![P]\!]^{\sigma}_{\mathcal{M}}$.*

$$\frac{\Gamma \vdash \forall_{\mathcal{C}}\, q.\, P}{\Gamma \vdash P[c/q]}\ \text{ALL\_EL\_C} \qquad \frac{}{\Gamma, P \vdash P}\ \text{AX} \qquad \frac{\Gamma \vdash \forall_{\mathcal{C}}\, q.\, q : S_1 \to q : S_2}{\Gamma \vdash S_1 \sqsubseteq S_2}\ \text{REF\_IN}$$

$$\frac{\Gamma \vdash c : S_1 \qquad \Gamma \vdash S_1 \sqsubseteq S_2}{\Gamma \vdash c : S_2}\ \text{REF\_EL} \qquad \frac{\Gamma \vdash c_1 : S_1 \qquad \Gamma \vdash c_2 : S_2}{\Gamma \vdash c_1 \times c_2 : S_1 || S_2}\ \text{PAR\_IN} \qquad \frac{\Gamma \vdash c : S_1 \sqcap S_2}{\Gamma \vdash c : S_1}\ \text{CONJ\_EL1}$$

$$\frac{q_1 \neq q_2 \qquad \Gamma \vdash \forall_{\mathcal{C}}\, q_1.\, \forall_{\mathcal{C}}\, q_2.\, q_1 : S \to q_1 \times q_2 : S}{\Gamma \vdash \mathsf{Assertional}\,(S)}\ \text{ASSN\_IN} \qquad \frac{\Gamma \vdash c : S_1 \sqcap S_2}{\Gamma \vdash c : S_2}\ \text{CONJ\_EL2}$$

$$\frac{\Gamma \vdash \mathsf{Assertional}\,(S) \qquad \Gamma \vdash c_1 : S}{\Gamma \vdash c_1 \times c_2 : S}\ \text{ASSN\_EL} \qquad \frac{q \notin \mathit{vars}\,(c) \qquad \Gamma \vdash \forall_{\mathcal{C}}\, q.\, q : S_1 \to q \times c : S_2}{\Gamma \vdash c : (S_1, S_2)}\ \text{CONT\_IN}$$

$$\frac{\Gamma \vdash c : S_1 \qquad \Gamma \vdash c : S_2}{\Gamma \vdash c : S_1 \sqcap S_2}\ \text{CONJ\_IN} \qquad \frac{\Gamma \vdash c : S_1 \sqcap S_3 \qquad \Gamma \vdash S_1 \sqsubseteq S_2}{\Gamma \vdash c : S_2 \sqcap S_3}\ \text{CR} \qquad \frac{\Gamma \vdash c_1 : S_1 \qquad \Gamma \vdash c_2 : (S_1, S_2)}{\Gamma \vdash c_1 \times c_2 : S_2}\ \text{CONT\_EL}$$

$$\frac{q' \notin \mathit{fcvars}\,(P) \qquad q' \notin \mathit{fcvars}\,(\Gamma) \qquad \Gamma \vdash P[q'/q]}{\Gamma \vdash \forall_{\mathcal{C}}\, q.\, P}\ \text{ALL\_IN\_C} \qquad \frac{\Gamma \vdash c =_{\mathcal{C}} c' \qquad \Gamma \vdash P[c/q]}{\Gamma \vdash P[c'/q]}\ \text{EQ\_EL\_C} \qquad \frac{\Gamma \vdash S =_{\mathcal{S}} S' \qquad \Gamma \vdash P[S/V]}{\Gamma \vdash P[S'/V]}\ \text{EQ\_EL\_S}$$

$$\frac{\begin{array}{l} q_1 \neq q_2 \\ q_1 \notin \mathit{vars}\,(c) \\ q_2 \notin \mathit{vars}\,(c) \\ \Gamma \vdash c : S_1 || S_2 \end{array}}{\Gamma \vdash \exists_{\mathcal{C}}\, q_1.\, \exists_{\mathcal{C}}\, q_2.\, q_1 : S_1 \wedge q_2 : S_2 \wedge c =_{\mathcal{C}} q_1 \times q_2}\ \text{PAR\_EL}$$

Fig. 6. Proof system rules.

Compared to the original proof system by Nyberg et al. [9], we define and prove sound rules for the sorted quantifiers and other standard FOL operators as part of the supplementary material, and also add equality elimination rules for rewriting, which were previously used implicitly.

Example 4. Let A, A_1, A_2, G, G_1, G_2 be specifications in the set

$$\Gamma_{ex} = \{\mathsf{Assertional}(A), A \sqsubseteq A_1, A \sqsubseteq G_1 \sqcap A_2, G_2 \sqcap G\}$$

and also

$$\begin{aligned} \Delta_{ex} = \{&\forall_{\mathcal{C}}\, q.\, \forall_{\mathcal{C}}\, q'.\, \forall_{\mathcal{C}}\, q''.\, (q \times q') \times q'' =_{\mathcal{C}} q \times (q' \times q''), \\ &G_1 \sqcap A =_{\mathcal{S}} A \sqcap G_1,\ G_1 \sqcap (A \sqcap \copyright) =_{\mathcal{S}} (A \sqcap G_1) \sqcap \copyright\} \end{aligned}$$

Define the predicate P_{ex} as

$$\begin{aligned} &\forall_{\mathcal{C}}\, q_1.\, q_1 : (A_1 \sqcap \copyright, G_1 \sqcap \copyright) \to \\ &\quad \forall_{\mathcal{C}}\, q_2.\, q_2 : (A_2 \sqcap \copyright, G_2 \sqcap \copyright) \to q_1 \times q_2 : (A \sqcap \copyright, G \sqcap \copyright) \end{aligned}$$

As shown formally in HOL4 in the supplementary material, it holds that $\Gamma_{ex} \cup \Delta_{ex} \vdash P_{ex}$; this corrects the previous non-mechanized proof [9]. Since the premises in Δ_{ex} are true for any $\mathcal{M}$ and σ, we can conclude using Theorem 2 that $[\![P_{ex}]\!]^{\sigma}_{\mathcal{M}}$ whenever $[\![P]\!]^{\sigma}_{\mathcal{M}}$ for every $P \in \{\mathsf{Assertional}(A), A \sqsubseteq A_1, A \sqsubseteq G_1 \sqcap A_2, G_2 \sqcap G\}$.

Finally, using proof system soundness we can obtain decomposition corollaries such as the following:

Corollary 1. *For all $\mathcal{M}$ and σ, whenever it holds that*

- $[\![S_1||S_2||S_3 \sqsubseteq S]\!]^{\sigma}_{\mathcal{M}}$,
- $[\![P]\!]^{\sigma}_{\mathcal{M}}$ *for every* $P \in \Gamma$,
- $\Gamma \vdash c_1 : S_1$, $\Gamma \vdash c_2 : S_2$, *and* $\Gamma \vdash c_3 : S_3$,

then $[\![c_1 \times c_2 \times c_3 : S]\!]^{\sigma}_{\mathcal{M}}$.

In other words, to establish that the composed system $c_1 \times c_2 \times c_3$ satisfies the specification S, it is sufficient to establish that some "component" specifications S_1, S_2, and S_3 together refine S, and then use the proof system to establish that each component satisfies its corresponding specification.

4 Specifications on Timed Words

The specification language of Sect. 2 is parameterized on an abstract set of system runs Ω, intuitively representing all possible executions of systems in the domain being modeled. As a first step towards reasoning about the FLD system (Fig. 1), we instantiate Ω as an (abstract) set Ω_{TW} of *timed words*, where each element may be viewed as an infinite sequence of samples of system states from a real-valued timeline, as made precise below. We also extend the specification language to include Metric Interval Temporal Logic (MITL) formulas [1,8]. In contrast to proof systems based simply on sets of runs (or traces) for temporal logics, the instantiation allows relational properties over timed words.

4.1 Specification Language Extension

To allow expressing properties at the level of individual timed words, we enrich the specification language with MITL formulas as shown in Fig. 7. Specifically, we add MITL formulas as a form of specification language constant $\mathbb{S}$, using the notation $\widehat{\phi}$ to indicate that the MITL formula ϕ is used as a specification. Formulas may contain closed or open intervals I, bounded by non-negative integers a and b. The MITL syntax is parameterized on "atomic" formulas p that may express properties of individual system states in runs.

Intuitively, a formula $\phi_1 \mathsf{U}_I \phi_2$ states that ϕ_1 is true *until* ϕ_2 becomes true inside the interval I (measured from the current point in time). Conversely, for $\phi_1 \mathsf{S}_I \phi_2$, ϕ_1 is true *since* ϕ_2 was true in the past.

Following this pattern, $\Box_I \phi$ and $\boxminus_I \phi$ state that ϕ is always true inside the interval I, forward and backward in time, respectively. $\Diamond_I \phi$ and $\diamondminus_I \phi$ state that ϕ is true at some point inside I, again forward and backward in time.

$$
\begin{aligned}
I &::= [a,b] \mid (a,b] \mid [a,b) \mid [a,\infty) \mid (a,b) \mid (a,\infty) \\
\phi &::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \,\mathsf{U}_I\, \phi' \mid \phi \,\mathsf{S}_I\, \phi' \\
&\quad \mid \Box_I\, \phi \mid \Diamond_I\, \phi \mid \boxminus_I\, \phi \mid \diamondminus_I\, \phi \\
\mathbb{S} &::= \mathbf{S} \mid \copyright \mid T_{||} \mid \widehat{\phi}
\end{aligned}
$$

Fig. 7. Extended specification syntax for timed words.

4.2 Semantics of MITL Formulas and Extended Specifications

This section describes our semantics of the specification language extension from above, which assigns sets of timed words to MITL formulas. This can be viewed as an adaptation of MITL from previous work [8] to our setting.

Definition 3 (Timed words). *Let $\mathcal{A}$ be a set of* system states*, and let τ be a function from natural numbers $\mathbb{N}$ to tuples $\mathcal{A} \times \mathbb{R}_{\geq 0}$. We call τ a* timed word*, and for $k \in \mathbb{N}$, we write $aval(\tau(k))$ for the first component of the tuple, the system state at k, and $tval(\tau(k))$ for the second component, representing the state's moment in time.*

Our intention is intuitively for a timed word τ to describe a (countable) set of samples of a system's state over real-valued time, and for this intuition to make sense, we must have $tval(\tau(i)) < tval(\tau(j))$ whenever $i < j$.

Before defining the semantics of MITL formulas, we define operations on intervals, following previous work using MITL.

Definition 4 (Positive and negative shift). *The* positive shift operator $\oplus$ *takes a real number t and an interval I and returns sets of non-negative real numbers, as defined below. The* negative shift operator $\ominus$ *is defined analogously.*

$$
\begin{aligned}
t \oplus [a,b] &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a \leq r \leq t + b \,\} \\
t \oplus (a,b] &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a < r \leq t + b \,\} \\
t \oplus [a,b) &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a \leq r < t + b \,\} \\
t \oplus [a,\infty] &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a \leq r \,\} \\
t \oplus (a,b) &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a < r < t + b \,\} \\
t \oplus (a,\infty) &= \{\, r \in \mathbb{R}_{\geq 0} \mid t + a < r \}
\end{aligned}
$$

Using shifts, we define the semantics of a MITL formula ϕ by mapping the formula and a natural number i (indicating the word position) to a set of timed words in Ω_{TW}, as shown in Fig. 8. The semantics is parameterized on a relation $\models$ which defines which MITL atomic propositions are true in a system state. Finally, we define the semantics of the extended specifications from Fig. 7 following the definitions in Fig. 3 (instantiating Ω to Ω_{TW}) by adding $[\![\widehat{\phi}]\!]^{\sigma}_{\mathcal{M}} = \mathcal{P}([\![\phi]\!]^0)$. Intuitively, this means that MITL formulas in specifications denote the set of sets of timed words that satisfy the formula.

$$
\begin{aligned}
[\![p]\!]^i &= \{ \tau \in \Omega_{TW} \mid aval(\tau(i)) \models p \} \\
[\![\neg\phi]\!]^i &= \Omega_{TW} \setminus [\![\phi]\!]^i \\
[\![\phi \wedge \phi']\!]^i &= [\![\phi]\!]^i \cap [\![\phi']\!]^i \\
[\![\phi \,\mathsf{S}_I\, \phi']\!]^i &= \{ \tau \in \Omega_{TW} \mid \exists j.\ tval(\tau(j)) \in tval(\tau(i)) \ominus I \wedge \tau \in [\![\phi']\!]^j \wedge \\
&\qquad \forall k.\ tval(\tau(k)) \in \{ r \in \mathbb{R} \mid tval(\tau(j)) \leq r \leq tval(\tau(i)) \} \rightarrow \tau \in [\![\phi]\!]^k \ \} \\
[\![\phi \,\mathsf{U}_I\, \phi']\!]^i &= \{ \tau \in \Omega_{TW} \mid \exists j.\ tval(\tau(j)) \in tval(\tau(i)) \oplus I \wedge \tau \in [\![\phi']\!]^j \wedge \\
&\qquad \forall k.\ tval(\tau(k)) \in \{ r \in \mathbb{R} \mid tval(\tau(i)) \leq r \leq tval(\tau(j)) \} \rightarrow \tau \in [\![\phi]\!]^k \ \} \\
[\![\boxminus_I \phi]\!]^i &= \{ \tau \in \Omega_{TW} \mid \forall j.\ tval(\tau(j)) \in tval(\tau(i)) \ominus I \rightarrow \tau \in [\![\phi]\!]^j \ \} \\
[\![\diamondminus_I \phi]\!]^i &= \{ \tau \in \Omega_{TW} \mid \exists j.\ tval(\tau(j)) \in tval(\tau(i)) \ominus I \wedge \tau \in [\![\phi]\!]^j \ \} \\
[\![\Box_I \phi]\!]^i &= \{ \tau \in \Omega_{TW} \mid \forall j.\ tval(\tau(j)) \in tval(\tau(i)) \oplus I \rightarrow \tau \in [\![\phi]\!]^j \ \} \\
[\![\Diamond_I \phi]\!]^i &= \{ \tau \in \Omega_{TW} \mid \exists j.\ tval(\tau(j)) \in tval(\tau(i)) \oplus I \wedge \tau \in [\![\phi]\!]^j \ \}
\end{aligned}
$$

Fig. 8. MITL semantics

4.3 Proof System Extension

Rather than extending our proof system from Fig. 6 with rules for general reasoning about MITL, we introduce the rule below to allow proofs to use MITL formulas as premises. We expect that MITL formula premises in proofs could be discharged in many ways, such as via semantic reasoning in a theorem prover or using a model checker.

$$\frac{\phi_1 \rightarrow \phi_2}{\Gamma \vdash \widehat{\phi_1} \sqsubseteq \widehat{\phi_2}}\ \text{MITL}$$

5 Fuel Level Display Verification Decomposition

In this section, we describe the application of specifications on timed words from Sect. 4 to the Fuel Level Display (FLD) system outlined in Sect. 2.

5.1 System Runs and Variables

To enable reasoning about the FLD system, we restrict the set of runs Ω_{TW} per Definition 3 to obtain the set Ω_{FLD}, where atomic propositions p_{FLD} express properties of individual FLD system states inside timed words. More specifically, we let p_{FLD} contain all propositions of the form $x = v$, $x < v$ and $x > v$ (and conjunctions of them) in the expected way, where x is an FLD variable and v a value, e.g., between 0 and 100. Moreover, we define an FLD system state as a mapping of all FLD variables to corresponding values.

Definition 5. *The set of possible runs of the FLD system is a set Ω_{FLD} of timed words τ such that:*

- *for all $i, j \in \mathbb{N}$ and all $\tau \in \Omega_{FLD}$, if $i < j$ then $tval(\tau(i)) < tval(\tau(j))$.*
- *for all $i \in \mathbb{N}$ and all $\tau \in \Omega_{FLD}$, $aval(\tau(i))$ maps variables to allowed values.*

5.2 Specification

The top-level FLD specification intuitively says that if the display level Dl shows r (a percentage), then at some recent point in the past, the actual level Al was also approximately r (a percentage). Using a MITL formula we call ϕ_{FLD}, we express this intuition more formally as follows:

$$\Box_{[0,\infty)}\,(Dl = r \rightarrow \Diamond_{[0,t]}\,(Al \approx_m r)).$$

Here, $x \approx_m v$ is a shorthand for the absolute difference of x and v being less or equal to a predetermined margin of error m, e.g., 3% points. Further, t is a positive integer that is similarly a predetermined margin of error, expressed in units of time (e.g., milliseconds) that gives an upper bound on the reaction time of the system. We then define the specification term S_{FLD} as $\widehat{\phi}_{FLD}$.

This means that if we consider the FLD system abstractly as a component c_{FLD}, to *verify* FLD, we need to establish that $c_{FLD} : S_{FLD}$.

5.3 Decomposition

Based on the system structure, the FLD system can be viewed as a composition of three components c_{meter}, c_{ctrl}, and c_{tank}, i.e.,

$$c_{FLD} = c_{meter} \times c_{ctrl} \times c_{tank}.$$

We can then give MITL formulas for each major component with corresponding intervals t_1, t_2, and t_3 (that together must add up to t or less):

$$\begin{aligned}
\phi_{meter} &: \Box_{[0,\infty)}\,(Dl = f(v) \rightarrow \Diamond_{[0,t_1]}\,(A_{v_{in}} \approx_m v)) \\
\phi_{ctrl} &: \Box_{[0,\infty)}\,(A_{v_{in}} \approx_m v \rightarrow \Diamond_{[0,t_2]}\,(A_{v_{out}} \approx_m v)) \\
\phi_{tank} &: \Box_{[0,\infty)}\,(A_{v_{out}} \approx_m v \rightarrow \Diamond_{[0,t_3]}\,(Al \approx_m f(v)))
\end{aligned}$$

where we assume f is a function translating analog voltages to percentage points.

To enable trustworthy compositional verification of the whole FLD system, we would like to reduce the verification of $c_{FLD} : S_{FLD}$ to verification of its components. Using Corollary 1, we therefore reduce verification of $c_{FLD} : S_{FLD}$ to obtaining proofs of the following:

- $S_{meter}||S_{ctrl}||S_{tank} \sqsubseteq S_{FLD}$, which reduces to reasoning on the semantics of MITL formulas—specifically, showing inclusion in the FLD specification timed word set for the intersection of the three component specification timed word sets.
- $c_{meter} : S_{meter}$ where $S_{meter} = \widehat{\phi}_{meter}$, whose proof can use the proof system.
- $c_{ctrl} : S_{ctrl}$ where $S_{ctrl} = \widehat{\phi}_{ctrl}$, whose proof can use the proof system.
- $c_{tank} : S_{tank}$ where $S_{tank} = \widehat{\phi}_{tank}$, whose proof can use the proof system.

6 Formalization in the HOL4 Theorem Prover

The HOL4 formalization of the specification language and its metatheory and instantiation for timed words is around 6000 lines of code and is available as supplementary material to the paper [10]. We used the Ott tool [12] to define the specification language and proof system and exported them to HOL4. We mainly chose HOL4 due to its easily accessible and trustworthy formalization of first-order logic and corresponding model theory, translated from the HOL Light system [5,6]. For convenience, we defined a syntax and semantics of FOL which uses strings for symbols such as atomic propositions rather than natural numbers as used by Harrison. However, we prove this string-based semantics equivalent to the numbers-based semantics that is part of the examples distributed with HOL4, release trindemossen-1.

7 Limitations and Challenges for Applications

The formalized specification language and metatheory (and its instantiation for timed words) aims provide a foundation for sound decomposition of correctness proofs for embedded systems consisting of many components to correctness proofs about individual components. We believe the specification language is sufficiently general to account for many practically relevant notions of "system runs" in addition to timed words, and also capture common idioms for contract based specifications of components in embedded systems, e.g., as expressed by Cimatti and Tonetta [3] or Benveniste et al. [2].

However, while the specification language and proof system can already be practically used in a sound way inside the HOL4 theorem prover, we do not believe this mode of use is feasible for large-scale applications due to the considerable training and expertise required for using interactive theorem provers. Instead, the HOL4 formalization can be viewed as a guide and ground truth for standalone practical tools for specification decomposition and proof search. For example, the HOL4 formalization could be used to provide a certified checker of proof system proofs encoded in some intermediate textual format. Such intermediate textual format proofs could be produced by actual users through some untrusted graphical manual proof tool, or by an untrusted automated proof tool using unverified heuristics. By having text-format proofs certified by a checker directly connected to the formalized theory, trust in such proofs is reduced to trust in HOL4, which uses the so-called *LCF approach* of a small trusted proof kernel [13]. Another option is to directly exploit the connection to FOL by using automated FOL solvers such as Vampire [7] to prove specification predicates, but we expect this approach to be less trustworthy unless the FOL proofs can also be reconstructed in HOL4, which may be difficult.

The specification language and proof system is meant to abstract from details about real-world embedded system components by allowing them to be represented by opaque constants, e.g., c_{meter}. We expect that during practical application, users will *assume* (by adding to the set of premises) that component constants satisfy a certain specification, and possibly later establish this fact using

some independent appropriate method such as model checking. Another way to control the complexity is by representing some (real-world) composed components by a single component constant. In this way, usage of the specification language and proof system can scale to large systems with many components. Nevertheless, to preserve trustworthiness, the metatheory of language should ideally be instantiated to the required notion of system run, which requires interactive theorem prover expertise. In addition, applications should validate that the formal definition of system run (given by the set Ω) corresponds to what is meant by run in practice, which might only be feasible to do experimentally.

8 Conclusions

We presented a general theory of compositional specifications which we formalized, along with its metatheory and proof system, in the HOL4 theorem prover. We also instantiated the general system to decompose verification for a fuel level display system.

Previous work on the specification language did not explicitly consider variables and substitutions, or explicitly elaborate the connection to FOL [9]. Most of our corrections to the language and proof system are related to variable management, but we also add predicates and rules for (sorted) equality and demonstrate that they are crucial for practical use, in particular for rewriting specification terms to assume the proper shape.

Following this abstract validation of the language, several tool-related developments become possible that enable practical use of the language and system in a trustworthy way. Most directly, it becomes possible to perform automatic proof search (inside or outside HOL4), e.g., using proof search strategies to automatically perform decomposition of a top-level specification. Thanks to the translation to first-order logic, automated semantic reasoning could also be performed using existing automated solvers.

References

1. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996). https://doi.org/10.1145/227595.227602
2. Benveniste, A., et al.: Contracts for system design. Found. Trends Electron. Des. Autom. **12**(2–3), 124–400 (2018). https://doi.org/10.1561/1000000053
3. Cimatti, A., Tonetta, S.: Contracts-refinement proof system for component-based embedded systems. Sci. Comput. Program. **97**, 333–348 (2015). https://doi.org/10.1016/j.scico.2014.06.011
4. Enderton, H.B.: A Mathematical Introduction to Logic, 2nd edn. Academic Press (2001)
5. Harrison, J.: Formalizing basic first order model theory. In: Grundy, J., Newey, M. (eds.) TPHOLs 1998. LNCS, vol. 1479, pp. 153–170. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055135
6. Harrison, J.: Compendium of old work on formalizing metatheory of first-order logic and proof procedures (2019). https://github.com/jrh13/hol-light/commit/013324af7ff715346383fb963d323138

7. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Computer Aided Verification, pp. 1–35. Springer, Heidelberg (2013)
8. Maler, O., Nickovic, D., Pnueli, A.: From MITL to timed automata. In: Asarin, E., Bouyer, P. (eds.) FORMATS 2006. LNCS, vol. 4202, pp. 274–289. Springer, Heidelberg (2006). https://doi.org/10.1007/11867340_20
9. Nyberg, M., Westman, J., Gurov, D.: Formally proving compositionality in industrial systems with informal specifications. In: Margaria, T., Steffen, B. (eds.) ISoLA 2020. LNCS, vol. 12478, pp. 348–365. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61467-6_22
10. Palmskog, K.: A theory of specifications, components, contracts, and compositionality formalized in HOL4 (2025). https://doi.org/10.5281/zenodo.15321668. https://zenodo.org/records/15321668
11. Roever, W.-P.: The need for compositional proof systems: a survey. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 1–22. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49213-5_1
12. Sewell, P., et al.: Ott: effective tool support for the working semanticist. J. Funct. Program. **20**(1), 71–122 (2010). https://doi.org/10.1017/S0956796809990293
13. Slind, K., Norrish, M.: A brief overview of HOL4. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 28–32. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_6