EFSMT: A Logical Framework for the Design of Cyber-Physical Systems

Chih-Hong Cheng
ABB Corporate Research, Germany

Saddek Bensalem
Verimag Laboratory, France

Harald Ruess
fortiss GmbH, Germany

Natarajan Shankar
SRI International, USA

Ashish Tiwari
SRI International, USA

ABSTRACT

The design of cyber-physical systems is challenging in that it involves the analysis and synthesis of software-intensive, distributed, real-time systems for controlling, possibly safety-relevant, plants in complex physical habitats. We tackle this formidable challenge with EFSMT, an exists-forall (EF) quantified first-order fragment of propositional combinations over constraints, which are interpreted with respect to an open-ended combination of suitable background theories (satisfiability modulo theories, SMT). We demonstrate the expressiveness of EFSMT by reducing a number of pivotal verification and synthesis problems to EFSMT, including a template-based reduction of the synthesis of Lyapunov functions in nonlinear control, distributed priority synthesis for orchestrating system components, and parameter synthesis for hybrid control systems. We are also proposing a verification procedure for automatically solving EFSMT problems based on the interplay between two SMT solvers for, respectively, solving universally and existentially quantified problems. Our verification procedure builds on many features found in modern SMT solvers, including backtracking search and model generation, and it generalizes the applicability of SMT solvers to quantifier reasoning by means of a candidate- and counterexample-guided search for witnesses for existentially-quantified variables. Altogether these developments demonstrate the suitability of EFSMT as the underlying logical framework for expressing and automating a large variety of typical design problems for cyber-physical system.

1. INTRODUCTION

Designing trustworthy cyber-physical systems (CPS) is challenging in that it includes the analysis and synthesis of software-intensive, distributed, real-time systems for controlling plants operating in uncertain, and unknown physical habitats, which may be evolving non-stop in continuous time. The development of a comprehensive and coherent design framework for complex CPS is needed, since current CPS are engineered and maintained at very high cost and sometimes with unknown risks. Such a design framework needs to address multidisciplinary modelling of mixed discrete-continuous systems — including software, electronics, mechanics, and physics — and consistent views thereof, concepts for composing cyber-physical components, and the automation of, possibly evolving, CPS designs.

In addressing these formidable challenges we are proposing an exists-forall fragment of first-order logic with an open-ended combination of background theories as a *logical framework* for formally expressing and automatically solving a wide range of typical CPS design problems. The automated verification engine for this quantified logic is based on the combination of two solvers for propositional constraints. This interplay between the two solvers works by constantly exchanging candidate witnesses for existentially quantified variables with counterexamples for universally quantified variables and by restricting search spaces for witnesses by means of lemma learning and extrapolation from failed proof attempts.

A significant leap forward in formal verification over the last two decades has been the development of high-performant SMT solvers for solving *forall* quantified problems over a rich class of theories that is adequate for a large number of cyber-physical design problems (e.g. [5, 6, 8]). For example, SMT has successfully been employed for optimal task scheduling, bounded model checking for timed automata [18] and other infinite-state systems [7], the detection of concurrent errors, and behavioral-level planning.

EFSMT extends SMT from top-level quantified *forall* problems to *exists-forall* and it brings a corresponding advance to formal synthesis [10, 11]. We are distinguishing between three basic challenges in any formal synthesis problem: (1) how to specify the design intent, (2) what class of solutions to search for, and (3), how to automate the search for solutions.

We show how a wide variety of synthesis problems for cyber-physical systems may be specified and solved using EFSMT. Universally quantified variables are used for modeling uncertainties, and the search for design parameters reduces to finding appropriate instantiations for existentially bound variables. We demonstrate this EFSMT-based synthesis approach by means of, first, parameter synthesis for hybrid control systems, and, second, by means of priority synthesis for scheduling and orchestrating distributed components.

In the template-based approach to synthesis, the designer sketches an outline of a desired solution, and the synthesis is expected to fill in the details. A trivial example of the template for a program invariant is, say, $ax + by < c$ for some parameters a, b, and c. Formally, this is encoded in EFSMT as $(\exists a, b, c)(\forall x, y)\, ax + by < c$, where x and y are program variables, and the parameters a, b, and c need to be instantiated by the synthesis procedure in the context of the program or design concerned to produce the concrete invariant. Variants on this formulation can be used to solve safety games, express assumption synthesis for finding the weakest environment in which a given component meets its requirements), supervisory con-

troller synthesis (design an algorithm to selectively disable component actions so that it satisfies some goal in the face of uncontrollable actions by the environment), and full program synthesis (design an algorithm to achieve some given goal). Here we are demonstrating the template-based EFSMT synthesis approach by means of constructing Lyapunov functions for nonlinear control systems.

For automating the search for solutions, we are proposing a verification engine for solving EFSMT formulas, which is based on the interplay of two SMT solvers for formulas of different polarity as determined by the top-level exists-forall quantifier alternation. The basic framework for combining two propositional solver and exchanging potential witnesses and counterexamples for directing the proof search has previously been proposed for solving 2QBF [16] and for counterexample-guided inductive synthesis [19]. We lift this procedure to the EFSMT logic and propose a number of optimizations, such as extrapolation, which is inspired by the concept of widening in abstract interpretation. The extended report [3] provides additional information on the design of our EFSMT solver together with an evaluation of this implementation on our case studies. This EFSMT solver is currently being integrated into Yices [8].

Altogether, the main contributions of this paper are (1) the design and implementation of an EFSMT solver based on established SMT solver technology, and (2) reductions of a large variety of pivotal analysis and synthesis tasks for CPS, including parameter synthesis, stability analysis, and scheduler synthesis for orchestrating interacting components, to logical problems in EFSMT. The large variety of pivotal design tasks expressible in EFSMT undergird our hypothesis that this logic may indeed form a suitable logical framework for expressing and automating a large number of CPS design tasks.

The rest of the paper is structured as follows. We describe the exists-forall problem and our approach to solving EFSMT problems in Section 2. Section 3 includes our case studies and reductions of pivotal CPS design tasks to logical EFSMT problems. Section 4 closes with concluding remarks and an outlook to further developments.

2. SOLVING EFSMT

Satisfiability modulo theories.

A satisfiability modulo theories (SMT) problem is a decision problem for logical formulas with respect to combinations of background theories expressed in classical first-order logic with equality. Examples of theories typically used in program analysis include the theory of real numbers, the theory of integers, and the theories of various data structures such as lists, arrays, or bitvectors. For example, in

$$x + 2 = y \Rightarrow f(\texttt{read}(\texttt{write}(a, x, 3), y - 2)) = f(y - x + 1)$$
$$(f(x)f(y)) = f(z) \wedge x + z \leq y \leq x \Rightarrow z < 0$$

the linear inequalities over real variables x, y, z are evaluated using the rules of the theory of linear real arithmetic, whereas predicates involving uninterpreted terms and function symbols are evaluated using the rules of the theory of uninterpreted functions with equality, and the array functions `read` and (`write` are evaluated using the rules of a theory of functional arrays (a is an array variable) .

SMT solvers such as Yices and Z3 are almost ubiquitous in research on program analysis and many industrial applications. These solvers decide satisfiability with respect to a given combination of background theories. It the input formula is satisfiable, an SMT solver constructs an explicit variable assignment or *witness* in the form of ground substutition for the free variables; therefore, free

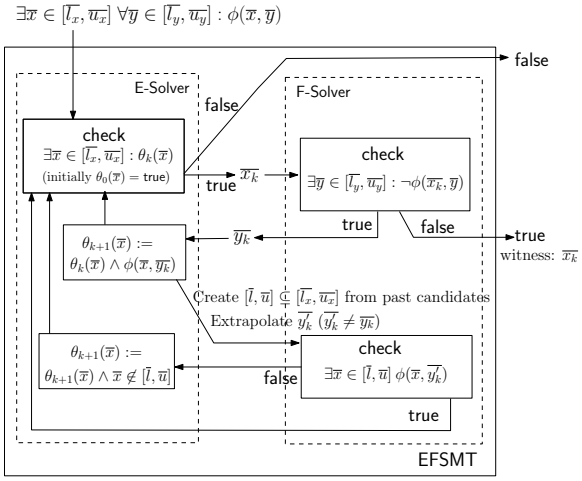

Figure 1: Solving EFSMT formulas

variables are existentially quantified. Dually, since $(\exists\overline{x})\varphi(\overline{x})$ is unsatisfiable (modulo theories) iff $(\forall\overline{x})\neg(\varphi(\overline{x}))$ is valid (modulo theories), an SMT solver can also be used decide validity of these formulas; here, $\overline{x}$ denotes a finite vector of variables. In the latter case, an SMT returns a *counterexample* in the form of a variable assignment for the universally quantified variables.

Exists-Forall SMT.

Formulas in the EFSMT fragment of first-order logic are slightly more general in that they allow for one top-level quantifier alternation of the form

$$(\exists\overline{x})\ (\forall\overline{y})\ \varphi(\overline{x}, \overline{y}).$$

The quantified variables are interpreted over their respective theory domains. The following simple EFSMT formula in linear arithmetic, will subsequently be used as a running example.

$$(\exists x \in [-30, 30] \cap \mathsf{Real})\ (\forall y \in [-30, 30] \cap \mathsf{Real})$$
$$(0 < y < 10) \Rightarrow (y - 2x < 7)$$

In fact, since the examples in this paper do not go beyond the exists-forall quantified non-linear real arithmetic constraints, we restrict the following outline of a verification procedure to such an arithmetic fragment. The EFSMT solver described below, however, can easily be extended to work with the rich combination of theories usually considered in SMT solving.

Notice also that, even though Tarski's seminal quantifier elimination procedure shows that the theory of real closed fields, and thus the full first order theory of the real numbers to be decidable, the quest for a high-performant decision procedure for non-linear real arithmetic is still a very active line of research (e.g. [12, 20]. Our current EFSMT implementation [3] is based on a solver for non-linear arithmetic as suggested by Munoz[13]. This verification procedure uses Bernstein polynomials, and it requires a normalization step of the range of variables to the unit interval; consequently, we restrict ourselves to variables interpreted over closed intervals $[l_x,\ u_x]$.

Solving EFSMT.

We outline a solver for EFSMT problems of the form

$$(\exists\overline{x} \in [\overline{l_x}, \overline{u_x}])\ (\forall\overline{y} \in [\overline{l_y}, \overline{u_y}])\ \phi(\overline{x}, \overline{y}).$$

This procedure relies on SMT solvers for deciding the satisfiability

of propositional combinations of constraints. The EFSMT solver in Figure 1 is based on two instances of SMT solvers, the so-called E-solver and F-solver. These two solvers are applied to quantifier-free formulas of different polarities and they are combined by means of a candidate witness and counterexample-guided refinement strategy for the search for witnesses of the existentially-bound variables. In this way, the E-solver generates a sequence of *candidate witnesses* x_k which are passed to the F-solver for checking if indeed the universally-quantified formula holds for one of these candidate witnesses. Whenever these checks fail the search space for candidate witnesses of the E-solver is constrained based on a counterexample y_k obtained by the F-solver.

A straightforward method for solving EFSMT is to guess a variable assignment, say $\overline{x_0}$, and to verify that the sentence $(\forall \overline{y})\phi(\overline{x_0}, \overline{y})$ holds. Now, the F-solver decides validity problems of the form $(\forall \overline{y})\psi(\overline{y})$ by reducing them to the equivalent unsatisfiability problem for $(\exists \overline{y})\neg\psi(\overline{y})$. Consider our running example 1: for the candidate witness $x \mapsto 0$ the F-solver validity problem $(\forall y \in [-30, 30])\,(0 < y < 10) \rightarrow (y < 7)$ fails to hold. Instead of blindly guessing new instantiations, one might use failed proof attempts and counterexamples y_0 provided by the F-solver to restrict the search space for assignments to the existential variables and to guide the selection of new assignments. If the F-solver generates, say, the counterexample $y \mapsto 9$, then $((0 < 9 < 10) \rightarrow (9 - 2x < 7))$, which is equivalent to $x > 1$, is passed to the E-solver. Using this constraint, the search space for the E-solver in the second round is cut in half.

This counterexample-guided verification procedure for EFSMT is illustrated in the upper part of Fig. 1. At the k-th iteration, the E-solver either generates an instance $\overline{x_k}$ for $\overline{x}$ or the procedure returns with false. An $\overline{x_k}$ provided by the E-solver is passed to the F-solver for checking if $(\exists \overline{y} \in [\overline{l_y}, \overline{u_y}])\,\neg\phi(\overline{x_k}, \overline{y})$ holds. In case there is a satisfying assignment $\overline{y_k}$, the F-solver passes the constraint $\phi(\overline{x}, \overline{y_k})$ to the E-solver, for ruling out such $\overline{x}$ as potential witnesses.

Extrapolation.

We describe a technique for accelerating convergence based on failed proof attempts. Consider, for example, the EFSMT formula $(\exists x \in [0, 10])(\forall y \in [0, 10])\; y \geq x$ and the sequence $2, \frac{1}{2}, \frac{1}{8}, \frac{1}{32}, \ldots$ of failed candidate witnesses. In these cases, ϕ may be extended with the constraint $x \notin (l, u]$. This (supposedly) converging sequence of failed witnesses is widened to the interval $(0, \frac{1}{32}]$. Since $(\forall x \in (0, \frac{1}{32}])(\exists y)\; y < x$ is valid, the *extrapolation* step in Figure 1 restricts further search for potential witnesses to values outside the interval $(0, \frac{1}{32}]$. In this example, after extrapolation, 0 is the only remaining candidate witness left. In general, the extrapolation step accelerates convergence by ruling out candidate witnesses $x \in (l, u]$ whenever $(\forall x \in (l, u])(\exists y)\neg(y \geq x)$. Notice that extrapolation is reminiscent to *widening* in abstract interpretation.

Linearization.

Even though the original EFSMT contains non-linear real arithmetic constraints, the formulas passed to the E-solver and the F-solver may, in some cases, be reduced to linear arithmetic constraints in the combination procedure in Figure 1. This is often the case when encoding parameter synthesis problems in EFSMT (cmp. Section 3.2). In particular, non-linear constraints are linearized when every monomial has at most two variables, one of which is existentially and the other universally bound. Consider, for example, the EFSMT constraint $(\exists s, t)(\forall y, z) sy + 2t + tz > 0$, and suppose that E-solving produces, say, the candidate witness $[s \mapsto 3, t \mapsto 4]$. Now, the F-solver processes $(\forall y, z) 3y + 4z + 8 > 0$ which involves only linear arithmetic, produces a counterexample, say, $[y \mapsto 1, z \mapsto -3]$, and the linear arithmetic constraint $s - t > 0$ is pushed to the logical context.

Incompleteness.

The EFSMT procedure in Figure 1 is sound in that it returns true only in case the input sentence holds and false only in cases it does not hold. However, the EFSMT procedure as stated in Figure 1 is incomplete. For the unsatisfiable EFSMT formula

$$(\exists x \in [0, 10] \cap \mathbb{R})\; (\forall y \in [0, 10] \cap \mathbb{R})$$
$$x > 0 \wedge ((y > 0 \wedge y \neq x) \rightarrow y > x)$$

the E-solver might produce the sequence $x_k = 2, \frac{1}{2}, \frac{1}{8}, \frac{1}{32} \ldots$ of potential witnesses, whereas the F-solver might produce the respective counterexamples $y_k = 1, \frac{1}{4}, \frac{1}{16}, \frac{1}{64} \ldots$. Each counterexample y_k shrinks the search space by posing an additional constraint $x < y_k$ to the E-solver, but the added restriction is not sufficient for the procedure to conclude false. In these cases, the procedure in Figure 1 might serves as a preprocessing step for complete, but typically more expensive, arithmetic solving techniques. On the other hand, many real-world CPS and hybrid systems examples rely on correctness up to a certain precision [**?**], effectively ruling the need for infinite sequences as in the example above.

Logical contexts.

SMT solvers such as Yices support *logical contexts*, that is, finite sequences of conjoined contextual constraints, together with operations for dynamically pushing and popping constraints as the basis for efficiently implementing backtracking search. Our EFSMT solver uses this programming interface, thereby avoiding the re-processing of formulas by the F-solver. Likewise, the E-solver pushes the constraints generated by the F-solver.

Partial Assignments.

SMT solvers such as Yices provide partial variable assignments. If a variable x is not in the codomain of such a partial assignment, then every possible interpretation of x in its domain yields a satisfying assignment. In this way, the EFSMT procedure utilizes partial variable assignments of the F-solver for speeding up convergence by further decreasing the search space for candidate witnesses for the E-solver in each iteration. Symbolic counterexamples, such as $7 \leq y < 10$ in our running example, have the potential of further convergence acceleration.

3. CASE STUDIES

We illustrate the expressive power of EFSMT logical framework by reducing a variety of pivotal design problems for cyber-physical systems to this fragment.

3.1 Stability analysis

We describe two template-based reductions for demonstrating, respectively, asympotic and BIBO stability to logical EFSMT queries.

Asympotic Stability.

Lyapunov's second method for demonstrating asymptotic stability — that is, under small disturbances it it possible to move back to the equilibrium point $\overline{x} = 0$ — relies on the existence of a real-valued, non-negative *energy function* V which is, roughly speaking, decreasing in some sphere around the equilibrium point. Checking this Lyapunov criteria for given candidate functions can easily be encoded in EFSMT using an existentially quantified real-valued variable r for the radius of the sphere and a complex-valued uni-

versally quantified variable z for the points of the sphere areound the origin.

$$(\exists r)\,(\forall z)\,(r > 0) \wedge \\ ((0 < |z| < r) \Rightarrow (V(z) > 0 \wedge \dot{V}(z) < 0))$$

If the designer of some dynamical system hyothesizes that there is a polynomial Lyapunov function $p(z) = a_n z^n + \ldots + a_1 z + a_0$ of degreee n with real-valued coefficientsa_i, then we may extend the EFSMT query above to the search of coeffients a_i for this polynomial template p.

$$(\exists a_n, \ldots, a_0, r)\,(\forall z)\,(r > 0) \wedge \\ ((0 < |z| < r) \Rightarrow (p(z) > 0 \wedge \dot{p}(z) < 0))$$

Consider, for example, the scalar non-linear system $\dot{z} = \frac{2}{2+z} - z - 1$ with equilibrium point $z = 0$. Using the hypothesis that their is a polynomial energy function of template $p(z) = az^2$, where a is a real-valued coeffient a, then $\dot{p}(z) = \frac{-2az(z^2+3z)}{(2+z)}$ and the EFSMT query above involves nonlinear arithmetic constraints of the form $2az(z^2+3z)(2+z) \geq 0$. Our EFSMT solver construct the witness $a \mapsto 8$ and $r \mapsto 1$ for this query, and therefore $V(z) = 8z^2$ is a Lyapunov function for the given dynamic system.

Bounded Input Bounded Output.

A *linear time-invariant* (LTI) system with input signal $x(t)$ and output signal $y(t)$ is BIBO stable if there are constants $M, N > 0$ s.t. if $|x(t)| \leq M$ for all $t \geq 0$ then $|y(t)| \leq N$ for all $t \geq 0$. It is well-known that LTI systems are BIBO stable iff the denominator $X(s)$ of the associated transfer function is a *Hurwitz polynomial*; that is, $Re(z) < 0$ for all poles z of $X(s)$.

Consider, for example, the design of a simple cruise control for maintaining the speed of a vehicle of mass m to the reference speed v_r on slopes θ by providing appropriate control forces u. The dynamics of this system is described as follows.

$$m\dot{\delta} = -k_p\delta - k_i \int_0^t \delta(\tau)d\tau - mg\theta - b(v_r + \delta) \qquad (1)$$

where the signal δ represents the error between the actual speed and the reference speed v_r. For simplicity we set the friction b to 0 and consider small angles θ only such that $\sin\theta \approx \theta$; then the transfer function of this simplified LTI is given by

$$\frac{\Delta(s)}{\Theta(s)} = \frac{-mgs}{ms^2 + k_p s + k_i}$$

Now, the design engineer chooses a *proportional-integral* (PI) controller template with two parameters k_p, k_i; that is

$$u = k_p(v_r - v) + k_i \int_0^t (v_r - v(\tau))d\tau$$

and the design challenge is to select control parameters k_p and k_i for making the cruise control system BIBO stable, where the mass of the vehicle is unknown but in a certain range. This design challenge about BIBO stability is immediately reduced to the following logical EFSMT query:

$$(\exists k_p, k_i)(\forall m, x, y) \\ m(x+iy)^2 + k_p(x+iy) + k_i = 0 \Rightarrow (x < 0)$$

Notice that the resulting PI control is BIBO stable for varying masses of the automobil. Further simplifications in the EFSMT encodings are often possible by using the Routh-Hurwitz and other well-known criteria.

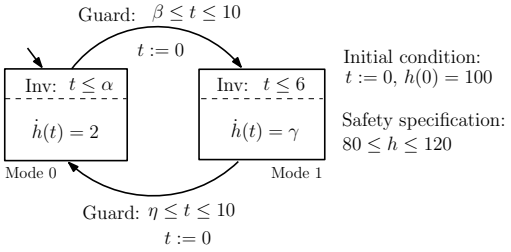

Figure 2: Hybrid Systems Temperature Control.

3.2 Parameter Synthesis for Hybrid Control Systems

Hybrid systems exhibit mixed discrete-continuous dynamic behavior in that they might *jump* as described by a control path or *flow* as prescribed by differential equations. Hybrid systems have been used to model physical systems with impact, logic-dynamic controllers, and even Internet congestion. The discrete dynamic behavior of a hybrid system is often used for modelling different control modes of a CPS, where characteristic flow dynamics of the physical habitat are associated with for each of these nodes. For example, a flight control system typically has different modes for climbing, cruising, and landing.

Consider, for example, the simple temperature control system as depicted in Figure 2.

This hybrid system consists of the two modes 0 and 1. A state $(m,\ t,\ h)$ of this hybrid system is determined by the discrete control mode $m \in \{0, 1\}$ and the values of the real-valued variables $t \in \mathsf{Real}^{\geq 0}$ and $h \in \mathsf{Real}$. The clock t measures the time elapsed since the last clock reset $t := 0$, and h measures the current temperature. Therefore, states of this hybrid systems are encoded by means of the triple (m, t, h), where

While in mode 0 the flow dynamics of the heat $h(t)$ is determined by the differential equation $\dot{h}(t) = 2$, whereas in mode 1 the heat variable flows according to $\dot{h}(t) = \gamma$, with γ an unknown parameter. Similarly to timed automata, the clock t implicitly flows according to $\dot{t} = 1$ in both modes. Continuous flow is permitted as long as the so-called mode invariants hold, while discrete transitions can occur as soon as the given jump conditions or guards are satisfied. In the case of our simple temperature control, continous flow in mode 0 is permitted as long as $t \leq \alpha$, with α an unknown parameter. Notice also that the two transition guards of our simple temperature control includes unknown parameters β η.

The design challenge we are addressing here is to find instantiations for the real-valued parameters α, β, γ, η such that $80 \leq h \leq 120$ is invariant. We are now demonstrating how to reduce this parameterized control problem to a logical problem in EFSMT. In a first step, the predicate $I(m, t, h) := m = 0 \wedge t = 0 \wedge h = 100$ logically encodes the singleton set of initial states. Then, the jump and flow transitions between a state $s := (m, t, h)$ and possible successor states $s' := (m', t', h')$ is encoded as a logical relation $\Delta(\alpha, \beta, \gamma, \eta)(s, s')$ as follows.

$$\begin{aligned} & m = 0 \wedge t \leq \alpha \wedge \beta \leq t \leq 10 \wedge m' = 1 \wedge t' = 0 \wedge h' = h \\ \oplus\ & m = 1 \wedge t \leq 6 \wedge \eta \leq t \leq 10 \wedge m' = 0 \wedge t' = 0 \wedge h' = h \\ \oplus\ & m = m' = 0 \wedge t \leq \alpha \wedge \\ & \quad (\exists \delta \geq 0)\, t' = t + \delta \wedge t' \leq \alpha \wedge h' = h + 2\delta \\ \oplus\ & m = m' = 1 \wedge t \leq 6 \wedge \\ & \quad (\exists \delta \geq 0)\, t' = t + \delta \wedge t' \leq 6 \wedge h' = h + \gamma\delta \end{aligned}$$

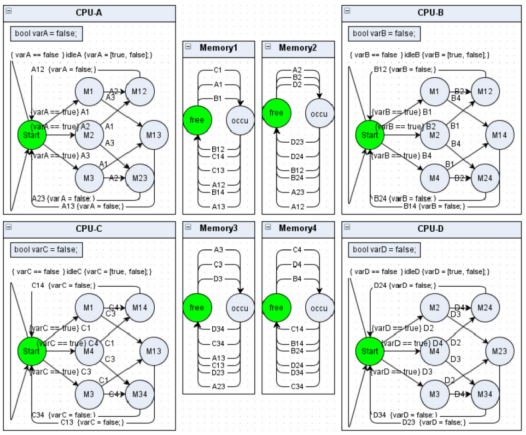

Figure 3: 3D Image Multiprocessing Scenario.

The use of exclusive disjunction operator $\oplus$ ensures the interleaving semantics, whereas the non-deterministic choices for time elapses δ of the flow dynamics in each mode is modelled by means of existential quantification; these existential quantifiers can easily be eliminated by solving for δ.

$$\begin{aligned} & (\exists \delta \geq 0)\ t' = t + \delta \wedge t' \leq \alpha \wedge h' = h + 2\delta \\ \Leftrightarrow \quad & t' - t \geq 0 \wedge t' \leq \alpha \wedge h' = h + 2(t' - t) \end{aligned}$$

Additional constraints are needed for ensuring totality of the transition relation; in particular the conjunction of a state invariant with guards of outgoing transitions should be satisfiable. Altogether, for the safety envelop $Safe(\_, h) := 80 \leq h \leq 120$, the parameterized synthesis problem for the simple temperature control in Figure 2 is encoded in terms of an EFSMT formula, where the parameters are existentially quantified.

$$\begin{aligned} & (\exists \alpha, \beta, \gamma, \eta)(\forall s, s') \\ & \quad Safe(s) \wedge \Delta(\alpha, \beta, \gamma, \eta)(s, s') \Rightarrow Safe(s') \end{aligned}$$

Our witness-producing EFSMT solver returns the ground substition for the existentially-bound variables $\alpha \mapsto 10$; $\beta \mapsto 10$; $\eta \mapsto -\frac{0}{3}$; $\gamma \mapsto 6$. Therefore, a safe control strategy is to repeatedly heat with ratio 2 for 10 seconds followed by cooling down with ratio $\frac{10}{3}$ for 6 seconds.

3.3 Orchestration of Interacting Components

CPS usually are constructed from a set of interacting components. The main design challenge here is to *orchestrate components* in order to achieve a common goal and/or to stay within a safety zone. Two robots collaborating on fabricating some work piece must not collide with each other — nor with human workers in the vicinity. More specifically, the multicore scheduling scenario from 3D image processing in Figure 3 motivates the need for orchestrating distributed components. Each of the four processors needs to allocate two out of four memory banks for processing. Clearly, the system may deadlock without further coordination between the individual processor strategies for allocating and releasing memory banks. Therefore, the design challenge is to avoid deadlock by orchestrating the allocation and the release transitions of the two components.

For simplicity, we demonstrate the *priority synthesis* approach for orchestrating interacting compents together with a reduction to EFSMT using the simple component system in Figure 4. Each of

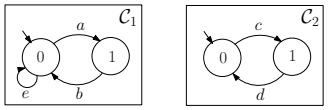

Figure 4: A simple component system.

the two components $\mathcal{C}_1$ and $\mathcal{C}_2$ has transitions a and c for allocating some shared ressource among the components, and transitions b and d for releasing this resource. Since usage of the shared resource is considered to be exclusive, the ressource allocation and release transitions of these need to be orchestrated such that the two components never simultaneously are in state 1. In addition, the orchestration of components should not introduce any possible deadlocks to the system.

Orchestration restricts the firing of transitions by means of *priorities* of the form $\alpha \prec \beta$. If both transitions α and β are enabled then this priority effectively blocks the firing of transition α, and β may be fired as long as there are no other blocking priorities for β. Generally speaking, given a set of interacting components and a set of safe states, the problem of *priority synthesis* is to find an irreflexive and transitive set of priorities such that the restricted system does not introduce any deadlocks and it does not reach unsafe states.

A set of priorities may be viewed as a controller for selecting an adequate, with respect to the given task, transition among a set of enabled transitions. Priorities are expressive in that many well-known scheduling strategies can be encoded in terms of priorities on interactions [9]. On the other hand, priority synthesis is a restricted form of *controller synthesis*, since there are control problems for which there is no solution in terms of priority synthesis. From an algorithmic point of view, the restricted expressiveness of priority synthesis has the advantage that this problem is NP-complete even when considering the more general case of *distributed priority synthesis* for synthesizing distributed conrols [4]. In constrast, distributed controller synthesis problem is undecidable [15].

The reduction of priority synthesis problems to corresponding EFSMT queries is illustrated by means of the simple example in Figure 4. In a first step we define the candidate set $P := \{\underline{\alpha \prec \beta} \,|\, \alpha \neq \beta,\ \alpha, \beta \in \{a, b, c, d, e\}\}$ of propositional variables for each possible priority among the transitions. The variables in P are the existentially-quantified variables in our EFSMT problem, and the solution $\{\alpha \prec \beta \,|\, \sigma(\underline{\alpha \prec \beta}) = true\}$ of the priority synthesis problem is determined by the witness σ as constructed by an EFSMT solver.

A state of the system in Figure 4 is simply a pair (s_1, s_2) with $s_i \in \{0, 1\}$, and the initial predicate is defined as $Init(s_1, s_2) := s_1 = 0 \wedge s_2 = 0$. The logical transition relation $\Delta(P)$ between a current state $s := (s_1, s_2)$ and a successor state $s' := (s'_1, s'_2)$ is the (exclusive, because of non-interleaving semantics) disjunction of the logical transition relations $\Delta_i(P)$ for each component $\mathcal{C}_i$. First, the logical transition relation $\Delta_1(P)(s, s')$ for the transitions a, b, and e is defined as

$$\begin{aligned} & s_1 = 0 \wedge \neg(Blocked(a)) \wedge s'_1 = 1 \wedge s'_2 = s_2 \\ \oplus \quad & s_1 = 1 \wedge \neg(Blocked(b)) \wedge s'_1 = 1 \wedge s'_2 = s_2 \\ \oplus \quad & s_1 = 0 \wedge \neg(Blocked(e)) \wedge s'_1 = 0 \wedge s'_2 = s_2. \end{aligned}$$

The blocking effect of priorities are encoded by means of conjoining transition guards with the blocking predicate

$$Blocked(\alpha) := \bigvee_{\{\beta \,|\, \alpha \neq \beta\}} \underline{\alpha \prec \beta},$$

where meta-variables α, β range over a finite number of transition labels. Second, the logical transition relation $\Delta_2(P)(s, s')$ for component $\mathcal{C}_2$ is defined as

$$\begin{aligned} & s_2 = 0 \wedge \neg(\mathit{Blocked}(c)) \wedge s_1' = s_1 \wedge s_2' = 1 \\ \oplus \quad & s_2 = 1 \wedge \neg(\mathit{Blocked}(d)) \wedge s_1' = s_1 \wedge s_2' = 0. \end{aligned}$$

Now, the safe states $\mathit{Safe}(s_1, s_2)$ are characterized the constraints $\neg(s_1 = 1 \wedge s_2 = 1)$ and deadlock states $DL(s_1, s_2)$ are simply encoded by means of the disjunction of $\neg((\exists s_i' \neq s_i)\Delta_i(P)(s_i, s_i'))$ for the two components i. Altogether, the priority synthesis problem for the example in Figure 4 is encoded in terms of the following EFSMT query:

$$\begin{aligned} & (\exists P)(\forall s, s')\ \mathit{Trans}(P) \wedge \\ & \quad \mathit{Safe}(s) \wedge \neg(DL(s)) \wedge \Delta(P)(s,\ s') \\ & \quad\quad \Rightarrow \mathit{Safe}(s') \wedge \neg(DL(s')). \end{aligned}$$

Solutions are irreflexive due to the construction of the candidate set P and the constraints $\mathit{Trans}(P)$ encodes the transivitiy requirement on valid solutions of priority synthesis problems; in particular, $\mathit{Trans}(P)$ is defined as $\bigwedge_{\alpha,\beta,\gamma} \underline{\alpha \prec \beta} \wedge \underline{\beta \prec \gamma} \Rightarrow \underline{\alpha \prec \gamma}$, where the meta-variables α, β, and γ range over finitely many transition labels. Given this query our EFSMT solver constructs the solution $\{a \prec d, c \prec b\}$ for solving the orchestration problem in Figure 4.

4. CONCLUSION

The developments in this paper undergird our hypothesis that EFSMT indeed is an adequate logical framework for the design of CPS. In particular, we have demonstrated that a large variety of CPS design problems are expressible in EFSMT, thereby reducing these CPS design problems to logical verification problems. In the case of stability analysis, textbook results from analysis and control theory have been incorporated into the EFSMT logical encodings. We have also outlined an effective verification procedure, which is based on a novel interplay of established SMT solvers, for automatically solving these design problems. This procedure builds on a number of features found in modern SMT solvers such as Yices or Z3. It remains to be seen, however, how the — nowadays almost ubiquitious — backjumping search of SMT compares with other search procedures such as MCMC in the E-solver engine.

Clearly, the logical framework ideas presented here are just a first tiny step towards realizing the vision of a coherent, logic-based engineering framework for CPS. In particular, we have so far restricted ourselves to developments on the mathematical design level, as we have not considered any artefacts of real-world engineering such as fault-tolerance or aspects of the deployment on computational platforms with bounded ressources. These require, among many others, topological and geometric reasoning, thermal and energy management, memory, processor, and I/O amanagement, electromagnetic compatibility, and combinations thereof. Specialized theory solvers for these additional CPS design tasks, however, may be integrated through open combination frameworks for theory solvers [17].

Moreover, real-world CPS engineering requires the synthesis of designs which may be Pareto-optimal with respect to, say energy consumption, temperature, processing time, memory consumption, weight, cost of production, or maintainability. These kind of optimization problems might be expressed and solved on the basis of the exists-forall-exists formula

$$(\exists \overline{x}^*)(\forall \overline{y})\varphi(\overline{x}^*, \overline{y}) \wedge \neg((\exists \overline{x})(\forall \overline{y}')\varphi(\overline{x}, \overline{y}') \wedge M(\overline{x}) \prec M(\overline{x}^*)),$$

where the measurement vector $M(\overline{x})$ strictly dominates the optimum $M(\overline{x}^*)$, i.e. at least one measurement of $\overline{x}$ is strictly smaller than the corresponding one for $\overline{x}^*$ and none of measurements of $\overline{x}$ are strictly larger than those of $\overline{x}^*$.

Finally, CPS are evolving over time as they constantly adapt to ever-changing requirements, goals, commands, and habitats. An EFSMT solver in this setting might therefore be even an integral part of such a CPS, and it should therefore be able to incrementally and efficiently construct and adjust solutions to changing conditions, possibly by means of any-time algorithms for solving optimization problems. Because of the underlying complexity one might need to focus on approximate or probabilistic solutions.

Acknowledgements. Chih-Hong Cheng conducted this research while working at fortiss and during a research visit at SRI International. This work is supported by the Distributed MILS (`d-mils.org`) project of the EC's FP7 programme, and the HACMS project funded by the IIO of DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

5. REFERENCES

[1] C.-H. Cheng, S. Bensalem, B. Jobstmann, R.-J. Yan, and H. Ruess. Model construction and priority synthesis for simple interaction systems. In: *NFM'11*.

[2] C.-H. Cheng, H. Ruess, and N. Shankar. JBernstein: A validity checker for generalized polynomial constraints. In: *CAV'13*.

[3] C.-H. Cheng, N. Shankar, H. Ruess, and S. Bensalem. EFSMT: A Logical Framework for Cyber-Physical Systems (full version). In: *CoRR abs/1306.3456 (2013)*.

[4] C.H. Cheng, R.Yan, S. Bensalem, H. Ruess. Distributed Priority Synthesis arXiv preprint arXiv:1211.6189, 2012.

[5] D. Detlefs, G. Nelson, and J.B. Saxe. Simplify: a theorem prover for program checking. *Journal of the ACM (JACM)*, 52.3 (2005): 365-473.

[6] L. De Moura, S. Owre, H. Ruess, J. Rushby, and N. Shankar. The ICS decision procedures for embedded deduction. In *Automated Reasoning*, pp. 218-222, Springer, 2004.

[7] L. De Moura, H. Rueß, and M. Sorea. Lazy Theorem Proving for Bounded Model Checking over Infinite Domains In: *CADE'02*.

[8] B. Dutertre. The Yices 2 SMT solver. In *Computer-Aided Verification* (CAV 2014), pp. 737–744, Springer, 2014.

[9] G. Gössler, J. Sifakis, J. Priority systems. In: *Formal Methods for Components and Objects*. Springer, pp. 314-329, 2004.

[10] S. Jha, S. Gulwani, S.A. Seshia, A. Tiwari. Oracle-guided component-based program synthesis. In: *ACM/IEEE 32nd Intern. Conf. on Software Engineering*. Vol. 1, pp. 215-224, 2010.

[11] S. Gulwani, S. Jha, A. Tiwari, R. Venkatesan. Synthesis of loop-free programs. In: *ACM SIGPLAN Notices*, Vol. 46, No. 6, pp. 62–73, 2011.

[12] D. Jovanović, L. de Moura. Solving non-linear arithmetic. In *Automated Reasoning*, pp. 339–354, Springer, 2012.

[13] C. Muñoz and A. Narkawicz. Formalization of a representation of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 2012.

[14] A. Platzer. Differential dynamic logic for hybrid systems. In *Journal of Automated Reasoning, 41 (2)*. pp. 143–189, 2008.

[15] A. Pnueli, R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. pp. 179-190, ACM, 1989.

[16] D. P. Ranjan, D. Tang, and S. Malik A Comparative Study of 2QBF Algorithms. In *SAT*, 2004.

[17] N. Shankar. Inference systems for logical algorithms. In *Foundations of Software Technology and Theoretical Computer Science*, pp. 60–78, Springer, 2005.

[18] M. Sorea. Bounded Model Checking for Timed Automata, In: *ENTCS*, 2002, Vol. 68(5),

[19] A. Solar-Lezama. Program synthesis by sketching. ProQuest, 2008.

[20] A. Tiwari, P. Lincoln, A Nonlinear Real Arithmetic Fragment. In International Conference on Computed-Aided Verification (CAV 2014), Springer, LNCS, 2014.