Optimizing Assume-Guarantee Contracts for Cyber-Physical System Design

Chanwook Oh[1], Eunsuk Kang[2], Shinichi Shiraishi[3], Pierluigi Nuzzo[1]
[1]Department of Electrical Engineering, University of Southern California, Los Angeles, Email: {chanwooo, nuzzo}@usc.edu
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, Email: eskang@cmu.edu
[3] Toyota InfoTechnology Center, Mountain View, Email: sshiraishi@us.toyota-itc.com

***Abstract*—Assume-guarantee (A/G) contracts are mathematical models enabling modular and hierarchical design and verification of complex systems by rigorous decomposition of system-level specifications into component-level specifications. Existing A/G contract frameworks, however, are not designed to effectively capture the behaviors of cyber-physical systems where multiple agents aim to maximize one or more *objectives*, and may interact with each other and the environment in a *cooperative* or *non-cooperative* way toward achieving their goals. We propose an extension of the A/G contract framework, namely *optimizing A/G contracts*, that can be used to specify and reason about properties of component interactions that involve optimizing objectives. The proposed framework includes methods for constructing new contracts via conjunction and composition, along with algorithms to verify system properties via contract refinement. We illustrate its effectiveness on a set of case studies from connected and autonomous vehicles.**

I. Introduction

Assume-guarantee (A/G) contracts are mathematical models enabling modular and hierarchical design of complex systems by rigorous decomposition of system-level requirements into component-level specifications [1]–[7]. Informally, a contract $C = (A, G)$ specifies the behavior of a component in terms of its expectation on the behavior of the environment (the *assumptions* A) and the promises that it makes (the *guarantees* G) in the context of its environment. If the assumptions are satisfied by the environment, any component that *implements* the contract will satisfy its guarantees.

Different A/G contract frameworks have been proposed over the years to support a variety of models of computation (e.g., synchronous, dataflow, or timed models) for embedded system design [1]. However, the development of contract frameworks that are rich enough to support cyber-physical system (CPS) design is an active research area. In this paper, we consider components that, in addition to satisfying their guarantees, may also attempt to produce behaviors that are considered "preferable" with respect to a certain goal or *objective*. Performance is a common example of such an objective; for instance, beside a safety requirement that a minimum distance to the neighboring vehicles must be maintained, an autonomous vehicle may have a performance objective of minimizing the time to destination or the fuel consumption. Unlike the guarantees, which a component must be designed to satisfy under all valid environments, an objective does not mandate the component to always behave in the optimal manner; nevertheless, it may play a crucial role in shaping its behaviors. Existing contract-based frameworks, however, do not provide a method to capture such a notion of objectives.

This paper proposes an extension of the A/G contracts framework termed *optimizing A/G contracts*. Along with A and G, an optimizing contract includes an *objective* O that captures how desirable are certain behaviors over the others. In particular, we define $O : G \to \mathbb{R}^n$ as an *objective function* that ranks each valid output behavior of a component with a (possibly multi-dimensional) quality metric. Building on this notion of contract, we propose a mathematical framework for the design of CPSs with objectives, including methods for constructing new contracts via conjunction and composition, and for verifying system properties out of component properties via contract refinement.

Some of the components in a system may have *conflicting* objectives, in that optimizing an objective for a certain component results in a sub-optimal behavior for another component. When these components are composed, potential conflicts may arise; our framework provides an approach to capture or resolve such conflicts by formulating this problem as a multi-objective optimization problem. In particular, the proposed framework provides the notion of *cooperative* composition to model system interactions where some of the components are designed to cooperate with each other to produce optimal behaviors for the overall system.

By developing a framework of optimizing contracts, we also establish a link between contract-based design and optimal control of multi-agent systems. *Optimal control* is the long-studied problem of synthesizing a controller that attempts to minimize a given cost function while satisfying a set of system constraints [8]. Among prior works, the ones that are closest to ours are approaches that employ a formal logic, e.g., linear temporal logic (LTL) or signal temporal logic (STL) [9], to specify the desired behavior of the system while producing an optimal control policy [10]–[15]. Unlike these works, which mostly deal with the control of a single system or component, our work provides a formal framework to enable compositional reasoning about the behaviors of multiple components that may have their own distinct objectives to be optimized. Moreover, our notion of objective is amenable to encapsulate other quantitative notions of robust satisfaction of logic formulas. We illustrate the effectiveness of the proposed framework on case studies from connected and autonomous vehicles, including the verification and synthesis of cooperative and non-cooperative behaviors from contracts expressed in signal temporal logic.

II. Motivating Example

Consider two autonomous vehicles at a T-junction where only one car can pass at a time. Each vehicle can either `Pass` or `Wait`. If both vehicles decide to pass through the T-junction at the same time, they will collide. On the other hand, if both decide to wait, they may enter a deadlock state in which they wait indefinitely. Ideally, we would like one vehicle to wait and the other to pass, so that they can traverse the T-junction one at a time. We model the preference between system behaviors by assigning a reward (or penalty) to each combination of actions of the two vehicles, (u_1, u_2), where u_1 is the action of vehicle 1 (M_1) and u_2 is that of vehicle 2 (M_2). The objectives of

TABLE I: M_1 Rewards

1 \ 2	Wait	Pass
Wait	-1	-1
Pass	3	-10

TABLE II: M_2 Rewards

1 \ 2	Wait	Pass
Wait	-1	4
Pass	-1	-10

M_1 and M_2 are to maximize the rewards associated with their actions, shown in Tab. I and II, respectively. Rewards are as low as -10 to penalize collision and take on a maximum value when a vehicle manages to traverse the junction.

From the perspective of M_1, the behavior that results in the optimal reward is $(u_1,u_2) = (\texttt{Pass},\texttt{Wait})$. However, if M_1 is not aware of the next action or associated reward of M_2, it will instead opt for `Wait` since it must act conservatively to guarantee safety (i.e., no collision) *for all* the possible actions of M_2. If M_2 selects its action in the same manner as M_1 (i.e., maximizes its own reward), the two vehicles will enter the deadlock state. On the other hand, if the vehicles were able to communicate their actions and rewards to each other, they may be able to avoid deadlock by making decisions in a cooperative manner.

We would like to formally reason about the overall behavior of this system using A/G contracts, where each vehicle is modeled as a component implementing a contract, and the composition of M_1 and M_2 represents the end-to-end system. To do so, we seek for a framework that (1) captures the behaviors of a component while optimizing its objective, and (2) supports a composition mechanism that takes into account the notion of cooperation and non-cooperation. In the following, we first provide an overview of the current A/G contract framework and then describe how we extend it to address these issues.

III. Assume-Guarantee Contracts

We provide an overview of the A/G contract framework [1] by starting with the notion of component. A component M is an element of a design, characterized by a set of *variables* V, a set of *ports* P, and a set of behaviors $[[M]]$ over variables and ports. Components can be connected to form larger systems by sharing certain ports under constraints on the values of certain variables. For simplicity, we use the term "variables" to denote both component variables and ports.

An A/G contract $C = (V,A,G)$ is a triple where V denotes the set of component variables, A denotes the *assumptions*, i.e., the set of behaviors that a component M expects from the environment, and G denotes the *guarantees*, i.e., the behaviors promised by M if the assumptions hold. A component M is a (valid) *implementation* of a contract C when $A \cap [[M]] \subseteq G$; we then say that M *implements* C and write $M \models C$. A component E is a (valid) *environment* of C when $[[E]] \subseteq A$; we then write $E \models_E C$. A contract $C = (V,A,G)$ is in *saturated form* if $G = G \cup \overline{A}$ where $\overline{A}$ is the complement of A. A contract $C = (V,A,G)$ can be *saturated* by turning it into $C' = (V,A',G')$ where $A' = A$ and $G' = G \cup \overline{A}$. C and C' are equivalent, since they are characterized by the same sets of environments and implementations. In the remainder of the paper, unless otherwise specified, we assume that all contracts are saturated, as saturated forms will be used to define the main contract operations and relations.

A contract C is *compatible* if there exists a valid environment E for M, i.e., if and only if $A \neq \emptyset$, where $\emptyset$ is the empty set. The intent is that an implementation M can only be used under a *compatible* environment. C is *consistent* if there exists a feasible implementation M for it. For a saturated contract, this is equivalent to $G \neq \emptyset$. To reason about different abstraction layers in a design, contracts can be ordered according to a *refinement* relation. A contract $C = (V,A,G)$ refines $C' = (V,A',G')$, written $C \preceq C'$, if and only if $A \supseteq A'$ and $G \subseteq G'$. Contract refinement amounts to weakening the assumptions and strengthening the guarantees. Thus, if $M \models C$ and $C \preceq C'$, then $M \models C'$. On the other hand, if $E \models_E C'$, then $E \models_E C$. We can then replace C' with C in the design.

Contracts can be combined into a larger contract via *conjunction* or *composition*. *Conjunction ($\wedge$)* is used to combine multiple requirements on the same component, such that all the requirements can be simultaneously satisfied. If a component M is an implementation of $C_1 \wedge C_2$, i.e., $M \models C_1 \wedge C_2$, then both $M \models C_1$ and $M \models C_2$ hold. *Composition ($\otimes$)* is used to express complex system-level requirements by combining simpler component-level requirements. Given C_1 and C_2 and their implementations M_1 and M_2, respectively, the behavior of the system composed by M_1 and M_2 can then be analyzed using $C_1 \otimes C_2$. We refer the reader to the literature [1] for details. We will recall the mathematical expressions of conjunction and composition below, as we introduce the proposed A/G contract extensions.

IV. Optimizing A/G Contracts

We start by defining *optimizing contracts* and the related notions of *compatibility* and *consistency*.

Definition IV.1 (Optimizing Contract)**.** An *optimizing contract* C is a tuple $C = (U,X,A,G,O)$, where

- U is a set of *uncontrolled (input) variables* and X is a set of *controlled (output) variables*, with $U \cap X = \emptyset$ and $V = U \cup X$;
- A and G are sets of behaviors over V representing the *assumptions* and the *guarantees*, respectively,
- $O : G \to \mathbb{R}^n$ is an n-dimensional *objective function* mapping behaviors in G to n-tuples of rewards.

We also denote by u, x, and $v = (u^T, x^T)^T$ a behavior over U, X, and V, respectively.

Definition IV.2 (Optimal Guarantees)**.** For an optimizing contract $C = (U,X,A,G,O)$, we define the set of *optimal guarantees* $G_{\max} = \arg\max_x \min_u O(v)$ s.t. $v \in A \cap G$.

Definition IV.2 amounts to stating that a component M implementing C assigns values to its controlled variables to maximize its reward, but has no information about the environment E and its objective, except that E satisfies the assumptions. The optimal guarantees are then computed in the worst-case scenario that the uncontrolled variables are assigned to minimize O.

Definition IV.3 (Compatibility and Consistency)**.** An optimizing contract $C = (U,X,A,G,O)$ with optimal guarantees $G_{\max}$ is *compatible* if and only if $A \neq \emptyset$ and *consistent* if and only if $G_{\max} \neq \emptyset$.

Example IV.1 (Optimizing Contracts)**.** Consider a contract $C = (\{U\},\{X\},1 \leq u \leq 2,\neg(1 \leq u \leq 2) \vee (0 \leq x \leq 4),u+x)$ where assumptions and guarantees are expressed by predicates on the real variables u and x. We obtain $G_{\max} =$

$(\arg\max_x \min_u (u+x)$ s.t. $(1 \le u \le 2) \wedge (0 \le x \le 4)) = (1,4)^T$. Since $A \neq \emptyset$ and $G_{\max} \neq \emptyset$, the contract C is both compatible and consistent.

We now extend the *conjunction* operation and define the new operations of *cooperative* and *non-cooperative composition* to encapsulate conflicting objectives of components.

Definition IV.4 (Conjunction)**.** Given the optimizing contracts $C_1 = (U, X, A_1, G_1, O_1)$ and $C_2 = (U, X, A_2, G_2, O_2)$, the *conjunction* of C_1 and C_2, denoted as $C_1 \wedge C_2$, is an optimizing contract on the same variable sets U and X and such that:

$$A = A_1 \cup A_2 \tag{1}$$
$$G = G_1 \cap G_2 \tag{2}$$
$$O = (O_1^T, O_2^T)^T. \tag{3}$$

Assumptions and guarantees in (1) and (2) are computed as in the traditional A/G contract framework [1]. O in (3) is obtained by conjoining the objectives of C_1 and C_2. $G_{\max}$ can then be computed as in Def. IV.2.

Computing the optimal guarantees results, in general, into a multi-objective optimization problem [16], which can be addressed by either constructing the Pareto optimal fronts to select a Pareto optimal solution, or by reducing the original problem to a standard, single objective optimization problem. For simplicity, we follow the latter approach in the examples of this paper. Specifically, we assume that a weight vector $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ is provided, which encodes, for example, the priority given by the designer (or the agents in the system) to the components of O. We then generate a single objective function as the *weighted sum* of all the components of O according to w, i.e., $O^w = \sum_{i=1}^n w_i O_i$. In the following, without loss of generality, we further assume $w = (1, \ldots, 1)$.

Example IV.2 (Conjunction)**.** We compute the conjunction of the following saturated contracts:

$$C_1 = (\{U\}, \{X\}, 0 \le u \le 3, (2 \le x \le 3) \vee \neg(0 \le u \le 3), u - x)$$
$$C_2 = (\{U\}, \{X\}, 0 \le u \le 2, (1 \le x \le 4) \vee \neg(0 \le u \le 2), 3x)$$

We obtain $C_1 \wedge C_2 = (\{U\}, \{X\}, 0 \le u \le 3, (2 \le x \le 3) \vee \neg(0 \le u \le 3), (u - x, 3x)^T))$ and $G_{1\wedge 2,\max} = \arg\max_x \min_u (u + 2x)$ s.t. $(0 \le u \le 3) \wedge (2 \le x \le 3) = (0,3)^T$.

Definition IV.5 (Cooperative Composition)**.** The *cooperative composition* of $C_1 = (U_1, X_1, A_1, G_1, O_1)$ and $C_2 = (U_2, X_2, A_2, G_2, O_2)$, denoted by $C_1 \hat{\otimes} C_2$, is an optimizing contract (U, X, A, G, O) such that

$$U = U_1 \cup U_2 \setminus X, \qquad X = X_1 \cup X_2 \tag{4}$$
$$A = (A_1 \cap A_2) \cup \overline{G}, \qquad G = G_1 \cap G_2 \tag{5}$$
$$O = (O_1^T, O_2^T)^T \tag{6}$$

and the optimal guarantees $G_{\max}$ can be computed as in Def. IV.2.

Definition IV.6 (Non-cooperative Composition)**.** We call *non-cooperative composition* of $C_1 = (U_1, X_1, A_1, G_1, O_1)$ and $C_2 = (U_2, X_2, A_2, G_2, O_2)$ the contract $C_1 \check{\otimes} C_2 = (U, X, A, G)$ such that U, X, A, G are defined as in Def. IV.5 and $G_{\max} = G_{1,\max} \cap G_{2,\max}$, where: (1) $G_{1,\max}$ are the optimal guarantees for the optimizing contract $\check{C}_1 = (U \cup Y_1, X \setminus Y_1, A, G, O_1)$, with $Y_1 = U_1 \cap X_2$; (2) $G_{2,\max}$ are the optimal guarantees for the optimizing contract $\check{C}_2 = (U \cup Y_2, X \setminus Y_2, A, G, O_2)$, with $Y_2 = U_2 \cap X_1$. $G_{1,\max}$ and $G_{2,\max}$ are separately computed as in Def. IV.2.

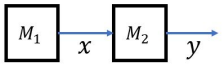


Fig. 1: Connection of Components M_1 and M_2

Cooperative composition reduces to a single optimizing contract because components are allowed to share (conjoin) their objectives and make decisions towards optimizing them. On the other hand, non-cooperative composition cannot be reduced to a single optimizing contract. Its behaviors are, instead, described by the two optimizing contracts $\check{C}_1$ and $\check{C}_2$. In non-cooperative composition, components may share some of their variables but have no information about each others' objective. Consequently, two separate optimization problems are solved to find the optimal guarantees of each contracts. Because each component makes worst-case assumptions about its environment, non-cooperative composition usually results into a sub-optimal solution from the overall system perspective.

Example IV.3 (Composition)**.** Consider a component M_1 implementing $C_1 = (\emptyset, \{X\}, \top, 0 \le x \le 3, -x)$ and a component M_2 implementing $C_2 = (\{X\}, \{Y\}, 0 \le x \le 4, 0 \le y \le 2, 2x + y)$, connected as shown in Fig. 1. We use $\top$ to denote the Boolean value `True`. We compute the composition of C_1 and C_2 with and without cooperation. After placing both contracts in saturated form, we obtain $C_1 \hat{\otimes} C_2 = (\emptyset, \{X, Y\}, \top, (0 \le x \le 3) \wedge (0 \le y \le 2), (-x, 2x + y)^T)$ and $G_{1\hat{\otimes}2,\max} = \arg\max_{x,y}(x + y)$ s.t. $(0 \le x \le 3) \wedge (0 \le y \le 2)\} = (3,2)^T$. Since $A_{1\hat{\otimes}2} \neq \emptyset$ and $G_{1\hat{\otimes}2,max} \neq \emptyset$, $C_1 \hat{\otimes} C_2$ is compatible and consistent.

On the other hand, the optimal guarantees for $C_1 \check{\otimes} C_2$ are $G_{1\check{\otimes}2,\max} = G_{1,\max} \cap G_{2,\max} = (0,2)^T$, since $G_{1,\max} = \arg\max_{x,y}(-x)$ s.t. $(0 \le x \le 3) \wedge (0 \le y \le 2) = \{(0,y)^T : 0 \le y \le 2\}$ and $G_{2,\max} = \arg\max_y \min_x (2x + y)$ s.t. $(0 \le x \le 3) \wedge (0 \le y \le 2) = \{(0,2)^T\}$.

A refinement relation between optimizing contracts can be expressed via *optimal refinement*, which enables reasoning about replaceability between components whose behaviors aim to optimize an objective.

Definition IV.7 (Optimal Refinement)**.** Given the optimizing contracts $C_1 = (U, X, A_1, G_1, O_1)$ and $C_2 = (U, X, A_2, G_2, O_2)$ on the same variables, we say that C_2 *optimally refines* C_1, written $C_2 \preceq^* C_1$, if and only if

$$A_2 \supseteq A_1, \qquad G_{2,\max} \subseteq G_{1,\max} \tag{7}$$

We observe that if the objective of an optimizing contract is empty, it reduces to a traditional contract, with $G_{\max} = G$. Accordingly, if both of the objectives of C_1 and C_2 are empty, optimal refinement reduces to the traditional refinement relation.

V. Contract-Based Verification and Synthesis

We define verification and synthesis problems based on the optimizing A/G contract framework introduced above. We express assumptions and guarantees using formulas in bounded signal temporal logic (STL) [9], i.e., STL formulas in which all the temporal operators are interpreted over bounded

time intervals. We refer the reader to the literature for the formal definition of the syntax and the semantics of bounded STL formulas [10], [11], [17]. The assumptions and guarantees of a contract can be defined by the sets of behaviors that satisfy the bounded STL formulas ϕ_A and ϕ_G, respectively. We further consider STL formulas expressing properties of discrete-time systems, and denote system behaviors by x^H, u^H, and v^H, defined over the variables X, U, and V, respectively. Behaviors are discrete-time traces (signals) of length H, which is the time horizon of interest.

Problem 1 (Compatibility and Consistency Checking)**.** Given a system M and an optimizing contract $C = (U, X, \phi_A, \phi_G, O)$ for M over the time horizon H, determine whether C is compatible, i.e., ϕ_A is satisfiable, or consistent, i.e., the following optimization problem is feasible: $\max_{x^H} \min_{u^H} O(v^H)$ s.t. $v^H \models \phi_A \wedge \phi_G$.

Problem 2 (Contract Optimal Refinement Checking)**.** Given a system M and optimizing contracts $C_1 = (U, X, \phi_{A1}, \phi_{G1}, O_1)$ and $C_2 = (U, X, \phi_{A2}, \phi_{G2}, O_2)$ defined over the system variables and a time horizon H, determine whether $C_2 \preceq^* C_1$, i.e., whether both of the following conditions hold: (i) $\phi_{A1} \wedge \neg\phi_{A2}$ is unsatisfiable; (ii) $G_{2,\max} \cap \overline{G_{1,\max}}$ is empty, where $G_{1,\max}$ and $G_{2,\max}$ are defined as follows:

$$G_{1,\max} = \arg\max_{x^H} \min_{u^H} O_1(v^H) \text{ s.t. } v^H \models \phi_{A1} \wedge \phi_{G1} \quad (8)$$

$$G_{2,max} = \arg\max_{x^H} \min_{u^H} O_2(v^H) \text{ s.t. } v^H \models \phi_{A2} \wedge \phi_{G2} \quad (9)$$

Problem 3 (Synthesis from Optimizing Contracts)**.** Given a system M and an optimizing contract $C = (U, X, \phi_A, \phi_G, O)$ for M over the time horizon H, determine a sequence x^H that solves the following optimization problem: $\max_{x^H} \min_{u^H} O(v^H)$ s.t. $v^H \models \phi_A \wedge \phi_G$.

The definition of Problems 1-3 directly descends from the definitions of contract compatibility, consistency, optimal refinement, and optimal guarantees. We introduce below algorithms for the solutions of these problems.

A. Contract-based Verification

We assume that a function SOLVEMAX(M, V, ϕ, O, H) is available, which returns sequences (x^H, u^H) by solving the optimization problem: $\max_{x^H, u^H} O(v^H)$ subject to $v^H \models \phi$. Moreover, we assume that the function SOLVEMAXMIN(M, U, X, ϕ, O, H) returns sequences (x^H, u^H) by solving the optimization problem: $\max_{x^H} \min_{u^H} O(v^H)$ subject to $v^H \models \phi$. Recent work [10], [11], [17] has proposed algorithms based on mixed integer linear programming (MILP) to solve these problems for certain classes of objectives and bounded STL formulas of practical interest. We leverage the encodings and the MILP-based algorithms proposed in the above references to implement SOLVEMAX and SOLVEMAXMIN.

We address compatibility checking by calling SOLVEMAX(M, V, ϕ_A, [], H). If there exists a solution trace, then C is compatible. Similarly, to check consistency, we call SOLVEMAXMIN(M, U, X, $\phi_A \wedge \phi_G$, O, H). If there exists a solution trace, then C is consistent. We can use SOLVEMAXMIN to also find an element of G_{max}.

We check the optimal refinement $C_2 \preceq^* C_1$ by using Algorithm 1. Lines 1-3 check whether $\phi_{A1} \rightarrow \phi_{A2}$ is a valid formula, which amounts to checking that $\phi_{A1} \wedge \neg\phi_{A2}$ is unsatisfiable. If this is not the case, the algorithm immediately returns $C_2 \not\preceq^* C_1$. Lines 4-12 check that $G_{2,\max} \subseteq G_{1,\max}$ based on Def. IV.7. We first find the rewards $O_{1,\max}$ and $O_{2,\max}$ for elements in $G_{1,\max}$ and $G_{2,\max}$, respectively. We then look for a trace v^H that belongs to $\overline{G_{1,\max}} \cap G_{2,\max}$, i.e., a trace such that $O_2(v^H) = O_{2,\max}$ and $O_1(v^H) < O_{1,\max}$. If there exists no such trace satisfying the additional constraints, the algorithm returns $C_2 \preceq^* C_1$; otherwise, we find $C_2 \not\preceq^* C_1$.

Algorithm 1: CHECKOPTREFINE

Input: M, $C_1 = (U, X, \phi_{A1}, \phi_{G1}, O_1)$, $C_2 = (U, X, \phi_{A2}, \phi_{G2}, O_2)$, H
Output: Refine = $\{0, 1\}$

1 $v^H \leftarrow$ SOLVEMAX(M, V, $\phi_{A1} \wedge \neg\phi_{A2}$, [], H)
2 **if** $v^H = \emptyset$ **then**
3 return 0;
4 $v^H \leftarrow$ SOLVEMAXMIN(M, U, X, $\phi_{A1} \wedge \phi_{G1}$, O_1, H)
5 $O_{1,\max} \leftarrow O_1(v^H)$
6 $v^H \leftarrow$ SOLVEMAXMIN(M, U, X, $\phi_{A2} \wedge \phi_{G2}$, O_2, H)
7 $O_{2,\max} \leftarrow O_2(v^H)$
8 $v^H \leftarrow$ SOLVEMAXMIN(M, U, X, $\phi_{A2} \wedge \phi_{G2} \wedge (O_2(v^H) = O_{2,\max}) \wedge (O_1(v^H) < O_{2,max})$, O_2, H)
9 **if** $v^H = \emptyset$ **then**
10 return 1;
11 **else**
12 return 0;

Algorithm 2: NONCOOPSYN

Input: M_1, M_2, H, $\check{C}_1 = (U \cup Y_1, X \setminus Y_1, \phi_A, \phi_G, O_1)$, $\check{C}_2 = (U \cup Y_2, X \setminus Y_2, \phi_A, \phi_G, O_2)$
Output: (v_1^H, v_2^H)

1 $v_1^H \leftarrow$ SOLVEMAXMIN($M_1 \times M_2$, $U \cup Y_1$, $X \setminus Y_1$, $\phi_A \wedge \phi_G$, O_1, H)
2 $v_2^H \leftarrow$ SOLVEMAXMIN($M_1 \times M_2$, $U \cup Y_2$, $X \setminus Y_2$, $\phi_A \wedge \phi_G$, O_2, H)

B. Synthesis from Optimizing Contracts

To solve Problem 3 for a contract $C = (U, X, A, G, O)$, we can directly use SOLVEMAXMIN(M, U, X, $\phi_A \wedge \phi_G$, O, H) to find a trace in $G_{\max}$. However, finding such a trace is less straightforward for a contract that is obtained via cooperative or non-cooperative composition. Let $C_1 = (U_1, X_1, \phi_{A1}, \phi_{G1}, O_1)$ and $C_2 = (U_2, X_2, \phi_{A2}, \phi_{G2}, O_2)$ be contracts specified for the composition between M_1 and M_2, written $M_1 \times M_2$. Because cooperative composition $C_1 \hat{\otimes} C_2 = (U, X, \phi_A, \phi_G, O)$ is an optimizing contract, we can generate a sequence x^H satisfying it by calling SOLVEMAXMIN($M_1 \times M_2$, U, X, $\phi_A \wedge \phi_G$, O, H), where $\phi_G := \phi_{G1} \wedge \phi_{G2}$, $\phi_A := \phi_G \rightarrow (\phi_{A1} \wedge \phi_{A2})$, and $O = (O_1^T, O_2^T)^T$.

Non-cooperative composition $C_1 \check{\otimes} C_2$ requires, instead, the solution of two optimization problems to generate a trace satisfying the composite contracts, as shown in Algorithm 2. We find a trace v_1^H in $G_{1,\max}$ and v_2^H in $G_{2,\max}$ and concatenate them to provide a solution in $G_{1,\max} \cap G_{2,\max}$.

VI. CASE STUDIES

We first show how the scenario proposed in Sec. II can be analyzed using optimizing contracts; then, we discuss two case studies in the context of autonomous vehicles.

A. Motivating Example

We can write an optimizing contract for each vehicle in the example of Sec. II as follows:

$$C_1 = (\{U_2\},\{U_1\},u_2 \in \{\texttt{Wait},\texttt{Pass}\},\\ (u_1 \in \{\texttt{Wait},\texttt{Pass}\}) \wedge ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})),R_1)$$
$$C_2 = (\{U_1\},\{U_2\},u_1 \in \{\texttt{Wait},\texttt{Pass}\},\\ (u_2 \in \{\texttt{Wait},\texttt{Pass}\}) \wedge ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})),R_2)$$

The cooperative and non-cooperative compositions can be computed as follows, where $C_1 \check{\otimes} C_2 = (\check{C}_1,\check{C}_2)$:

$$C_1 \hat{\otimes} C_2 = (\emptyset,\{U_1,U_2\},\top,((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge \\ ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})),R_1+R_2)$$
$$\check{C}_1 = (\{U_2\},\{U_1\},\top,((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge \\ ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})),R_1)$$
$$\check{C}_2 = (\{U_1\},\{U_2\},\top,((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge \\ ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})),R_2)$$

The optimal guarantees with cooperation are $G_{1\hat{\otimes}2,\max} = \arg\max_{u_1,u_2}(R_1+R_2)$ s.t. $((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})) = \{(\texttt{Wait},\texttt{Pass})\}$. The ones without cooperation are $G_{1\check{\otimes}2,\max} = G_{1,\max} \cap G_{2,\max} = \{(\texttt{Wait},\texttt{Wait})\}$ where $G_{1,\max} = \arg\max_{u_1} \min_{u_2} R_1$ s.t. $((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})) = \{(\texttt{Wait},u_2) : u_2 \in \{\texttt{Wait},\texttt{Pass}\}\}$ and $G_{2,max} = \arg\max_{u_2} \min_{u_1} R_2$ s.t. $((u_1,u_2) \in \{\texttt{Wait},\texttt{Pass}\}^2) \wedge ((u_1,u_2) \neq (\texttt{Pass},\texttt{Pass})) = \{(u_1,\texttt{Wait}) : u_1 \in \{\texttt{Wait},\texttt{Pass}\}\}$. With cooperation, the optimal behavior of the composite is to let M_2 `Pass` first while M_1 `Wait`. Without cooperation, the optimal behavior requires for both vehicles to `Wait` for the other vehicle to `Pass`.

B. Automotive Case Studies

We study the control of vehicles on a highway merging and a T-junction scenario with and without cooperation. We encode the satisfaction problem for STL formulas as a MILP problem and use a model predictive control (MPC) scheme [10]. MILP problems are solved using YALMIP [18] and GUROBI [19] on an Intel core i5 with 8-GB RAM.

We consider a closed-loop discrete-time vehicle i, as shown in Fig. 2, and its environment including N interacting vehicles. We denote by $x_{i,k} = (p_{x,i,k}, p_{y,i,k}, v_{x,i,k}, v_{y,i,k})^T$ the vehicle state at step k. The control input $u_{i,k} = (a_{x,i,k}, a_{y,i,k})^T$ is determined by the current vehicle states $x_{i,k}$ and the states of the other vehicles $x_{j,k}$, $j = 1,\ldots,N$, $j \neq i$. $(p_{x,i,k}, p_{y,i,k})$ is the (x,y)-position, $(v_{x,i,k}, v_{y,i,k})$ is the (x,y)-velocity, and $(a_{x,i,k}, a_{y,i,k})$ is the (x,y)-acceleration of a vehicle i at the k^{th} time step. We describe a trace of the system within the time horizon H as $(x_1^H,\ldots,x_N^H)$.

Each vehicle M_i is specified by a contract C_i and controlled using an MPC scheme with the prediction horizon of 2 s and the control horizon of 1 s. We use a time discretization step of 0.1 s and assume that all vehicles within the environment have the same dynamics. Each vehicle can sense the position and the velocity of the surrounding vehicles at each time step.

Physical constraints include formulas such as $\phi_{Pi} = \mathbf{G}_{[0,H]}((|a_i| \leq 5) \wedge (|v_i| \leq 10))$, no collision constraints expressed as $\phi_{NC,(i,j)} = \mathbf{G}_{[0,H]}((|p_{x,i} - p_{x,j}| \geq 2) \wedge (|p_{y,i} - p_{y,j}| \geq 2))$, and path planning objectives as O_i. The value of the path planning objective function $O_i(x_i^H)$ increases as a vehicle approaches the destination and decreases if a vehicle travels off the designated course or collides to another vehicle.

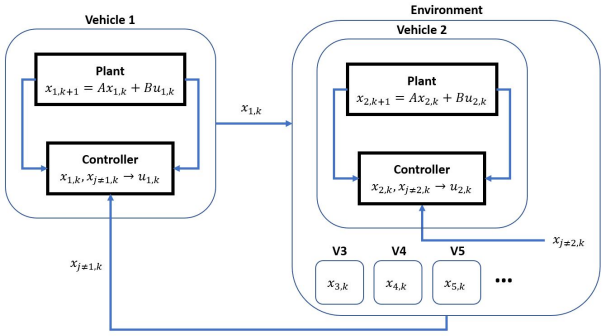


Fig. 2: A closed system with N vehicles

1) 2-D T-junction: We first consider two autonomous vehicles implementing the contract in (10) on a T-junction for 8 s of simulation. Driven by its objective, vehicle 1 (2) starting at the red (blue) dot gains higher reward as it approaches the destination at the red (blue) cross as shown on Fig. 3a. We compose two contracts with and without cooperation and synthesize control trajectories for the composed components.

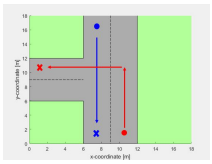
$$C_i = (\{x_{j\neq i}^H\},\{x_i^H\},\bigwedge_{j=1}^{N} \phi_{P_{j\neq i}},\phi_{P_i} \wedge \bigwedge_{j=1}^{N} \phi_{NC,(i,j\neq i)},O_i) \quad (10)$$

2) 2-D Highway Merging: We also consider the control of three autonomous vehicles on a 2-D highway merging scenario for 8 s. Vehicle 1, 2, and 3 implement contracts as in (10). Similarly to the highway merging example, vehicle 1, 2, and 3 (red, blue, green) start from the dot and try to approach the destination cross, as shown in Fig. 4a, driven by their objective functions. O_1, O_2, and O_3 are modified accordingly.

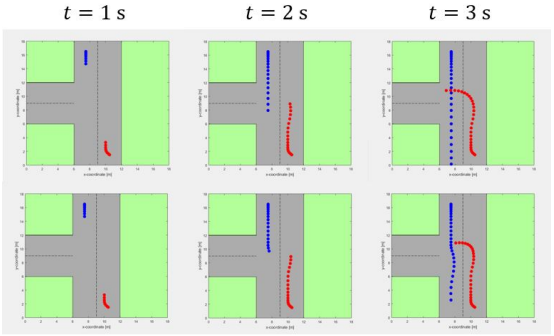
3) Results and Analysis: The synthesized traces of the vehicles for a T-junction and highway merging scenarios are given in Fig. 3 and 4, respectively. In the non-cooperative cases in both scenarios, we observed behaviors where vehicles act conservatively and slow down to avoid a low reward due to a collision. For instance, vehicle 2 (blue) in Fig. 3b at $t = 2$ s is decelerating to maximize its reward against possibly hostile (objective minimizing) behaviors of vehicle 1 (red). Vehicle 1 (red) on Fig. 4b at $t = 2$ s also displays a similar behavior. On the other hand, when there is cooperation, there are less occurrences of such deceleration, since the vehicles are aware of and act to accommodate each others' next maneuver.

Tab. III summarizes the performance of the controller under cooperation and non-cooperation in the two scenarios. We report the arrival time of each vehicle and the sum of the arrival times of both the vehicles. In both the T-junction and highway merging scenarios, cooperating vehicles arrive at the destination earlier than non-cooperating vehicles. We also report the sum of the absolute value of the accelerations (the last four columns), which is directly related to fuel consumption. With cooperation, the value is always lower than that of non-cooperating vehicles, suggesting that the vehicles can reach their destinations more efficiently under co-operation.

The examples in this paper utilize MILP, which is NP-hard. However, for certain fragments of STL the control synthesis



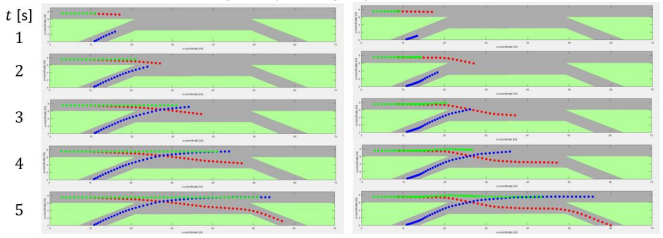
(a) T-junction



(b) With cooperation (Top) and without cooperation (Bottom)

Fig. 3: T-junction scenario for two vehicles

(a) Highway merging destinations



(b) With cooperation (Left) and without cooperation (Right)

Fig. 4: Highway merging for three vehicles

problem reduces to convex optimization problems that can be solved in polynomial time [10]. Tab. IV summarizes the number of variables and constraints of each MILP solved in the highway merging case study. Generating a control strategy in a 5-vehicle scenario required at most 10.3 s.

Overall, these examples illustrate the capabilities and expressiveness of the proposed contract framework to enable reasoning about the overall behavior of a system with multiple, possibly (non-)cooperating components.

VII. Conclusions

We proposed an extension of the A/G contract framework that explicitly includes objectives as first-class entities, along with assumptions and guarantees, and provides additional composition operators to capture component interactions that involve (non-)cooperation. There are several possible extensions of this work, including further investigation of the properties of the newly defined operators, the relationship between optimizing contracts and the contract meta-theory in the literature [1], and concrete instances of the framework in the context of multi-agent dynamical systems and hybrid games.

TABLE III: Results from cooperation and non-cooperation

		Arrival Time [s]				$\sum_{i=1}^{H-1} \lvert u_i \rvert$ $[m/s^2]$			
		v_1	v_2	v_3	Total	v_1	v_2	v_3	Total
T-junction	Coop	3.9	2.9		6.8	37.4	22.5		59.9
	Non-Coop	4.7	3.3		8.0	30.8	34.8		65.6
Highway Merging	Coop	5.3	6.7	7.0	19.0	5.4	10.7	0.1	16.2
	Non-Coop	5.3	7.5	> 8.0	> 20.8	9.3	14.4	15.3	39.0

TABLE IV: Runtime for the highway merging scenario

		Number of MILPs	Number of Variables Binary	Number of Variables Continuous	Number of Constraints	Runtime [sec]
Highway Merging 2 V	Coop	8	380	240	38	19.6
	Non-Coop	48	380	240	46	53.1
Highway Merging 3 V	Coop	8	540	360	56	31.1
	Non-Coop	72	540	360	80	140.9
Highway Merging 5 V	Coop	8	1280	600	132	82.4
	Non-Coop	120	1280	600	212	364.0

References

[1] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, K. G. Larsen *et al.*, "Contracts for system design," *Foundations and Trends in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[2] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems," *European Journal of Control*, vol. 18, no. 3, pp. 217–238, 2012.

[3] S. Graf, R. Passerone, and S. Quinton, "Contract-based reasoning for component systems with rich interactions," in *Embedded Systems Development, From Functional Models to Implementations*, 2014, pp. 139–154.

[4] W. Damm, H. Hungar, B. Josko, T. Peikenkamp, and I. Stierand, "Using contract-based component specifications for virtual integration testing and architecture design," in *Proc. Design, Automation and Test in Europe*, Mar. 2011, pp. 1–6.

[5] P. Nuzzo, H. Xu, N. Ozay, J. B. Finn, A. L. Sangiovanni-Vincentelli, R. M. Murray, A. Donzé, and S. A. Seshia, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[6] P. Nuzzo, A. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proc. IEEE*, vol. 103, no. 11, Nov. 2015.

[7] A. Cimatti and S. Tonetta, "Contracts-refinement proof system for component-based embedded systems," *Science of Computer Programming*, vol. 97, Part 3, pp. 333 – 348, 2015.

[8] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*. Prentice hall New Jersey, 1996, vol. 40.

[9] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[10] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model Predictive Control for Signal Temporal Logic Specification," *ArXiv e-prints*, Mar. 2017.

[11] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *HSCC*, 2015, pp. 239–248.

[12] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *CDC*, 2008, pp. 2117–2122.

[13] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Trans. Automat. Contr.*, vol. 59, no. 5, pp. 1244–1257, 2014.

[14] M. Maasoumy, P. Nuzzo, F. Iandola, M. Kamgarpour, A. Sangiovanni-Vincentelli, and C. Tomlin, "Optimal load management system for aircraft electric power distribution," in *CDC*, 2013, pp. 2939–2945.

[15] M. Svorenová, I. Cerna, and C. Belta, "Optimal control of mdps with temporal logic constraints," in *CDC*, 2013, pp. 3938–3943.

[16] W. L. Winston, *Operations Research: Applications & Algorithms*. Thomson Business Press, 2008.

[17] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia, "Diagnosis and repair for synthesis from signal temporal logic specifications," in *HSCC*, 2016, pp. 31–40.

[18] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Symp. Computer Aided Control Systems Design*, Taipei, Taiwan, 2004.

[19] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com