Contract-based Design and Verification of Multi-Agent Systems with Quantitative Temporal Requirements

Rafael Dewes, Rayna Dimitrova

CISPA Helmholtz Center for Information Security
{rafael.dewes, dimitrova}@cispa.de

Abstract

Quantitative requirements play an important role in the context of multi-agent systems, where there is often a trade-off between the tasks of individual agents and the constraints that the agents must jointly adhere to. We study multi-agent systems whose requirements are formally specified in the quantitative temporal logic LTL[$\mathcal{F}$] as a combination of local task specifications for the individual agents and a shared safety constraint. The intricate dependencies between the individual agents entailed by their local and shared objectives make the design of multi-agent systems error-prone, and their verification time-consuming. In this paper we address this problem by proposing a novel notion of quantitative assume-guarantee contracts, that enables the compositional design and verification of multi-agent systems with quantitative temporal specifications. The crux of these contracts lies in their ability to capture the coordination between the individual agents to achieve an optimal value of the overall specification under any possible behavior of the external environment. We show that the proposed framework improves the scalability and modularity of formal verification of multi-agent systems against quantitative temporal specifications.

1 Introduction

Multi-agent systems (MAS) have become increasingly important in various domains, such as robotics, distributed computing, and autonomous vehicles. In MAS (Wooldridge 2009), agents operate autonomously but may need to interact with each other to achieve shared objectives. Compared to larger monolithic systems, the ability of multiple agents to work collaboratively towards a common goal offers significant advantages in terms of efficiency and flexibility.

Coordination in MAS is a major challenge, as it involves not only the synchronization of actions among agents but also handles interdependencies and emergent conflicts. While it is crucial for aligning strategies, it can also introduce overhead and complexity, especially in dynamic and unpredictable environments. Additionally, achieving shared goals often requires balancing individual agent preferences and global objectives, which can introduce conflict.

To address these challenges, *contract-based design* (Benveniste et al. 2018) offers a structured way to define and manage interactions between agents. Compositional design approaches employ different forms of contracts, such as assume-guarantee contracts (Benveniste et al. 2007). However, traditional contract frameworks fall short in handling complex quantitative requirements and ensuring optimal collective performance, where agents must adapt to varying conditions and still aim for the best possible outcomes.

In this paper, we tackle the problem of designing an expressive contract framework for the compositional and best-effort satisfaction of quantitative specifications in MAS with collective goals. Our proposed framework extends traditional contract-based methods by incorporating quantitative measures and best-effort guarantees. Agents must strive for best possible outcomes even when perfect satisfaction is not feasible, ensuring that the system remains functional and effective in suboptimal environments. The approach includes formal verification methods to establish that the decomposition of global specifications into local contracts maintains the integrity of overall system goals, ensuring that individual agent behaviors align with the collective objectives.

We consider compositional specifications in the quantitative temporal logic LTL[$\mathcal{F}$] (Almagor, Boker, and Kupferman 2016), which extends linear temporal logic (LTL) with quality operators. For a given trace, an LTL[$\mathcal{F}$] formula evaluates to a value from a finite set of possible satisfaction values. We focus on specifications combining requirements *local* to individual agents and *shared* requirements constraining the agents' interaction. Local requirements only include actions of the respective agent, while shared objectives can contain any actions but must be safety properties. Our framework ensures that individual agent behavior aligns with both local and shared system-wide objectives. Existing work mostly studies the qualitative case where agents have a common objective. In our model agents are cooperative, unlike non-cooperative MAS where concepts like Nash equilibria are central (Kupferman, Perelli, and Vardi 2016). Closest to our approach is (Dewes and Dimitrova 2023), however our framework is more expressive and, unlike (Dewes and Dimitrova 2023), our decomposition contracts are complete. We will highlight these differences in Section 4.

Best-effort satisfaction means the highest satisfaction value enabled by the environment must be met by the system. In contrast to previous work, our contracts take this into account to allow for different levels of coordination based

on the inputs. It relates to the qualitative notion of dominant strategies (Damm and Finkbeiner 2014) that perform as good as the best alternative. We build on the idea of "good-enough" realizability (Almagor and Kupferman 2020) to encompass quantitative compositional specifications.

The following example illustrates how the best-effort interpretation of compositional quantitative specifications leads to conditional obligations in a MAS.

Example 1.1. *In this example, we consider a MAS with three agents modelling vehicles: two of them (agents a_1 and a_2) can transport goods to a construction site, and the third (agent a_3) provides a visual feed. The desired behavior of the overall system is specified as an LTL$[\mathcal{F}]$ formula*

$$\Phi = \tfrac{1}{6}\varphi^1_{local} \oplus \tfrac{1}{6}\varphi^2_{local} \oplus \tfrac{1}{6}\varphi^3_{local} \oplus \tfrac{1}{2}\Psi_{shared},$$

which is the weighted sum ($\oplus$) of the sub-formulas φ^i_{local} and ψ_{shared}. Each agent a_i has a local specification φ^i_{local} that prescribes their individual objectives. The shared requirement Ψ_{shared} encodes a task assigned collectively to all agents. Here, the local requirements are of equal priority, while the shared requirement Ψ_{shared} is weighed higher.

The agents operate in an environment that can issue calls to move to the site and send weather warnings, modeled with propositions call *and* badw *respectively. Ψ_{shared} requires that after each call, a_3 and at least one of a_1 and a_2 should respond in one or two time steps. According to φ^i_{local}, vehicles a_1 and a_2 cannot travel from the base to the site in an instant, and it is undesirable to travel during bad weather. We also require that each of a_1 and a_2 spends two consecutive time steps at the base infinitely often. The vehicle modelled by a_3 travels faster, and is unaffected by the weather, but it must be at the base infinitely often (to archive data).*

Best-effort satisfaction of Φ means that it must be satisfied as well as possible under the current environment, by the system as a whole. Whenever call *and* badw *occur together, to satisfy Ψ_{shared} either a_1 and a_2 must travel in bad weather, thus lowering the overall value of Φ. If the frequency of this is high enough, then either both a_1 and a_2 must travel in bad conditions, or one of them will be unable to stay at the base for two steps, violating φ^i_{local}.*

There is an obvious need for coordination between the agents to collectively good-enough satisfy Φ. Here, the agents must distribute the tasks to satisfy Ψ_{shared} and agree on values with which the local specifications for a_1 and a_2 are satisfied. Existing contract notions are not suitable for expressing this balancing of quantitative obligations. We present a framework for contract-based design that makes it possible to capture such coordination in the form of a contract. This enables independent design of the individual agents, leaving flexibility for the implementation of each agent outside of their shared obligations.

2 Preliminaries

In this section we recall definitions and notation necessary for presenting our framework, such as formal languages, the specification logic LTL$[\mathcal{F}]$, and the formalization of best-effort, or "good-enough", satisfaction.

Languages Let Σ be a finite alphabet. The set of finite (infinite) words over Σ is denoted by Σ^* (resp. Σ^ω). For $\sigma = \sigma_0\sigma_1\sigma_2\ldots \in \Sigma^\omega$, we denote $\sigma[i] = \sigma_i$, and with $\sigma[i,\infty) = \sigma_i, \sigma_{i+1}, \ldots$ the suffix of σ starting at position i. For $\sigma' \in \Sigma^*$ and $\sigma'' \in \Sigma^* \cup \Sigma^\omega$, we denote with $\sigma' \cdot \sigma''$ their concatenation. A language $L \subseteq \Sigma^\omega$ is a *safety language* if and only if for every $\sigma \in \Sigma^\omega \setminus L$ there exists a finite prefix σ' of σ such that $\sigma' \cdot \sigma'' \notin L$ for every $\sigma'' \in \Sigma^\omega$.

For a set X, we denote with 2^X the powerset of X. For a word σ over alphabet 2^X and a subset $Y \subseteq X$ we denote with $\mathsf{proj}(\sigma, Y)$ the projection of σ onto the alphabet 2^Y. For $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. Given disjoint sets $X_1, \ldots, X_m$ and for each $i \in \{1, \ldots, m\}$ a word σ_i over the alphabet 2^{X_i}, we define the parallel composition $\|_{i=1}^m \sigma_i$ of $\sigma_1, \ldots, \sigma_m$ such that $(\|_{i=1}^m \sigma_i)[j] = \bigcup_{i=1}^m \sigma_i[j]$ for all j.

The Specification Language LTL$[\mathcal{F}]$ We now recall the temporal logic LTL$[\mathcal{F}]$ introduced in (Almagor, Boker, and Kupferman 2016). Let AP be a set of Boolean atomic propositions, and $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] \mid k \in \mathbb{N}\}$ a set of functions. The LTL$[\mathcal{F}]$ formulas are generated by the grammar

$$\varphi ::= p \mid \mathsf{true} \mid \mathsf{false} \mid f(\varphi_1, \ldots, \varphi_k) \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2,$$

where $p \in AP$ is an atomic proposition, and $f \in \mathcal{F}$.

We consider sets $\mathcal{F}$ that include functions representing the Boolean operators, i.e., $\{f_\neg, f_\wedge, f_\vee\} \subseteq \mathcal{F}$, where $f_\neg(x) := 1 - x$, $f_\wedge(x,y) := \min\{x,y\}$ and $f_\vee(x,y) := \max\{x,y\}$. For ease of notation, we use the operators $\neg, \wedge, \vee$ instead of the corresponding functions. One useful function is the weighted average $x \oplus_\lambda y := \lambda \cdot x + (1-\lambda)\cdot y$, where $\lambda \in \{0, 1\}$. Abusing notation, we often use $\oplus$ with more than two arguments. $\bigcirc$ and $\mathcal{U}$ are the LTL *next* and *until* operators respectively (Baier and Katoen 2008). We define the usual derived temporal operators *finally* $\diamondsuit\varphi := \mathsf{true}\,\mathcal{U}\,\varphi$ and *globally* $\square\varphi := \neg(\diamondsuit\neg\varphi)$. $\square$ requires its argument to hold at every step from now on; $\diamondsuit$ requires that it holds at some point in the future; $\bigcirc$ requires it to be true in the next step.

The semantics of LTL$[\mathcal{F}]$ is defined with respect to words in $(2^{AP})^\omega$, and maps an LTL$[\mathcal{F}]$ formula φ and a word $\sigma \in (2^{AP})^\omega$, to a value $[\![\varphi, \sigma]\!] \in [0,1]$. For $f \in \mathcal{F}$, we define $[\![f(\varphi_1, \ldots, \varphi_k), \sigma]\!] := f([\![\varphi_1, \sigma]\!], \ldots, [\![\varphi_k, \sigma]\!])$. The semantics of the *until* operator $\mathcal{U}$ is $[\![\varphi_1 \,\mathcal{U}\, \varphi_2, \sigma]\!] := \max_{i \geq 0}\{\min\{[\![\varphi_2, \sigma[i,\infty)]\!], \min_{0 \leq j < i}[\![\varphi_1, \sigma[j,\infty)]\!]\}\}$. We refer the reader to (Almagor, Boker, and Kupferman 2016) for the full definition of the semantics. We denote with $\mathit{Vals}(\varphi) := \{[\![\varphi, \sigma]\!] \mid \sigma \in (2^{AP})^\omega\}$ the set of possible values of an LTL$[\mathcal{F}]$ formula φ. (Almagor, Boker, and Kupferman 2016) showed that $|\mathit{Vals}(\varphi)| \leq 2^{|\varphi|}$ for every LTL$[\mathcal{F}]$ formula φ, where $|\varphi|$ is the formula's description size. Thus, each formula's set of possible values is finite.

Example 2.1. *We formalize Example 1.1 using LTL$[\mathcal{F}]$. The location and movement of the vehicles are modelled by propositions* $\mathsf{at\_base}_i$ *and* $\mathsf{at\_site}_i$*. For φ^i_{local} for $i \in \{1, 2\}$:*

$$\begin{aligned}\varphi^i_{local} :=\ & \square(\neg\mathsf{at\_base}_i \vee \neg\mathsf{at\_site}_i) \wedge \\ & \square(\mathsf{at\_base}_i \rightarrow \bigcirc\neg\mathsf{at\_site}_i) \wedge \\ & \square\diamondsuit(\mathsf{at\_base}_i \wedge \bigcirc\mathsf{at\_base}_i) \wedge \\ & \square(\mathsf{badw} \rightarrow \bigcirc((\mathsf{at\_base}_i \vee \mathsf{at\_site}_i) \oplus_{\frac{1}{2}} \top)).\end{aligned}$$

This formula evaluates to one of three values: It is 0 if any of the first three conjuncts are violated. Otherwise, it is 1 if

agent a_i never travels during bad weather, and $\frac{1}{2}$ if it does (even just once). Further, $\varphi^3_{local} := \square\Diamond\neg\mathsf{at\_site}_3$.

$\Psi_{shared} := \square\big(\mathsf{call} \rightarrow \bigvee_{i=1}^{2}(respond_{i,3} \vee \bigcirc respond_{i,3})\big)$, *where* $respond_{i,3} := \neg\mathsf{at\_site}_i \wedge \bigcirc\mathsf{at\_site}_i \wedge \bigcirc\mathsf{at\_site}_3$. *Thus, vehicles must be at the site within two units of time. Vehicles a_1 and a_2 must arrive and not already be there, to model transporting goods.*

3 Quantitative Temporal Specifications of Multi-Agent Systems

We now present a formal model of MAS and the notion of *compositional LTL[$\mathcal{F}$] specifications* of such systems introduced in (Dewes and Dimitrova 2023).

Formal Model for Multi-Agent Systems We consider MAS where agents interact both with each other and the external environment via Boolean variables, i.e., atomic propositions. We fix a set AP of atomic propositions such that $AP = I \uplus O$, for a finite set I of *input atomic propositions* and a finite set O of *output atomic propositions* that are disjoint, i.e., $I \cap O = \emptyset$. The input propositions I are updated by the external environment and the output propositions O are updated by the agents in the system. We call words in $(2^{I\cup O})^\omega$ *(execution) traces* and those in $(2^I)^\omega$ *input traces*.

An *agent* in the system is modelled as a Moore machine $M = (I_M, O_M, S, s^{init}, \rho, Out)$, where I_M and O_M are M's sets of input and output propositions, S is a finite set of states, with initial state $s^{init} \in S$, transition function $\rho : S \times 2^{I_M} \to S$, and output labeling function $Out : S \to 2^{O_M}$. For input trace $\sigma_{I_M} \in (2^{I_M})^\omega$, M produces the output trace $M(\sigma_{I_M}) \in (2^{O_M})^\omega$ such that $M(\sigma_{I_M})[i] = Out(s_i)$, where the sequence of states $s_0 s_1 \ldots \in S^\omega$ is such that $s_0 = s^{init}$, and $s_{i+1} = \rho(s_i, \sigma_{I_M}[i])$ for all i.

A *multi-agent system* (MAS) is a tuple $\mathcal{S} = (I, O, \mathcal{M})$, where I and O are the sets of input and output propositions respectively, $\mathcal{M} = \langle M_1, \ldots M_n\rangle$ is a tuple of Moore machines representing the agents, where $M_a = (I_a, O_a, S_a, s_a^{init}, \rho_a, Out_a)$ is such that (1) for every $a, a' \in [n]$ with $a \neq a'$ it holds that $O_a \cap O_{a'} = \emptyset$, (2) $\biguplus_{a=1}^{n} O_a = O$, and (3) $I_a = I \cup (O \setminus O_a)$. Conditions (1) and (2) state that the sets of output propositions of the individual agents partition O. Condition (3) stipulates that each agent observes all input propositions I and all output propositions of the other agents. We denote with $\mathsf{Agt} := [n]$ the set of all agents, and with $O_{\overline{a}} = \bigcup_{a'\in\mathsf{Agt}\setminus\{a\}} O_{a'}$ the set of outputs of agents different from a. Our model uses a composition of Moore machines, similar to Moore synchronous game structures in (Alur, Henzinger, and Kupferman 2002), however here each machine models the implementation of an agent. Given an input trace $\sigma_I \in (2^I)^\omega$, a MAS generates an output trace $\sigma_O \in (2^O)^\omega$, which we denote by $(\|_{c=1}^{n} M_a)(\sigma_I)$, such that $\mathsf{proj}(\sigma_O, O_a) = M_a(\sigma_I \parallel \mathsf{proj}(\sigma_O, O_{\overline{a}}))$ for all a. That is, $(\|_{a=1}^{n} M_a)(\sigma_I)$ is the *composition of all the output traces* of the agents in $\mathcal{S}$. We define the set of traces $\mathcal{L}(M_a) \subseteq (2^{I\cup O})^\omega$ as $\mathcal{L}(M_a) := \{\sigma \in 2^{I\cup O})^\omega \mid M_a(\mathsf{proj}(\sigma, I_a)) = \mathsf{proj}(\sigma, O_a)\}$.

Good-Enough Satisfaction of LTL[$\mathcal{F}$] Specifications Following (Almagor and Kupferman 2020), we consider *good-enough satisfaction* of an LTL[$\mathcal{F}$] specification Φ by a system $\mathcal{S}$. Intuitively, we require that for every input trace, the system's output results in the best value for Φ possible for this input trace. To formalize this, we use the notion of *v-hopeful input sequences* (Almagor and Kupferman 2020). Those are the input sequences for which there exists output sequences such that Φ has value v in the resulting executions. Formally, for $v \in [0,1]$, the set of *v-hopeful input sequences for the formula* Φ is $\mathsf{Hopeful}(v, \Phi) := \{\sigma_I \in (2^I)^\omega \mid \exists \sigma_O \in (2^O)^\omega.\ [\![\Phi, (\sigma_I \parallel \sigma_O)]\!] = v\}$.

Definition 1 (Good-Enough Satisfaction)**.** *Given an LTL[$\mathcal{F}$] formula Φ over $AP = I \uplus O$, and a MAS $\mathcal{S} = (I, O, \mathcal{M})$, we say that $\mathcal{S}$* satisfies *Φ, denoted $\mathcal{M} \models \Phi$, if and only if for every $v \in [0,1]$ and every v-hopeful input sequence $\sigma_I \in \mathsf{Hopeful}(v, \Phi)$, it holds that $[\![\Phi, \sigma_I \parallel \mathcal{M}(\sigma_I)]\!] \geq v$.*

Compositional LTL[$\mathcal{F}$] Specifications In this paper we focus on a particular class of LTL[$\mathcal{F}$] specifications for MAS, called *compositional specifications*, introduced in (Dewes and Dimitrova 2023), which studies the problem of automatic synthesis of MAS from such specifications. Compositional specifications are given as a combination of local specifications for the individual agents and a shared requirement. Formally, a *compositional specification* is of the form $\Phi = comb(\varphi^1_{local}, \ldots, \varphi^n_{local}, \Psi_{shared})$ where

(1) $comb : [0,1]^{n+1} \to [0,1]$ is non-decreasing in each subset of arguments: if $comb(v'_1, \ldots, v'_{n+1}) < comb(v_1, \ldots, v_{n+1})$, then, $v'_i < v_i$ for some $i \in [n+1]$.

(2) The *shared specification* Ψ_{shared} is a *safety* LTL[$\mathcal{F}$] specification. This means that for every $\sigma \in (2^{AP})^\omega$ and $v \in Vals(\Psi_{shared})$, if $[\![\Psi_{shared}, \sigma]\!] < v$, then there exists a prefix σ' of σ, such that $[\![\Psi_{shared}, \sigma' \cdot \sigma'']\!] < v$ for every infinite continuation $\sigma'' \in (2^{AP})^\omega$ of σ'.

(3) For each agent a, the *local specification* φ^a_{local} for agent a refers only to atomic propositions in $O_a \cup I$.

Such specifications occur in MAS where we have task specifications for individual agents describing their desired behavior, as well as a shared safety constraint. Going forward, we write Φ to mean a compositional specification $\Phi = comb(\varphi^1_{local}, \ldots, \varphi^n_{local}, \Psi_{shared})$.

4 Quantitative Assume-Guarantee Contracts

In this section we introduce a new notion of contracts for MAS with compositional LTL[$\mathcal{F}$] specifications as defined in Section 3. These contracts, termed *good-enough decomposition contracts* (GEDC) can express coordination between the agents while taking into account the quantitative nature of LTL[$\mathcal{F}$] requirements. In contrast to traditional contracts used for compositional design and verification, the type of contracts we propose allows the specification of the quantitative contribution of each agent's local requirements to the satisfaction value of the overall specification. This, in turn, allows for assumptions and guarantees that impose weaker restrictions on agents' behavior by stating *what* the satisfaction values of the local specifications should be, and not *how* these values should be achieved. After introducing GEDC, we define what it means for a MAS to satisfy a GEDC, and show that this notion is sound and complete.

A GEDC for a compositional LTL$[\mathcal{F}]$ specification Φ consists of two components. The first, called *allowed decompositions map*, specifies for each possible value v of Φ a set of *value decompositions* describing how the satisfaction of each local specification and the shared specification contribute to achieving value v for Φ. The second component of a GEDC associates with each value v an *assume-guarantee contract* that specifies, in terms of behaviors, how the agents cooperate to achieve the desired satisfaction values.

We begin with the definition of value decompositions. Intuitively, the decompositions of v with respect to $\Phi = comb(\varphi^1_{local}, \ldots, \varphi^n_{local}, \Psi_{shared})$ are tuples of values for $\varphi_1, \ldots, \varphi_n, \Psi_{shared}$ such that their combination results in v. Formally, given $v \in Vals(\Phi)$, a *value decomposition* of v is a vector $\mathbf{d} \in (\bigtimes_{a=1}^{n} Vals(\varphi^a_{local})) \times Vals(\Psi_{shared})$ where $comb(\mathbf{d}) = v$. If $\mathbf{d} = (\langle u_a \rangle_{a=1}^{n}, w)$, we denote the components of $\mathbf{d}$ by $\mathbf{d}[a] := u_a$ for $a \in [n]$, and $\mathbf{d}[s] := w$. For every $v \in Vals(\Phi)$, we denote with $Dec(\Phi, v) \subseteq \mathbb{R}^{n+1}$ the set consisting of all possible value decompositions of v.

Next, we define the notion of *allowed decompositions map* for Φ. Intuitively, the set $ADec$ associates with each $v \in Vals(\Phi)$ the value decompositions allowed by the contract for v-hopeful input sequences. To ensure generality, $ADec$ allow different sets of decompositions to be associated with different subsets H of $\mathsf{Hopeful}(v, \Phi)$.

Definition 2. *An* allowed decompositions map *for Φ is a set $ADec \subseteq Vals(\Phi) \times 2^{((2^I)^\omega)} \times 2^{\bigcup_{v \in Vals(\Phi)} Dec(\Phi, v)}$ where:*

1. *(Input coverage) For every $v \in Vals(\Phi)$ and $\sigma_I \in \mathsf{Hopeful}(v, \Phi)$, there exists $(v', H, D) \in ADec$ such that $v' \geq v$ and $\sigma_I \in H$. This ensures that every v-hopeful input sequence is covered by some set H for value $v' \geq v$.*
2. *(Satisfiable decompositions) For every $(v, H, D) \in ADec$, $D \subseteq Dec(\Phi, v)$ and for every $\mathbf{d} \in D$ there exists $\sigma \in (2^{I \cup O})^\omega$ such that $[\![\Psi_{shared}, \sigma]\!] = \mathbf{d}[s]$ and $[\![\varphi^a_{local}, \sigma]\!] = \mathbf{d}[a]$ for all $a \in [n]$. That is, D only contains satisfiable decompositions from $Dec(\Phi, v)$.*

For every input sequence $\sigma_I \in (2^I)^\omega$ and $v \in Vals(\Phi)$ we define $ADec(v, \sigma_I) := \{\mathbf{d} \mid \exists (v, H, D) \in ADec.\ \sigma_I \in H \text{ and } \mathbf{d} \in D\}$ to be the set of decompositions of v allowed by $ADec$ for the input sequence σ_I. Since every input sequence σ_I is hopeful for some value in $Vals(\Phi)$, condition 1 in Definition 2 implies that $ADec(v, \sigma_I)$ is nonempty for some v. This ensures that a system which conforms with $ADec$ has appropriate obligations w.r.t. every input sequence. Furthermore, condition 2 guarantees that these obligations are met by satisfiable value decompositons.

For a given specification Φ, there may be multiple possible allowed decompositions maps that differ in the obligations they impose on the individual agents. Next, we show the possible value decompositions for the formula in Example 1.1 and one possible allowed decompositions map.

Example 4.1. *The worst-case satisfaction value for Φ in Example 1.1 is $\frac{5}{6}$. If* badw *and* call *are constantly true, this is the best that can be achieved: Ψ_{shared} must be satisfied, meaning that a_1 or a_2 must travel in bad weather, or not be at the base often enough. The four potential value decompositions for $v = \frac{5}{6}$ are $(\frac{1}{2}, \frac{1}{2}, 1, 1)$, $(0, 1, 1, 1)$, $(1, 0, 1, 1)$, $(1, 1, 0, 1)$. The last tuple is impossible to achieve in the described environment. $(\frac{1}{2}, \frac{1}{2}, 1, 1)$ corresponds to case where both a_1 and a_2 travel in bad weather. The other two, to either a_1 or a_2 not being at the base for two steps infinitely often. All three are possible for any $\frac{5}{6}$-hopeful input sequence. A most permissive $ADec$ will thus contain $(\frac{5}{6}, \mathsf{Hopeful}(\frac{5}{6}), \{(\frac{1}{2}, \frac{1}{2}, 1, 1), (0, 1, 1, 1), (1, 0, 1, 1)\})$. An alternative is to take a subset, enforcing specific priorities.*

While allowed decompositions maps impose requirements on the contribution by each agent to the overall satisfaction value, they do not specify *how* the agents cooperate to achieve these values. This is done by providing for each $v \in Vals(\Phi)$ an *assume guarantee contract* which specifies what assumptions each agent makes on the behavior of other agents and what guarantees it provides in return. We recall the definition from (Dewes and Dimitrova 2023).

Definition 3. *An* assume-guarantee contract *for n agents is a tuple $\langle (A_a, G_a) \rangle_{a=1}^{n}$, where for every agent $a \in \mathsf{Agt}$:*

- *$A_a, G_a \subseteq (2^{AP})^\omega$ are safety languages defining the assumption and guarantee, respectively, of agent a.*
- *$\bigcap_{a' \in \{1, \ldots, n\} \setminus \{a\}} G_{a'} \subseteq A_a$.*
- *For every finite trace $\sigma \in (2^{AP})^*$:*
  1. *If there exists an infinite word $\sigma' \in (2^{AP})^\omega$ such that $\sigma \cdot \sigma' \in A_a$, then, for every $o_a \in 2^{O_a}$, there exists $\sigma'' \in (2^{AP})^\omega$ with $\sigma \cdot \sigma'' \in A_a$ and $\mathsf{proj}(\sigma''[0], O_a) = o_a$.*
  2. *If there exists an infinite word $\sigma' \in (2^{AP})^\omega$ such that $\sigma \cdot \sigma' \in G_a$, then, for every $o_{\overline{a}} \in 2^{O_{\overline{a}}}$, there exists $\sigma'' \in (2^{AP})^\omega$ with $\sigma \cdot \sigma'' \in G_a$ and $\mathsf{proj}(\sigma''[0], O_{\overline{a}}) = o_{\overline{a}}$.*

The above conditions ensure two properties. Agent a cannot violate its own assumption A_a by selecting a bad output o_a. The remaining agents cannot violate the guarantee which agent a must provide by selecting a bad output $o_{\overline{a}}$.

We are now ready to give the definition of a good-enough decomposition contract, which pairs up an allowed decompositions map with a function mapping each $v \in Vals(\Phi)$ to an assume-guarantee contract. As a result, for each $v \in Vals(\Phi)$, the contract prescribes allowed decompositions, depending on the v-hopeful input sequences, and an associated assume-guarantee contract $\langle (A^v_a, G^v_a) \rangle_{a=1}^{n}$.

Definition 4 (Good-Enough Decomposition Contract)**.** *Let Φ be a compositional LTL$[\mathcal{F}]$ specification over $AP = I \uplus O$, where $O = O_1, \ldots, O_n$ is a partitioning of O among n agents. Let $ADec$ be a map of allowed decompositions for Φ, and let AG be a function that maps each $v \in Vals(\Phi)$ to an assume-guarantee contract $\langle (A^v_a, G^v_a) \rangle_{a=1}^{n}$. We say that $(ADec, AG)$ is a* good-enough decomposition contract *for Φ and $O_1, \ldots, O_n$ if and only if for every $v \in Vals(\Phi)$ and $\sigma_I \in \mathsf{Hopeful}(v, \Phi)$, there exist $(v', H, D) \in ADec$, $\mathbf{d} \in D$ and $\sigma_O \in (2^O)^\omega$, such that $v' \geq v$ and the following hold:*

1. *$\sigma_I \in H$, $[\![\Psi_{shared}, \sigma_I \parallel \sigma_O]\!] = \mathbf{d}[s]$,*
2. *$[\![\varphi^a_{local}, \sigma_I \parallel \sigma_O]\!] = \mathbf{d}[a]$ for all $a \in [n]$, and*
3. *$(\sigma_I \parallel \sigma_O) \in \bigcap_{a \in \{1, \ldots, n\}} A^{v'}_a$.*

The conditions in Definition 4 ensure compatibility between $ADec$ and AG. More concretely, for each v-hopeful input sequence σ_I there exists a corresponding output trace

that agrees with some allowed value decomposition for value $v' \geq v$, and, in addition is consistent with the assumptions of all agents associated with value v'. The next example illustrates the importance of these conditions.

Example 4.2. *A GEDC for Φ in Example 1.1 combines some $ADec$ with an assume-guarantee contract to capture the coordination between the agents. First, in input sequences where* call *is true at least once, each agent must rely on help from other agents to cover Ψ_{shared}. More concretely, agents a_1 and a_2 assume that a_3 will always respond.*

When the best possible value is $v = \frac{5}{6}$, a_1 and a_2 must coordinate whether one of them should respond to all calls, or if both will reduce their local satisfaction value to $\frac{1}{2}$.

This is reflected in $ADec$ and the assume-guarantee contract. If agent a_1 guarantees to cover all calls in bad weather for some input sequences, and agent a_2 guarantees the same for all of the remaining input sequences, then value decomposition $(\frac{1}{2}, \frac{1}{2}, 1, 1)$ is not attainable, and both $(0, 1, 1, 1)$ and $(1, 0, 1, 1)$ must be in $ADec$. Alternatively, if the assumptions and guarantees force a_1 and a_2 to alternate responding to calls, with both potentially travelling in bad weather, then $(\frac{1}{2}, \frac{1}{2}, 1, 1)$ is the only matching value decomposition for value $v = \frac{5}{6}$.

Since a GEDC associates assumptions with each value v, it can enforce coordination for worst-case input sequences, while giving the agents more freedom when the environment is more helpful (allows a higher satisfaction value).

Remark. GEDC are more general than the contracts in (Dewes and Dimitrova 2023) in two key aspects. First, GEDC allow us to relate obligations to the hopefulness level of input sequences with respect to the overall specification, instead of being independent of the global combined value. Second, GEDC include explicit value decompositions which offer a more natural way to quantify the required contribution of each agent, as well as flexibility in specifying obligations. This results in a complete decomposition rule.

The value decompositions of an allowed decompositions map $ADec$ in a GEDC specify obligations for each agent a in terms of values for the local and the shared specification which a must ensure. An agent a should meet its obligation with respect to the value of Ψ_{shared} only if the other agents provide output sequences that permit that, that is, if *behaviors of the other agents are collaborative*. Otherwise, a is expected to achieve the best possible value for φ^a_{local}. We call a sequence $\sigma_{O_{\overline{a}}} \in (2^{O_{\overline{a}}})^\omega$ *D-collaborative for input sequence* σ_I, if there exists an output sequence $\sigma_{O_a} \in (2^{O_a})^\omega$ for agent a and a value decomposition $\mathbf{d} \in D$ where

- $[\![\varphi^a_{local}, (\sigma_I \parallel \sigma_{O_{\overline{a}}} \parallel \sigma_{O_a})]\!] \geq \mathbf{d}[a]$, and
- $[\![\Psi_{shared}, (\sigma_I \parallel \sigma_{O_{\overline{a}}} \parallel \sigma_{O_a})]\!] \geq \mathbf{d}[s]$.

We denote by $\mathsf{Collab}(D, \sigma_I)_{\bar{a}}$ the set of all such sequences.

Intuitively, D-collaborative behaviors of $\overline{a}$ are those that permit agent a to provide output such that the resulting values for φ^a_{local} and Ψ_{shared} conform to at least one decomposition in D. This is illustrated by the following example.

Example 4.3. *In the setting of Example 1.1, for input sequences where* call *is always true and* badw *is always false, a value $v = 1$ for Φ is possible with $D = \{(1, 1, 1, 1)\}$. In this case, the D-collaborative outputs for a_1 are those where agents a_2 and a_3 take care of* enough *calls such that a_1 can achieve full satisfaction of φ^1_{local}. More specifically, if a_2 sets* $\mathsf{at\_site}_2$ *to true every k steps, that will give a_1 enough freedom to set* $\mathsf{at\_base}_1$ *to true for two consecutive turns every k steps, and still meet its obligations towards Ψ_{shared}.*

We will now state what it means for a MAS $\mathcal{S}$ to satisfy a GEDC. Intuitively, the implementation M_a of each agent a should satisfy certain obligations for each $v \in Vals(\Phi)$ and combination of v-hopeful input sequence σ_I with outputs of the agents in $\overline{a}$ satisfying the assumption A^v_a. If the outputs of the agents in $\overline{a}$ are collaborative for σ_I and D, the obligation is determined by the value distributions in D, and otherwise, by the best value of φ^a_{local} possible for σ_I.

Definition 5 (GEDC Satisfaction)**.** *Let Φ be a compositional LTL$[\mathcal{F}]$ specification over $AP = I \uplus O$, $O_1, \ldots, O_n$ be a partitioning of O, and let $(ADec, AG)$ be a GEDC for Φ and $O_1, \ldots, O_n$. We say that a MAS $\mathcal{S} = (I, O, \mathcal{M})$* satisfies a GEDC *$(ADec, AG)$ iff for every agent $a \in$ Agt, its implementation M_a satisfies the following condition.*

For every $\sigma_I \in (2^I)^\omega$ and every sequence of outputs of the other agents $\sigma_{O_{\overline{a}}} \in (2^{O_{\overline{a}}})^\omega$, for every $(v, H, D) \in ADec$ with $\sigma_I \in H$ and $(\sigma_I \parallel M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}}) \parallel \sigma_{O_{\overline{a}}}) \in A^v_a$, where $AG(v) = \langle (A^v_a, G^v_a) \rangle_{a=1}^n$, all *of the following hold:*

1. *If $\sigma_{O_{\overline{a}}} \in \mathsf{Collab}(D, \sigma_I)_{\bar{a}}$, then for all $\mathbf{d} \in D$,*
   if there is an output sequence $\sigma_{O_a} \in (2^{O_a})^\omega$, with
   - $[\![\varphi^a_{local}, (\sigma_I \parallel \sigma_{O_a} \parallel \sigma_{O_{\overline{a}}})]\!] \geq \mathbf{d}[a]$, *and*
   - $[\![\Psi_{shared}, (\sigma_I \parallel \sigma_{O_a} \parallel \sigma_{O_{\overline{a}}})]\!] \geq \mathbf{d}[s]$,

   then, there exists $\mathbf{d}' \in D$ where
   - $\mathbf{d}'[a'] = \mathbf{d}[a']$ *for all $a' \neq a$, and*
   - $[\![\varphi^a_{local}, (\sigma_I \parallel M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}}) \parallel \sigma_{O_{\overline{a}}})]\!] \geq \mathbf{d}'[a]$,
   - $[\![\Psi_{shared}, (\sigma_I \parallel M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}}) \parallel \sigma_{O_{\overline{a}}})]\!] \geq \mathbf{d}'[s]$.
2. *If $\sigma_{O_{\overline{a}}} \in \mathsf{Collab}(D, \sigma_I)_{\bar{a}}$, then the guarantee G^v_a is satisfied, that is, $(\sigma_I \parallel M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}}) \parallel \sigma_{O_{\overline{a}}}) \in G^v_a$.*
3. *If $\sigma_{O_{\overline{a}}} \notin \mathsf{Collab}(D, \sigma_I)_{\bar{a}}$, then for $u \in Vals(\varphi^a_{local})$, if it holds that $\sigma_I \in \mathsf{Hopeful}(u, \varphi^a_{local})$, then it also holds that $[\![\varphi^a_{local}, (\sigma_I \parallel M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}}) \parallel \sigma_{O_{\overline{a}}})]\!] \geq u$.*

Generally, for the contract to be satisfied, the output $M_a(\sigma_I \parallel \sigma_{O_{\overline{a}}})$ produced by an agent a must 1 achieve best-effort satisfaction for φ^a_{local} and Ψ_{shared} depending on σ_I, in accordance with the decompositions $\mathbf{d}$ contained in $ADec$ and 2 satisfy the guarantees G^v_a as specified by the contract $(ADec, AG)$. What section of the contract applies primarily depends on which set H the sequence σ_I belongs to. There are two exceptions to the above requirement, depending on the behavior $\sigma_{O_{\overline{a}}}$ of other agents $\bar{a}$. First, if some of the assumptions A^v_a is violated, agent a is free from any obligations. Second, as stated by 3, if $\sigma_{O_{\overline{a}}}$ is not collaborative, agent a must only maximize the satisfaction of φ^a_{local}.

The next theorem states that GEDCs are *sound*. That is, a system that satisfies a GEDC $(ADec, AG)$ for a compositional LTL$[\mathcal{F}]$ specification Φ, is guaranteed to satisfy Φ.

Theorem 1 (Soundness of GEDC)**.** *Let Φ be a compositional LTL$[\mathcal{F}]$ specification and $(ADec, AG)$ be a GEDC for Φ. Then, for every MAS $\mathcal{S} = (I, O, \langle M_1, \ldots, M_n \rangle)$ it holds that if $\mathcal{S} \models (ADec, AG)$, then $\mathcal{S} \models \Phi$.*

Proof Sketch. To show that $\mathcal{S} \models \Phi$, we show for every $\sigma_I \in \mathsf{Hopeful}(v, \Phi)$ that $[\![\Phi, \sigma_I \parallel \mathcal{M}(\sigma_I)]\!] \geq v$. To do so, we prove that for some $a \in \mathsf{Agt}$, the outputs of the other agents are D-collaborative for σ_I and some D, and satisfy the assumption A_a^v. Applying 1 in Definition 5 this yields $[\![\Phi, \sigma_I \parallel \mathcal{M}(\sigma_I)]\!] \geq v$. To establish the existence of agent a with this property, we use the fact that Ψ_{shared} is a safety LTL$[\mathcal{F}]$ formula and that 3 enforces maximizing the satisfaction value of the local specifications for non-collaborative inputs, to show that the converse is impossible. □

GEDCs are also complete, i.e., if $\mathcal{S} \models \Phi$, then there exists a GEDC $(ADec, AG)$ for Φ with $\mathcal{S} \models (ADec, AG)$.

Theorem 2. *Let Φ be a compositional specification. If $\mathcal{S} = (I, O, \langle M_1, \ldots, M_n \rangle)$ is a MAS with $\mathcal{S} \models \Phi$, then there exists a GEDC $(ADec, AG)$ for Φ where $\mathcal{S} \models (ADec, AG)$.*

Proof Sketch. $\mathcal{S}$ produces unique outputs for each input sequence, the resulting trace satisfying exactly one value decomposition $\mathbf{d}$. $ADec$ is chosen to contain exactly the finitely many value decompositions realized by $\mathcal{S}$. The input traces matching each $\mathbf{d}$ form the respective set H. For AG, we define each G_a^v as the safety language encoding the implementation M_a, and the assumption A_a^v as $\bigwedge_{a' \neq a} G_{a'}^v$. Thus, AG prescribes the precise implementation M_a of each agent a. Full proof in (Dewes and Dimitrova 2024). □

5 Compositional Verification

In this section we present an automata-theoretic method for checking contract satisfaction as formalized in Definition 5: Given a compositional LTL$[\mathcal{F}]$ specification Φ, a GEDC $(ADec, AG)$ for Φ, and a system $\mathcal{S} = (I, O, \langle M_1, \ldots, M_n \rangle)$, decide whether $\mathcal{S} \models (ADec, AG)$. Further, we show how to check whether a given pair $(ADec, AG)$ is a GEDC for Φ, that is, it satisfies the conditions of Definition 2, Definition 3 and Definition 4.

By Theorem 1, if $\mathcal{S} \models (ADec, AG)$, we can conclude that $\mathcal{S} \models \Phi$. We remark that the specification Φ can also be checked directly, using standard techniques, by constructing the product of $M_1, \ldots, M_n$. This, however, is often undesirable. Working with the full specification and the product system hampers scalability. Furthermore, a GEDC allows for the independent re-implementation of the individual agents as long as the conditions in Definition 5 are satisfied.

Procedure for Checking Contract Satisfaction We first recall necessary automata definitions and constructions.

A *generalized nondeterministic Büchi automaton* (GNBA) over an alphabet Σ is a tuple $\mathcal{A} = (\Sigma, Q, \delta, Q_0, \alpha)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, and $\alpha \subseteq 2^Q$ a set of sets of accepting states. A run ρ of a GNBA is accepting iff for every $F \in \alpha$, ρ visits F infinitely often. A word $\sigma \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on σ. The language of $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) := \{\sigma \in \Sigma^\omega \mid \sigma \text{ is accepted by } \mathcal{A}\}$.

In (Almagor, Boker, and Kupferman 2016) it is shown that for any LTL$[\mathcal{F}]$ formula φ over AP, and $V \subseteq Vals(\varphi)$, there exists a GNBA $\mathcal{A}_{\varphi,V}$ such that $\mathcal{L}(\mathcal{A}_{\varphi,V})$ consists of the words $\sigma \in (2^{AP})^\omega$ with $[\![\varphi, \sigma]\!] \in V$, and $\mathcal{A}_{\varphi,V}$ has at most $2^{(|\varphi|^2)}$ states and at most $|\varphi|$ sets of accepting states. We employ this construction to reduce the problem of checking contract satisfaction to checking emptiness of GNBA.

For the remainder of this section we assume that $(ADec, AG)$ is finitely represented using ω-automata. More concretely, the set H in each $(v, H, D) \in ADec$ is represented by a GNBA $\mathcal{H}$ where $\mathcal{L}(\mathcal{H}) = \{\sigma_I \parallel \sigma_O \in (2^{AP})^\omega \mid \sigma_I \in H\}$. For each $v \in Vals(\Phi)$, the assume-guarantee contract $AG(v) = \langle (A_a^v, G_a^v) \rangle_{a=1}^n$ is represented by tuple of pairs of GNBA with alphabet 2^{AP}, one for the assumption language A_a^v, and one for the complement $2^{AP} \setminus G_a^v$ of the guarantee. We denote these automata by $\mathcal{A}_a^v$ and $\overline{\mathcal{G}_a^v}$. We assume that all automata in the contract are given as nondeterministic Büchi automata, i.e., have one accepting set.

In the next proposition we outline the construction of a GNBA $\mathcal{C}_a$ for agent a, whose language consists of the traces that violate some of the conditions in Definition 5. The full construction is given in the extended version (Dewes and Dimitrova 2024).

Proposition 1. *For each agent $a \in \mathsf{Agt}$, we can construct a GNBA $\mathcal{C}_a$ such that for any $\sigma = \sigma_I \parallel \sigma_O \in (2^{AP})^\omega$ we have that $\sigma \in \mathcal{L}(\mathcal{C}_a)$ if and only if there exists $(v, H, D) \in ADec$ such that $\sigma_I \in H$, and $\sigma \in A_a^v$, and at least one of the conditions in Definition 5 is violated for agent a.*

Proof Sketch. From the formulas φ_{local}^a and Ψ_{shared}, we construct for $\sim \in \{<, \geq, =\}$, $u \in Vals(\varphi_{local}^a)$ and $w \in Vals(\Psi_{shared})$ the GNBA $\mathcal{B}_a^{\sim u}$, and $\mathcal{B}_s^{\sim w}$ such that $\sigma \in \mathcal{L}(\mathcal{B}_a^{\sim u})$ iff $[\![\varphi_{local}^a, \sigma]\!] \sim u$, and similarly for $\mathcal{B}_s^{\sim w}$.

Next, we construct a GNBA $\mathcal{C}_a$ that characterizes the language of traces $\sigma = \sigma_I \parallel \sigma_{O_a} \parallel \sigma_{O_{\overline{a}}}$ such that there exists $(v, H, D) \in ADec$ with $\sigma_I \in H$, and $\sigma \in A_a^v$, and some of the conditions 1, 2 or 3 in Definition 5 is violated. $\mathcal{C}_a$ is constructed from $\mathcal{B}_a^{\sim u}, \mathcal{B}_s^{\sim w}, \mathcal{A}_a^v, \overline{\mathcal{G}_a^v}$ and $\mathcal{H}$ using standard automata operations on GNBA. □

Using the automata $\mathcal{C}_a$ constructed in Proposition 1, we can automatically check whether $\mathcal{S} \models (ADec, AG)$ by checking for each agent $a \in \mathsf{Agt}$, whether the intersection of the languages of $\mathcal{C}_a$ and the implementation M_a is empty. If $\mathcal{L}(\mathcal{C}_a) \cap \mathcal{L}(M_a) = \emptyset$, then M_a satisfies conditions of Definition 5, otherwise we find a counterexample. If for all $a \in \mathsf{Agt}$ we establish $\mathcal{L}(\mathcal{C}_a) \cap \mathcal{L}(M_a) = \emptyset$, we conclude that $\mathcal{S} \models (ADec, AG)$, and hence $\mathcal{S} \models \Phi$. Otherwise, we have that $\mathcal{S} \not\models (ADec, AG)$. This, however, does not imply $\mathcal{S} \not\models \Phi$, since the GEDC might be insufficient to verify Φ.

Remark. When checking whether $\mathcal{L}(\mathcal{C}_a) \cap \mathcal{L}(M_a) = \emptyset$, this can be done separately for each value $v \in Vals(\Phi)$, and incrementally when constructing $\mathcal{C}_a$. We further check emptiness on automata intersections in an on-the-fly manner.

The number of states of the GNBA $\mathcal{C}_a$ is at most $2^{2^{\mathcal{O}(|\varphi_{local}^a|^2 + |\Psi_{shared}|^2)}} \cdot \sum_{(v,\mathcal{H},D) \in ADec} |\mathcal{H}| \cdot |\mathcal{A}_a^v| \cdot |\overline{\mathcal{G}_a^v}|$. Checking contract satisfaction is done by checking for each agent the emptiness of the intersection of $\mathcal{C}_a$ and the respective implementation M_a, which can be done in time linear in the size of their product (Baier and Katoen 2008).

Theorem 3 (Checking GEDC Satisfaction)**.** *Checking if a MAS $\mathcal{S} = (I, O, \langle M_1, \ldots, M_n \rangle)$ satisfies a GEDC $(ADec, AG)$ for a compositional LTL$[\mathcal{F}]$ specification Φ*

can be done for each agent $a \in \mathsf{Agt}$ in time linear in $|M_a| \cdot 2^{2^{\mathcal{O}(|\varphi^a_{local}|^2 + |\Psi_{shared}|^2)}} \cdot \sum_{(v,\mathcal{H},D)\in ADec} |\mathcal{H}| \cdot |\mathcal{A}^v_a| \cdot |\overline{\mathcal{G}_a}^v|$.

Procedure for Checking $(ADec, AG)$ is a GEDC for Φ To check that $(ADec, AG)$ is a GEDC for Φ, we construct the automaton for Φ, of size at most $2^{(|\Phi|^2)}$. The conditions are established by checking language inclusion and emptiness for GNBA obtained from those for Φ and $(ADec, AG)$. Both can be done in time exponential in the size of the resulting automata, resulting in a procedure that for each entry $(v, H, D) \in ADec$ runs in time exponential in the size of the automata $\mathcal{H}$, $\mathcal{A}^v_a$ and $\overline{\mathcal{G}_a}^v$ and double exponential in $|\Phi|^2$.

Modular Verification The contract-based approach allows for modular verification and design. In a MAS that satisfies a given GEDC, the local modification of an agent's implementation or specification can be analyzed without considering the full specification or implementation.

Consider a compositional specification Φ, a GEDC $(ADec, AG)$ for Φ, and a MAS $\mathcal{S} = (I, O, \langle M_1, \ldots, M_n\rangle)$ such that $\mathcal{S} \models (ADec, AG)$. Suppose that the local specification for agent a is modified from φ^a_{local} to $\widehat{\varphi}^a_{local}$, resulting in a revised global specification $\widehat{\Phi}$. Let $\widehat{M_a}$ be a new implementation of agent a, and $\widehat{\mathcal{S}} = (I, O, \langle M_1, \ldots \widehat{M_a}, \ldots, M_n\rangle)$ the resulting MAS. In order to check whether $\widehat{\mathcal{S}} \models (ADec, AG)$, it suffices to verify only the condition in Definition 5 pertaining to agent a, with respect to the new implementation $\widehat{M_a}$ of agent a and the original local specification φ^a_{local}. If this is the case, by Theorem 1, we can conclude that the modified system $\widehat{\mathcal{S}}$ satisfies the original specification Φ. Otherwise, the new implementation $\widehat{M_a}$ of agent a is incompatible with the given contract.

Considering the new local specification $\widehat{\varphi}^a_{local}$, if $\widehat{M_a}$ satisfies the condition in Definition 5 with respect to $\widehat{\varphi}^a_{local}$, we can still conclude that $\widehat{M_a}$ satisfies agent a's obligation towards the remaining agents, but not necessarily $\widehat{\mathcal{S}} \models \widehat{\Phi}$. The reason is that if $\widehat{\varphi}^a_{local}$ imposes weaker constraints on the coordination between agents than φ^a_{local}, the old GEDC $(ADec, AG)$ might be overly constraining and the overall system sub-optimal. This is illustrated in the next example.

Example 5.1. *If the local requirements for an agent are tightened in the design process, the agent's new specification might become incompatible with the existing contract, and thus require revision of the contract and the implementations of the other agents. If, on the other hand, the designer relaxes the local requirements, the satisfaction of the existing contract becomes easier. However, the contract might not be a GEDC for the new specification $\widehat{\Phi}$, and the existing implementation of the MAS might no longer be optimal with respect to $\widehat{\Phi}$. This is because in contrast to traditional requirements, where weaker requirements are always satisfied by a stricter implementation, here we impose an optimality criterion. To see this, suppose that in the setting of Example 1.1, the local specification for agent a_1 no longer reduces in value if the vehicle travels in bad weather. That is, $\widehat{\varphi}^1_{local}$ is obtained from φ^1_{local} by dropping the last conjunct. Regardless of the given GEDC, any implementation for agent a_1 that conforms to the original contract clearly still satisfies the conditions of Definition 5 with respect to $\widehat{\varphi}^1_{local}$. However, if the given contract requires agent a_2 to travel during bad weather, the current implementation does not good-enough satisfy the new specification $\widehat{\Phi}$ and the contract is not a GEDC for $\widehat{\Phi}$. The reason is that $\widehat{\varphi}^1_{local}$ allows vehicle a_1 to travel in bad weather and still achieve local satisfaction of 1, making implementations where vehicle a_2 travels in bad weather sub-optimal. Intuitively, relaxing the local specification for a_1 enables the satisfaction of φ^2_{local} with a higher value than stipulated by the GEDC for Φ, resulting in a higher possible satisfaction value for the new specification $\widehat{\Phi}$. As the old contract permits implementations that are not good-enough with respect to $\widehat{\Phi}$, it needs to be revised.*

Example	$\lvert\mathsf{Agt}\rvert$	$\lvert\Phi\rvert$	$\lvert\mathcal{B}_\Phi\rvert$	Runtime		
				Comp.	Rev.	Mono.
delivery_vehicles	3	102	1974	329	24	TO
tasks_collab	4	124	349	50	4	34
tasks_scaled	6	136	547	156	13	289
robots_help_a3	3	46	86	1.5	0.2	1.3
robots_help_a5	5	78	470	15.7	3.1	9.1
synchr_response	3	62	759	439	10	TO

Table 1: Runtime for compositional, local revision and monolithic in seconds, with timeout of 30 minutes.

6 Experimental Evaluation and Conclusion

To evaluate our framework, we implemented the proposed compositional verification procedure in a prototype in Python using the Spot automata library (Duret-Lutz et al. 2022). We ran experiments on 6 MASs each with a number of agents between 3 and 6. The results, obtained on a consumer laptop with 16 GB of memory, are shown in Table 1. Benchmarks are provided in (Dewes and Dimitrova 2024).

Evaluation We evaluated the compositional verification (column "Comp.") performed via checking GEDC satisfaction (including the verification that it is a GEDC for Φ), and compared that to direct monolithic verification of Φ (column "Mono.") by constructing the product system. For the more complex examples, only the compositional algorithm finishes within the time limit, as the automata grow too large, and are limited by memory. In the compositional case, we need to handle more but smaller automata. This addresses the memory consumption but involves more operations. The overhead causes the monolithic approach to be faster when the specifications are small enough. Overall, the modular approach scales better. We report on the verification of local specification and implementation revision (column "Rev."), which, as expected, is faster than full verification.

Conclusion We proposed a framework for modular design and analysis of multi-agent systems, based on good-enough decomposition contracts. It offers greater expressivity compared to existing notions of contracts, and we show that it enhances scalability and modularity of verification. More generally, our approach opens up a new direction in contract-based design of MAS, by extending it to the setting of quantitative specifications and good-enough satisfaction.

References

Almagor, S.; Boker, U.; and Kupferman, O. 2016. Formally Reasoning About Quality. *J. ACM*, 63(3): 24:1–24:56.

Almagor, S.; and Kupferman, O. 2020. Good-Enough Synthesis. In Lahiri, S. K.; and Wang, C., eds., *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, 541–563. Springer.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM*, 49(5): 672–713.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press. ISBN 978-0-262-02649-9.

Benveniste, A.; Caillaud, B.; Ferrari, A.; Mangeruca, L.; Passerone, R.; and Sofronis, C. 2007. Multiple Viewpoint Contract-Based Specification and Design. In de Boer, F. S.; Bonsangue, M. M.; Graf, S.; and de Roever, W. P., eds., *Formal Methods for Components and Objects, 6th International Symposium, FMCO 2007, Amsterdam, The Netherlands, October 24-26, 2007, Revised Lectures*, volume 5382 of *Lecture Notes in Computer Science*, 200–225. Springer.

Benveniste, A.; Caillaud, B.; Nickovic, D.; Passerone, R.; Raclet, J.; Reinkemeier, P.; Sangiovanni-Vincentelli, A. L.; Damm, W.; Henzinger, T. A.; and Larsen, K. G. 2018. Contracts for System Design. *Found. Trends Electron. Des. Autom.*, 12(2-3): 124–400.

Damm, W.; and Finkbeiner, B. 2014. Automatic Compositional Synthesis of Distributed Systems. In Jones, C. B.; Pihlajasaari, P.; and Sun, J., eds., *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, volume 8442 of *Lecture Notes in Computer Science*, 179–193. Springer.

Dewes, R.; and Dimitrova, R. 2023. Compositional High-Quality Synthesis. In André, É.; and Sun, J., eds., *Automated Technology for Verification and Analysis - 21st International Symposium, ATVA 2023, Singapore, October 24-27, 2023, Proceedings, Part I*, volume 14215 of *Lecture Notes in Computer Science*, 334–354. Springer.

Dewes, R.; and Dimitrova, R. 2024. Contract-based Design and Verification of Multi-Agent Systems with Quantitative Temporal Requirements (Extended Version). arXiv:2412.13114.

Duret-Lutz, A.; Renault, E.; Colange, M.; Renkin, F.; Aisse, A. G.; Schlehuber-Caissier, P.; Medioni, T.; Martin, A.; Dubois, J.; Gillard, C.; and Lauko, H. 2022. From Spot 2.0 to Spot 2.10: What's New? In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, volume 13372 of *Lecture Notes in Computer Science*, 174–187. Springer.

Kupferman, O.; Perelli, G.; and Vardi, M. Y. 2016. Synthesis with rational environments. *Ann. Math. Artif. Intell.*, 78(1): 3–20.

Wooldridge, M. J. 2009. *An Introduction to MultiAgent Systems, Second Edition*. Wiley. ISBN 978-0-470-51946-2.