Abstract

We propose CONTRAST, a formal framework for comparing normative multiagent system (nMAS) specifications by computing tradeoffs among liveness (something good happens) and safety (nothing bad happens). Safety-focused specifications restrict agents’ actions to avoid undesired executions. However, such restrictions hinder liveness, particularly in situations such as medical emergencies. Moreover, tradeoffs arise due to limited resources or inherent contradictions. We formalize such tradeoffs using norms, and develop an approach for understanding how much a specification promotes liveness or safety. We propose normative patterns to guide the design of an nMAS with respect to liveness and safety. We propose metrics to measure liveness and safety, and perform experiments on an emergency healthcare scenario using constraint logic programming.

Introduction

Normative multiagent system (nMAS) specifications consist of social norms that regulate how autonomous agents interact with each other as well as control how agents access nonautonomous components. One important problem in such systems is to understand how well the nMAS addresses liveness and safety requirements. Liveness asserts that something good will eventually happen on every enactment, whereas safety asserts that something bad never happens on any enactment (Lamport 1977). An example liveness requirement is to provide guaranteed service for users: physicians should always be able to access patients’ electronic health records (EHRs). An example safety requirement is preserving privacy: patients’ protected health information (PHI) should not be disclosed.

Tradeoffs often arise between liveness and safety (CCC 2015). For example, we may choose whether to preserve privacy or act upon private data, and decide that violating privacy to save a patient’s life is better than preserving the patient’s privacy. Moreover, tradeoffs might occur among liveness (or safety) properties when limited resources prevent agents from fulfilling multiple tasks. Modern ICT applications can partially implement such tradeoffs via dynamic access control policies between agents and software (Marinovic, Duly, and Sloman 2014), e.g., relaxing privacy decisions in emergency situations by allowing a physician to access a patient’s EHR without the patient’s consent. However, they fail to incorporate social interactions among agents, e.g., how should a physician consult a colleague regarding a patient’s PHI so as to protect the patient’s privacy? Therefore, comparing nMAS specifications based on how well they satisfy liveness and safety requirements in various operating modes (e.g., regular practice and medical emergency) is crucial.

Accordingly, we present CONTRAST, a framework for formalizing tradeoffs in nMASs using social norms. Norms have been widely studied (Andrighetto et al. 2013; King et al. 2015; Sergot 2013; Singh 2013), in particular, for compliance verification, monitoring, and revision (Alechina, Dastani, and Logan 2008; Chesani et al. 2013). However, little effort has been spent to evaluate and compare norms for the purposes of understanding tradeoffs between liveness and safety.

We develop an approach for comparing nMAS specifications. We first develop a strength relation to compare norms using their formally stated properties. For example, a norm that prohibits physicians from disclosing patients’ PHI to third parties or sharing it with colleagues is stronger than a norm that only prohibits physicians from disclosing patients’ PHI to third parties. That is, the latter can be replaced with the former to provide better safety. We then generalize the strength relation to sets of norms, and compare nMAS specifications with respect to liveness and safety. Following the above example, an nMAS specification that includes the stronger prohibition is safer than an nMAS specification that includes the weaker one given that all other norms are identical.

Our contributions are as follows: First, we formalize tradeoffs between liveness and safety using the elements of an nMAS specification. Second, we develop an approach for comparing nMAS specifications. Third, we propose design patterns based on norm strength, and prove that they increase liveness or safety provided by an nMAS specification. Fourth, we propose metrics for measuring the liveness and safety of an nMAS specification, and perform experiments using constraint logic programming (CLP) to demonstrate how close a specification is to a perfectly live or safe specification.
**Normative Framework**

An nMAS specification is generated by the grammar given in Table 1, where AG represents an agent identifier, Expr is a logical expression, and φ is an atomic proposition. A specification is a set of capability, action, and norm definitions. Capability cap(AG, φ) means that AG can bring about φ. Action act(AG, Expr, φ) succeeds if the performing agent has the necessary capability and the precondition is satisfied. That is, φ holds if cap(AG, φ) is part of the nMAS specification, and Expr holds.

<table>
<thead>
<tr>
<th>Spec</th>
<th>→</th>
<th>Spec ∪ Spec</th>
<th>Capability</th>
<th>Action</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability</td>
<td>→</td>
<td>cap(AG, φ)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>→</td>
<td>act(AG, Expr, φ)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norm</td>
<td>→</td>
<td>n(AG, AG, Expr, Expr)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Definition 1.** A norm n(X, Y, ant, con) represents a social relationship between its subject (X) and object (Y) regarding its consequent (con) when its antecedent (ant) holds.

Let us see example norms. Authorization (a(PHY, HOS, consent, EHR) means that a physician is authorized by a hospital to access a patient’s EHR if there is consent. Practical commitment c(PHY, HOS, true, operate) means that a physician is (unconditionally) committed to operating on patients. Dialectical commitment d(PHY, HOS, emergency, available) means that a physician is committed to the fact that she will be available when there is an emergency. Prohibition p(PHY, HOS, true, consult ∨ disclose) means that a physician is forbidden from consulting a colleague about patients or disclosing their PHI. Note that PHY and HOS represent agent roles to be instantiated with actual agent labels AG₁ and AG₂. We model five norm states: conditional, expired, detached, satisfied, and violated.¹

Next, we present reasoning postulates regarding norms adapted from Chopra and Singh (2009). We omit the agents or the specific norm type whenever they are not relevant, i.e., writing a norm as n(ant, con), where n stands for a, c, d, p.

**Postulate 1.** n(r ∨ s, u) if and only if n(r, u) and n(s, u)

**Postulate 2.** a(r, u ∨ v) if and only if a(r, u) and a(r, v)

**Postulate 3.** c(r, u ∨ v) if and only if c(r, u) and c(r, v)

**Postulate 4.** d(r, u ∨ v) if and only if d(r, u) and d(r, v)

**Postulate 5.** p(r, u ∨ v) if and only if p(r, u) and p(r, v)

¹See supplementary material for details of the norm lifecycle.

Postulate 1 covers cases where multiple norms of the same type with the same consequent can be combined into a single norm. Postulates 2–5 cover cases of multiple norms with the same antecedent and different consequents.

Φ is the set of all domain propositions including the effects of agents’ actions. A normative specification, N, is the union of the sets of authorizations (A), commitments (C and D for practical and dialectical, respectively), and prohibitions (P). N captures the social specification of an nMAS. Note that agent capabilities and actions listed in Table 1 are not included in the normative specification as they are not relevant to our further development for comparing nMASs.

For pairwise comparison of norms and formal understanding of which norms can replace others, we adopt Chopra and Singh’s (2009) strength relation for commitments and prohibitions (P). Below, ⊢ represents logical consequence.

**Definition 2.** Authorizations: a₁ = a(X, Y, s, v), denoted a₁ ≫ a₂, if and only if s ⊢ r and v ⊢ u. Practical commitments: c₁ = c(X, Y, s, v) ≫ c₂ = c(Y, s, v) if and only if s ⊢ r and v ⊢ r. Dialectical commitments: d₁ = d(X, Y, r, u) ≫ d₂ = d(Y, s, v) if and only if s ⊢ r and u ⊢ v. Prohibitions: p₁ = p(X, Y, s, v) if and only if s ⊢ r and v ⊢ u.

Table 2 shows examples of norm comparison with the strength (≫) relation. We can verify that ≫ is a partial order, i.e., it is reflexive, antisymmetric, and transitive. Thus, n₁ = n₂ if n₁ ≫ n₂ and n₂ ≫ n₁.

**Table 2: Comparison of norm strength.**

| a(authenticate ∨ consent, EHR) ≫ a(consent, EHR) |
| c(true, operate ∧ clinic) ≫ c(true, operate) |
| d(true, available ∧ expert) ≫ d(emergency, available) |
| p(true, consult ∨ disclose) ≫ p(emergency, consult) |

Normative specifications are closed under norm strength, e.g., a₁ ∈ A if a₁ ∈ A and a₁ ≫ a₂.

**Example 1.** Let a₁ = a(PHY, HOS, authenticate ∨ consent, EHR) ∈ A₁. Then, a₂ = a(PHY, HOS, authenticate, EHR) ∈ A₁ because a₁ ≫ a₂. Similarly, a₃ = a(PHY, HOS, authenticate ∧ own_patient, EHR) ∈ A₁ and a₄ = a(PHY, HOS, authenticate, EHR ∧ operate) ∈ A₁ because a₂ ≫ a₃ and a₂ ≫ a₄.

**Definition 3.** Norm n is maximal in N, denoted n ∈ max(N), if and only if ∀ n′ ∈ N: n′ ⋄ n ≪ n.

Definition 3 defines maximal norms in a given normative specification. Consider the authorizations in Example 1 and let A₁ be {a₁, a₂, a₃, a₄}. Then, a₁ ∈ max(A₁).

**Understanding Tradeoffs**

We represent liveness and safety requirements as the following distinct sets of atomic propositions: (i) desired properties
(Rₐ) that need to be achieved (liveness), (ii) undesired properties (Rᵤ) that need to be avoided (safety), and (iii) control properties (Rₑ) that enforce constraints on agents’ actions (safety). Liveness and safety requirements often compete with themselves as well as each other. Examples 2 and 3 describe HIPAA (US Health Insurance Portability and Accountability Act) scenarios pertaining to regular and emergency medical practices (HHS 2014).

**Example 2.** Hospitals are bound by law to secure their patients’ PHI. Accordingly, Hospital A requires its physicians to authenticate with the hospital’s servers (safety) before accessing patients’ EHRs (liveness), and prohibits its physicians from operating upon patients (liveness) or disclosing their PHI without consent (safety).

Example 2 describes the medical practice adopted by a hospital in order to comply with the law. The authentication constraint controls access to EHRs. A prohibition provides accountability for physicians regarding the disclosure of patients’ PHI. Note that these safety regulations would not affect liveness under regular medical practice, but in medical emergencies.

**Example 3.** There is a public emergency near Hospital A, and several unconscious patients need to be operated upon immediately. There are two aspects to consider: there is no opportunity for obtaining consent, and Hospital A does not have the required number of physicians on staff to attend to the emergency situation.

Example 3 describes a tradeoff between liveness and safety. Note that in regular practice, only staff of Hospital A has consent to operate upon the patients. Following the guidelines of The American College of Emergency Physicians (ACEP) for disasters (ACEP 2013), Hospital A may waive the consent requirement for procedures and assign credentials to outside physicians to cope with the load (increasing liveness). However, allowing outside physicians to access patients’ EHRs is a safety concern (i.e., not having a control property to restrict access to EHRs).

Figure 1 depicts such tradeoffs between liveness and safety. Each point represents an nMAS specification. The dashed curve corresponds to specifications that are optimal in that an increase in one dimension would decrease the other. Let us review some important cases.

**Suboptimal** describes a non-ideal specification. Consider authorization a(PHY, HOS, authenticate ∧ expert, operate). Expert physicians who are authenticated can operate upon patients. This specification provides liveness and safety for regular practice. However, liveness is decreased in emergencies due to the restrictions on physicians, e.g., authentication is not possible for outside physicians.

**Loss** describes a transition to a specification in which one dimension is decreased without affecting the other (e.g., from Suboptimal to Diminished). Consider a(PHY, HOS, true, disclose). Liveness is not affected, but safety is decreased since physicians are allowed to disclose patients’ PHI.

**Gain** describes a transition to a specification in which one dimension is increased without affecting the other (e.g., from Suboptimal to Enhanced). Consider norms a(PHY, HOS, state_authenticate, operate) and d(PHY, HOS, true, expert). Physicians from other hospitals may perform surgical procedures (state_authenticate represents state-wide authentication). Liveness is increased due to the utilization of outside physicians. Safety is preserved with the dialectical commitment from physicians for being experts, which can be used to hold physicians accountable for their operations on patients.

**Boost** describes a transition to a specification in which both dimensions are increased (e.g., from Suboptimal to Ideal). Consider norms a(PHY, HOS, credentials, operate) and d(PHY, HOS, true, expert). Physicians from international hospitals can operate upon patients, which increases liveness. However, safety is increased due to the prohibition for restricting access to EHRs without credentials.

**Tradeoff** describes a transition between optimal specifications, where only one dimension can be increased. Consider authorization a(PHY, HOS, credentials ∨ international, operate). Physicians from international hospitals can operate upon patients, which increases liveness. However, safety is decreased due to difficulty in sanctioning international physicians in case a misuse happens.

### Comparing nMAS Specifications

An operating mode M describes agents’ capabilities for a specific situation (i.e., the effects of actions that can be performed in M). For brevity, we omit the agents from the capability set and only include the propositions (see Table 1) when describing operating modes. In regular practice (reg), all actions can be performed (reg = Φ). However, in a medical emergency (emg), patients cannot give consent (emg = Φ \ {consent}). And, when there is a server failure (srv), physicians cannot authenticate (srv = Φ \ {authenticate}).

We begin describing how to compare nMAS specifica-
tions by defining whether a specification enables a proposition. Definition 4 describes propositional enablement, \(N \xrightarrow{M} \phi\), which means that \(N\) enables proposition \(\phi\) in mode \(M\). First, \(\phi\) must be a member of \(M\) (i.e., \(\phi\) is possible in \(M\)). Second, there must be an authorization whose antecedent is enabled (i.e., the authorization can be detached), and whose consequent is logically entailed by the proposition. Third, there must not be a prohibition that can be detached whose consequent is entailed by the proposition.

**Definition 4.** \(N \xrightarrow{M} \phi\) if and only if
\[
\begin{align*}
&\bullet \ \phi \in M, \text{ and} \\
&\bullet \ \exists a_i = a(x_i, y_i, \text{ant}_i, \text{con}_i) \in N: \phi \vdash \text{true} \text{, and either ant}_i \\
& \quad = \text{true or } N \xrightarrow{M} \text{ant}_i, \text{ and} \\
&\bullet \ \exists p_j = p(x_j, y_j, \text{ant}_j, \text{con}_j) \in N: \phi \vdash \text{con}_j \text{, and either ant}_j \\
& \quad = \text{true or } N \xrightarrow{M} \text{ant}_j.
\end{align*}
\]

\(N \xrightarrow{M} (r \land s)\) if and only if \(N \xrightarrow{M} r\) and \(N \xrightarrow{M} s\). \(N \xrightarrow{M} (r \lor s)\) if and only if \(N \xrightarrow{M} r\) or \(N \xrightarrow{M} s\).

**Example 4.** Let \(N_1\) be \(\{a(\text{PHY}, \text{HOS}, \text{consent}, \text{EHR}), a(\text{PAT}, \text{HOS}, \text{true}, \text{consent})\}\). Then, \(N_3 \xrightarrow{rev} \text{EHR}\) and \(N_1 \xrightarrow{req} \text{EHR}\).

Next, we describe how CONTRAST compares normative specifications using the above formalization. We begin with liveness: \(N_i \geq^M N_j\) means that \(N_i\) is at least as live as \(N_j\) in mode \(M\). Definition 5 compares two specifications \(N_i\) and \(N_j\) in mode \(M\) with respect to liveness requirements (\(R_i\)), and concludes that \(N_i \geq^M N_j\) if two conditions hold: (i) when there is a commitment in \(N_j\) whose consequent entails \(l\), there must be a stronger commitment in \(N_i\), and (ii) when \(N_j\) enables \(l\), \(N_i\) must enable \(l\) as well.

**Definition 5.** \(N_i \geq^M N_j\) if and only if \(\forall l \in R_d:\)
\[
\begin{align*}
&\bullet \ \exists c_j = c(x, y, \text{ant}, \text{con}) \in N_j: \text{con} \vdash l \text{, and either ant} \\
& \quad = \text{true or } N_j \xrightarrow{M} \text{ant}, \text{ and then } \exists c_i \in N_i: c_i \triangleright c_j, \\
&\bullet \ \text{if } N_j \xrightarrow{M} l \text{, then } N_i \xrightarrow{M} l.
\end{align*}
\]

Observe that \(\geq^M\) is a partial order. Thus, \(N_i = N_j\) if \(N_i \geq^M N_j\) and \(N_j \geq^M N_i\). We write \(N_i \succ^M N_j\) (\(N_i\) is more live than \(N_j\) in mode \(M\)) if and only if \(N_i \geq^M N_j, N_j \not{\succ^M} N_i\).

**Example 5.** Consider the following specifications: \(N_2 = \{a(\text{PHY}, \text{HOS}, \text{authenticate}, \text{prescription}), a(\text{PHY}, \text{HOS}, \text{true}, \text{authenticate}), c(\text{PHY}, \text{PAT}, \text{diagnosis} \lor \text{request}, \text{prescription}), a(\text{PAT}, \text{HOS}, \text{true}, \text{request})\}\). \(N_3 = \{a(\text{PHY}, \text{HOS}, \text{true}, \text{authenticate}), c(\text{PHY}, \text{PAT}, \text{diagnosis} \lor \text{request}, \text{prescription}), a(\text{PAT}, \text{HOS}, \text{true}, \text{request})\}\).

We have one desired property for Example 5, \(R_d = \{\text{prescription}\}\), meaning that physicians must prescribe drugs to patients. Let us compare \(N_2\) and \(N_3\) in two modes: \(\text{reg}\) and \(\text{srv}\). In regular practice, \(N_3\) enables prescribing drugs according to Definition 4, \(N_3 \xrightarrow{req} \text{prescription}\). Similarly, \(N_2 \xrightarrow{req} \text{prescription}\), since the antecedent of the authorization (authenticate) is possible in the mode, \(\text{reg} = \Phi\). Moreover, both specifications have the same commitment toward prescription of drugs. According to the first condition of Definition 5, the antecedent of the commitment must be enabled. \(N_3 \xrightarrow{req} \text{request}\) since there is an authorization, and we can infer \(N_3 \xrightarrow{req} \text{(diagnosis} \lor \text{request)}\) using the disjunction rule in Definition 4. Thus, \(N_2 \geq^M N_3\). When there is a server failure, \(N_3\) still enables prescription of drugs, since the antecedent of the authorization is true. However, \(N_2\) does not enable prescription of drugs any more, since authentication is not possible (\(\text{srv} = \Phi \setminus \{\text{authenticate}\}\)). Thus, we conclude that \(N_2 \not{\geq^M} N_3\) and \(N_3 \not{\geq^M} N_2\). Therefore, \(N_3 \geq^M N_2\).

We continue with safety: \(N_i \geq^M N_j\) means that \(N_i\) is at least as safe as \(N_j\) in mode \(M\). Definition 6 compares two specifications \(N_i\) and \(N_j\) in mode \(M\) with respect to safety requirements (\(R_u\) and \(R_c\)), and concludes that \(N_i \geq^M N_j\) if two conditions hold: (i) for each undesired property \(s\), when \(N_j\) enables \(s\), \(N_i\) must enable \(s\) as well. That is, when \(N_i\) allows a bad thing to happen, then \(N_j\) must allow the same bad thing to happen, and (ii) for each control property \(s'\), when there is an authorization in \(N_j\) whose antecedent entails \(s'\), there must be a weaker authorization in \(N_i\). That is, if \(N_j\) imposes a restriction on some property, then \(N_i\) must impose a stronger restriction (weaker authorization).

**Definition 6.** \(N_i \geq^M N_j\) if and only if
\[
\begin{align*}
&\bullet \ \forall s \in R_u: \text{if } N_i \xrightarrow{M} s, \text{ then } N_j \xrightarrow{M} s, \text{ and} \\
&\bullet \ \forall s' \in R_c: \text{if } \exists a_j = a(x, y, \text{ant}, \text{con}) \in N_j: \text{ant} \vdash s', \text{ then} \\
& \quad \exists a_i \in N_i: a_j \triangleright a_i.
\end{align*}
\]

Observe that \(\geq^M\) is a partial order. Thus, \(N_i = N_j\) if \(N_i \geq^M N_j\) and \(N_j \geq^M N_i\). We write \(N_i \succ^M N_j\) (\(N_i\) is safer than \(N_j\) in mode \(M\)) if and only if \(N_i \geq^M N_j, N_j \not{\succ^M} N_i\).

**Example 6.** Consider the following specifications: \(N_4 = \{a(\text{PHY}, \text{HOS}, \text{authenticate} \land \text{consent}, \text{EHR})\}\). \(N_5 = \{a(\text{PHY}, \text{HOS}, \text{authenticate}, \text{EHR})\}\).

We have one control property for Example 6, \(R_c = \{\text{authenticate}\}\), meaning that physicians must be authenticated. Let us compare \(N_4\) and \(N_5\). \(N_4\) has an authorization whose antecedent entails authentication. \(N_5\) must have a weaker authorization to be at least as safe as \(N_4\) according to Definition 6. However, the authorization in \(N_5\) is stronger than the authorization in \(N_4\). Thus, we conclude that \(N_5 \not{\geq^M} N_4\) and \(N_4 \geq^M N_5\). Therefore, \(N_4 \not{\geq^M} N_5\).

**nMAS Design via Patterns**

We propose normative patterns to guide nMAS design.

**Construction Patterns**

The following design patterns create norms based on the liveness and safety requirements.

**Achievement pattern** creates authorization \(a(X, Y, \text{true}, r_d)\) with respect to a desired property \(r_d\).

**Avoidance pattern** creates prohibition \(p(X, Y, \text{true}, r_u)\) with respect to an undesired property \(r_u\).

**Fortify pattern** replaces authorization \(a_j = a(X, Y, s, v)\) with \(a_j = a(X, Y, s \land r_c, v)\) with respect to a control property \(r_c\).
Note that the fortify pattern can only be applied on an existing specification, whereas the achievement and avoidance patterns can be used to design an nMAS from scratch.

Tradeoff Patterns

The following refinement patterns implement tradeoffs between specifications. That is, applying a refinement pattern increases one dimension (liveness or safety), but might decrease the other.

Flexibility pattern transforms specification N into N′, N ⊨ N′. Let l be a desired property (i.e., l ∈ R_d) and φ be a proposition that is not a member of some mode M (i.e., φ /∈ M). N′ is constructed from N as follows:

\[ N' = N \cup \{ a_i = a(x, y, \text{ant}, \text{con}) | l \vdash \text{con}, \text{ant} \vdash \phi, a_i \in \text{max}(N) \} \]

\[ \{ p_k = p(x_k, y_k, \text{ant}_k, \text{con}_k) | l \vdash \text{con}_k \} \]

Applying the flexibility pattern on a specification ensures that a desired property that is not originally enabled in mode M becomes enabled in M. This is done by adding an authorization that is stronger than the maximal authorization (which provides more flexibility) and removing all the prohibitions that prevent enablement of the property.

Theorem 1. If N ⊨ N′, ∃ M: N′ ⊳^M N and ∀ M: N′ ⊳^M N.

Let us revisit Example 5. The flexibility pattern adds \( a(\text{PHY, HOS, true, prescription}) \) in N_3. This is a stronger authorization than \( a(\text{PHY, HOS, authenticate, prescription}) \) in N_2, and N_3 ⊳^opt N_2.

Restraint pattern transforms specification N into N′, N ≺ N′. Let s be a control property (i.e., s ∈ R_c). N′ is constructed from N as follows:

\[ N' = N \cup \{ a_i = a(x, y, \text{ant, con}) | \text{ant} \vdash s, a_i \in \text{max}(N) \} \]

Applying the restraint pattern on a specification provides additional safety for a control property by removing the maximal authorization, and allowing a weaker (more restrictive) authorization to take its place due to closure under norm strength.²

Theorem 2. If N ≺ N′, ∀ M: N′ ⊳^M N.

Let us revisit Example 6. The restraint pattern removes \( a(\text{PHY, HOS, authenticate, EHR}) \) from N_5. The weaker authorization \( a(\text{PHY, HOS, authenticate} \land \text{consent, EHR}) \) is left in N_4, and N_4 ⊳^M N_5.

Combining the Patterns

Patterns are used in sequence to guide the design of an nMAS with respect to the liveness and safety requirements. For example, assume we have one desired property \( R_d = \{ \text{EHR} \} \) and one control property \( R_c = \{ \text{authenticate} \} \). An authorization \( a(\text{PHY, HOS, true, EHR}) \) can be created by applying the achievement pattern, and replaced with \( a(\text{PHY, HOS, authenticate \land consent, EHR}) \) by applying the fortify pattern. The resulting authorization corresponds to specification N_5 from

³See supplementary material for proofs of Theorems 1 and 2.

Example 6. Now, the authorization can be refined to \( a(\text{PHY, HOS, authenticate} \lor \text{emergency, EHR}) \) by applying the flexibility pattern so that liveness is increased because physicians are authorized to access patients’ EHRs in emergencies. As an alternative, the authorization can be refined to \( a(\text{PHY, HOS, authenticate} \land \text{consent, EHR}) \) by applying the restraint pattern so that safety is increased because physicians need consent from patients to access their EHRs.

CLP Experiments

We adopt ECLiPSe, a constraint logic programming framework (Apt and Wallace 2007), to perform experiments with various nMAS specifications. ECLiPSe offers a conceptual modeling language that extends Prolog and provides constraint solver libraries for solving integer constraints. We implement tradeoff scenarios to compare the level of liveness and safety various specifications provide. We perform experiments by generating integer costs for surgical procedures to compare two nMASs: a Suboptimal specification and an Enhanced specification.³ We propose two metrics to measure the liveness and safety provided by an nMAS specification in the hospital setting.

Liveness score measures the distance of a given nMAS specification from a perfectly live specification, which represents a (possibly fictitious) specification with no constraints on any surgical procedure. We compute liveness as follows:

\[ \text{Liveness score} = \frac{\text{supported procedures}}{\text{all procedures}} \]

Safety score measures the distance of a given nMAS specification from a perfectly safe specification, which represents a specification where outside physicians are not allowed to perform any surgical procedure. That is, the safety score measures how many bad alternatives are avoided. We compute safety as follows:

\[ \text{Safety score} = 1 - \frac{\text{procedures by outside physicians}}{\text{supported procedures}} \]

Note that the liveness and safety scores are not necessarily complements of each other, i.e., they do not add up to one.

Table 3 compares the results between two specifications, Suboptimal and Enhanced, in three modes of operation. Suboptimal ⊳^M Enhanced in all modes since the Suboptimal specification puts more security restrictions on physicians. More specifically, it does not allow outside physicians to perform any surgical procedures, thus leading to a perfect safety score. In regular practice, Enhanced ⊳^ref Suboptimal because the relaxation conditions of the Enhanced specification only work in nonregular practice. In regular practice, both specifications support nine out of the 48 procedures supported by a perfectly live specification, leading to a liveness score of 0.19. Thus, Suboptimal ⊳^ref Enhanced.

In a medical emergency, it is expected that the demand for surgical procedures increases. The Suboptimal specification fails to meet this demand, supporting only five alternatives.

³See supplementary material for the CLP implementation.
leading to a liveness score of 0.10. The Enhanced specification, on the other hand, supports 35 alternatives, leading to a liveness score of 0.73. This means that it covers more desired properties (Rd) described in terms of surgical procedures. Therefore, Enhanced $\succ_{\text{emg}}$ Suboptimal. However, the safety score of the Enhanced specification for the medical emergency case is low (0.14), since most of the procedures supported are performed by outside physicians who are not authenticated in Hospital A. Therefore, Suboptimal $\succ_{\text{srv}}$ Enhanced.

When there is a server failure and authentication is not possible, no procedures are supported by the Suboptimal specification, whereas the Enhanced specification supports ten alternatives, leading to a liveness score of 0.21. Thus, Enhanced $\succ_{\text{srv}}$ Suboptimal. However, the Enhanced specification is completely unsafe for the server failure case, since all the procedures supported are performed by outside physicians. Therefore, Suboptimal $\succ_{\text{srv}}$ Enhanced.

### Discussion

We developed CONTRAST, an approach for comparing nMAS specifications. We proposed normative patterns to design an nMAS with respect to liveness and safety, and provided experiments using constraint logic programming. The example metrics we have proposed for the healthcare scenarios can be adapted to various other domains, e.g., in consumer banking, liveness corresponds to how easily customers can make payments, whereas safety corresponds to absence of false payments. Such metrics can be used to guide the nMAS design process. It would be interesting to apply CONTRAST to systems where tradeoffs are not obvious.

Sergot (2013) discusses the correspondence of normative relations among agents with Hohfeldian legal concepts such as duties and rights, and presents semantics using deontic logics. Alechina et al. (2013) focus on conditional norms with deadlines, and extend CTL and ATL with sanctions to reason about the effects of normative update. They measure norm compliance by verifying if specific states are reached before the deadline, and enforce norms via sanctions. King et al. (2015) propose a similar hierarchical governance model for institutions using a multi-tier normative system using answer set programming. We go beyond verification of compliance, and provide pairwise comparison of normative specifications.

Kafali and Yolum (2016) propose an approach for monitoring an agent’s interactions to determine whether the agent is progressing as expected. In particular, they verify whether the agent’s expectations (represented by a set of propositions and commitments) are satisfiable by its current state. Governatori (2013) proposes a conceptual abstract framework to model normative requirements, formalizes different types of obligations, and verifies whether a business process is compliant with requirements (set of obligations). The above works are limited to the representation and verification of commitments and obligations. While verification is important, it is not sufficient to understand whether a specification does not satisfy its requirements. Therefore, we expand on verification, and provide a metrics for measuring the liveness and safety of specifications.

Vasconcelos et al. (2009) propose methods for resolving conflicts among norms. Their resolution method, norm curtailment, manipulates the constraints associated with norms, e.g., reduce the scope of a prohibition to avoid conflict with an obligation. Our propositional enablement relation detects such conflicts in an nMAS specification via the authorization or prohibition checks. Zhang et al. (2016) discuss the application of probabilistic commitments in open environments with uncertainty. Extending CONTRAST with probabilistic dialectical commitments is an interesting direction, e.g., a physician commits to a patient having cancer with 80% certainty.

Artikis (2009) proposes an infrastructure to specify dynamic protocol specifications for open multiagent systems. Specifications are modeled as metric spaces, and the infrastructure enables agents to specify protocols at design-time, modify protocols at run-time. According to Artikis’ metric, protocols are evaluated based on the distance between specifications at two distinct time points. In a sense, our metrics resemble theirs since we measure the distance from a perfectly live or safe specification. However, their metric space does not cover any liveness and safety requirements.

Role-based access control (RBAC) mechanisms (Ferraiolo, Kuhn, and Chandramouli 2007; Brucker and Petritis 2009) provide flexibility in accessing patient information in healthcare systems. However, they suffer from a common drawback, namely, lack of computational mechanisms to formally compute tradeoffs as we do in our work.

Our work opens up interesting directions for future research. Extending our framework with sanctions (Dastani et al. 2009; Nardin et al. 2016) would add another dimension to the normative comparison. Sanctions provide compensation for norm violations (liveness) as well as deterrence against violating norms (safety). Moreover, it would be important to investigate the computational complexity of CLP for specifications containing large sets of norms.

<table>
<thead>
<tr>
<th>Mode of operation</th>
<th>Suboptimal Liveness score</th>
<th>Suboptimal Safety score</th>
<th>Enhanced Liveness score</th>
<th>Enhanced Safety score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular practice</td>
<td>0.19</td>
<td>1.00</td>
<td>0.19</td>
<td>1.00</td>
</tr>
<tr>
<td>Medical emergency</td>
<td>0.10</td>
<td>0.73</td>
<td>1.00</td>
<td>0.14</td>
</tr>
<tr>
<td>Server failure</td>
<td>0.00</td>
<td>0.21</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 3: Liveness and safety scores for comparing the Suboptimal and Enhanced specifications.
References


