A Semantic Framework for Socially Adaptive Agents
Towards strong norm compliance

M. Birna van Riemsdijk*
Delft University of Technology,
The Netherlands
m.b.vanriemsdijk@tudelft.nl

Louise Dennis†
University of Liverpool, UK
l.a.dennis@liverpool.ac.uk

Michael Fisher
University of Liverpool, UK
mfisher@liverpool.ac.uk

Koen V. Hindriks
Delft University of Technology
k.v.hindriks@tudelft.nl

ABSTRACT

We address the question of how an agent can adapt its behavior to comply with newly adopted norms. This is particularly relevant in the case of open systems where agents may enter and leave norm-governed social contexts not known at design time. This requires norms to be explicitly and separately stated and presented to an agent as rules to which it then can try to adapt its behavior.

We propose a formal semantic framework that specifies an execution mechanism for such socially adaptive agents. This framework is based on expressing norms using Linear Temporal Logic. The formality of the framework allows us to rigorously study its norm compliance properties. A weak form of norm compliance allows agents to abort execution in order to prevent norm violation. In this paper we investigate a stronger notion of norm compliance that is evaluated over infinite traces. We show that it is not possible for all agents to be strongly compliant with any arbitrary set of norms. We then investigate situations when strong norm compliance can be guaranteed.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures, multiagent systems*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; F.3.2 [**Logics and Meaning of Programs**]: Semantics of Programming Languages

General Terms

Theory, Languages

Keywords

Norm-aware agents; Agent programming languages; Norm compliance; Formal semantics; Executable temporal logic

*First author was supported by the NWO Vidi project CoreSAEP.

†Second and third authors supported by EPSRC Grant EP/L024845/1, "Verifiable Autonomy".

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

1. INTRODUCTION

Norms have been proposed as a way of regulating multi-agent systems in order to prevent undesirable or ineffective behavior [3]. Norms describe how agents should ideally behave. When agents enter a social context that is governed by a set of norms, they should be able to recognize the norms, decide whether they want to follow them, and if they decide to do so, adapt their behavior accordingly to generate norm-compliant behavior.

In this paper we address the latter issue, focusing on the following question: *how can agents adapt their behavior to comply with newly adopted norms?* We refer to such agents as *socially adaptive agents*. Creating socially adaptive agents is particularly relevant in the case of open systems [13] where agents may enter and leave norm-governed social contexts not known at design time. This means they have to be able to adapt to new norms at run-time. Examples are social robots and personal assistant agents that support people in their daily lives in various social contexts.

Our aim is to develop an *execution mechanism* for socially adaptive agents that can be integrated into existing agent programming languages [5]. As argued in [32], treating adopted norms "simply as additional beliefs, goals, or rules that the agent may adopt at runtime is insufficient, as it will be difficult (if not impossible) to do it in such a way that norm compliant behavior is generated". Since it is not known at design time which combinations of norms may arise, we need an execution mechanism that can adapt the agent's proactive and reactive behavior to arbitrary sets of norms (in some normative language) at run-time. Moreover, agent behavior should not be adapted if this is not required according to norms.

In this paper we propose a formal *semantic framework* that specifies such an execution mechanism for socially adaptive agents. Formally specifying the semantics allows us to rigorously study its norm compliance properties. In previous work [32] we proposed an execution semantics for agents that had the option of aborting in order to prevent norm violation (weak norm compliance). Compliance of an execution trace with norms was evaluated up to the point where a norm violation was about to occur. In this paper, we investigate a stronger notion that evaluates compliance over infinite traces. We argue that strong norm compliance corresponds to an intuitive idea of norm compliant behavior. Strong norm compliance requires that any (infinite) normative agent trace complies with norms. Not all agents can be strongly norm compliant with any set of norms, for example when norms are inconsistent. We investigate when strong norm compliance *can* be guaranteed.

We start by discussing related work on reasoning about norm

compliance and propose an architecture for socially adaptive agents that shows how the execution mechanism can be integrated with other types of norm compliance reasoning (Section 2). We use an *abstract decision mechanism* to define an agent's proactive and reactive behavior, and express norms using the next-fragment of *Linear Temporal Logic (LTL)* (Section 3). An agent satisfies a norm expressed as an LTL formula if it generates behavior that satisfies the formula. The main contributions of this paper are the following:

- A *normative agent semantics* that specifies how the agent's decision mechanism is adapted to comply with norms (Section 4). We introduce a number of desired properties of normative agent semantics and show that our semantics satisfies them.
- A definition and semantic exploration of *strong norm compliance* (Section 5). We show that satisfiability of norms corresponds to the existence of a strongly norm compliant agent, provide a semantic characterization of strong norm compliance by proving that it coincides with absence of conflict, and identify sets of "safe" norms that guarantee strong norm compliance.

2. NORM COMPLIANCE REASONING

We position our work in the broader area of reasoning about norm compliance by discussing related work (Section 2.1) and introducing an architecture to show how different types of reasoning about norm compliance can be integrated (Section 2.2).

2.1 Related Work

The term norm compliance has been used in different ways and in different contexts in the literature. There has been considerable research on how agents can *reason about whether they want to comply* with norms by determining whether the benefits from complying outweigh the expected costs of violation (e.g., induced by sanctions) or if complying is instrumental in achieving the agent's goals [9, 4, 17, 26, 27, 31, 1, 11]. Our aim is complementary: we focus on generating behavior that complies with norms, *once the agent has decided it wants to comply* with them.

In the context of business process modelling, techniques have been developed for *verifying whether a business process complies with a set of norms*, for example through model checking [22, 21, 23, 30, 12, 15]. These approaches are aimed at compliance checking at design time. In [21] the declarative representation of norms and goals for a business process is proposed in a modal defeasible theory. Compliance by design can be achieved if from this one creates a new process that satisfies goals and norms. There is also research on run-time verification of process models [28], in which business processes are monitored during execution in order to detect (potential) problems. Monitoring is also investigated in the area of normative multi-agent systems [7, 10, 8] in which mechanisms for detecting norm violations are developed. While verification and monitoring approaches *check* whether a given agent or business process complies with norms, our aim is to develop an execution mechanism that can *generate* norm compliant behavior (while preserving the original agent behavior if possible). In [21] it is suggested to generate a compliant business process from a declarative specification. This is different from defining an execution mechanism that generates norm compliant behavior by *adapting a given procedural specification*.

In [29] another technique is proposed that allows agents to adapt their behavior to newly adopted norms. The main difference with our work is that they do not specify norm compliance and the execution mechanism formally and consequently cannot formally investigate norm compliance properties. In [31] a norm-based planning approach is proposed for generating agent behavior that maximizes an agent's utility. Planning can be used to generate norm compliant behavior. However, it requires that agents are modelled as planners which is not always desirable. In particular, planning assumes a finite horizon which our semantic framework does not. We define an operational semantics of what it means that an agent adapts its behavior to norms, which facilitates integration of our execution mechanism into agent programming languages.

2.2 Architecture

In this section we illustrate how we envisage the execution mechanism for generating norm compliant behavior to be integrated into an overall architecture for socially adaptive agents. The architecture (Figure 1) shows how different types of reasoning about norm compliance, as discussed in Section 2.1, can be integrated.

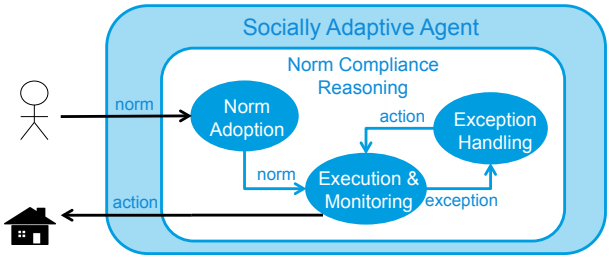

Figure 1: Socially Adaptive Agent Architecture

When an agent identifies new norms at run-time (called *normative beliefs* in [2]), it has to decide whether to adopt them by reasoning about whether it wants to comply. Once norms have been adopted (called *normative goals* in [2]), the agent's execution mechanism should ensure that norm compliant behavior is generated (if possible). This is the focus of this paper. Monitoring can be used to detect if norm violations (exceptions) unexpectedly occur. Exceptions may occur if the execution mechanism cannot fully guarantee norm compliant behavior. In this paper we analyze to what extent the execution mechanism can provide such guarantees. If a norm violation occurs, an exception handling process needs to be in place to determine the best way to respond. Exception handling is related to contrary-to-duty reasoning [24, 22], which concerns the derivation of new norms that should be fulfilled to repair or compensate for violation of the primary norm. Contrary-to-duty reasoning is outside the scope of this paper. Our architecture is in line with that proposed in [29].

3. PRELIMINARIES

We use the abstract agent semantics (Section 3.1) and normative language (Section 3.2) from [32], which we summarize here.

3.1 Abstract Agent Semantics

The semantic framework we propose comprises three "layers": 1) a transition function $\mathcal{T}$ specifying how the execution of an action changes a state, representing the actions that an agent can execute 2) an agent decision mechanism *Dec* specifying which actions enabled by $\mathcal{T}$ an agent would choose to execute if there were no norms in place, and 3) a normative agent semantics $[\![A]\!]_N$ specifying how *Dec* is adapted to comply with norms N. *Dec* can be generated, for example, through an agent program that specifies the agent's reactive and proactive behavior. We here abstract from how

Dec is generated to ensure that our normative agent semantics can be integrated with various agent programming frameworks.

We assume two mutually disjoint sets: a set of propositional atoms At with typical element p over which properties of states are defined, and a set of agent actions Act with typical element a. We assume a language of propositional logic $\mathcal{L}_{\mathsf{At}}$ with typical element ϕ defined over a finite set of atoms At, with $\top, \bot \in \mathcal{L}_{\mathsf{At}}$ denoting the true and false sentence, respectively. We also assume a finite set of abstract agent states S and an entailment relation $s \models_{\mathcal{L}_{\mathsf{At}}} \phi$, which defines when ϕ holds in s, such that for each $\phi \in \mathcal{L}_{\mathsf{At}}$ with $\phi \not\models_{\mathcal{L}_{\mathsf{At}}} \bot$ there is a state $s \in S$ such that $s \models_{\mathcal{L}_{\mathsf{At}}} \phi$. Moreover, states are complete, i.e., for each state $s \in S$ and $p \in \mathsf{At}$ either $s \models_{\mathcal{L}_{\mathsf{At}}} p$ or $s \models_{\mathcal{L}_{\mathsf{At}}} \neg p$. This is important to prove correspondence of the semantics with LTL. We also assume atoms are independent, i.e., for any two atoms, p and q both $p \wedge q$ and $p \wedge \neg q$ are consistent[1]. We provide no formal definition for $\models_{\mathcal{L}_{\mathsf{At}}}$ abstracting from this for the same reasons we abstracted from *Dec*.

The partial function $\mathcal{T}$ of type : $(\mathsf{Act} \times S) \rightarrow S$ specifies how the execution of an action changes a state. If an action a cannot be executed in state s, $\mathcal{T}(a, s)$ is undefined. We assume an abstract agent decision mechanism $Dec(\mathsf{Act}, S, \mathcal{T})$ as a set of transitions $s \xrightarrow{a} s'$ with $s, s' \in S$ and $a \in \mathsf{Act}$ for which $\mathcal{T}(a, s) = s'$. This set of transitions defines which actions may be executed in which state according to the agent's decision mechanism. This is a subset of the transitions that can occur according to $\mathcal{T}$.

3.2 Language of Norms

We use Linear Temporal Logic (LTL) [16] to specify norms and thus our syntax of norms is that of LTL. In order to comply with a norm expressed as an LTL formula, agents should exhibit behavior that satisfies the LTL formula. An important advantage of using LTL for specifying norms is that LTL is an expressive formal language for specifying properties of traces of computational systems. Norm compliance directly translates to satisfaction of the LTL formulas over the traces that the agent generates.

To facilitate executing a temporal logic specification, in [18] a normal form for temporal logic called Separated Normal Form (SNF) is introduced. SNF is a concise clausal form for temporal formulas which facilitates operationalization (see, e.g., METATEM [19, 20]). An SNF formula consists of an implication with a conjunction of propositional literals as the antecedent and a temporal formula as the consequent. The temporal formula consists of a temporal operator $\bigcirc$ (next) or $\diamond$ (sometime/eventually) applied to a disjunction of propositional literals. Any LTL formula can be transformed into an equivalent set of Separated Normal Form (SNF) rules [18]. By way of example, the LTL formula

$$\Box(\phi \rightarrow (\mathit{done}(a_1) \text{ before } \mathit{done}(a_2)))$$

may represent a norm for a robot in a rescue scenario expressing that it should always ($\Box$) be the case if the robot enters a room (ϕ) it should send a message to the other agents ($\mathit{done}(a_1)$) before it leaves the room ($\mathit{done}(a_2)$). This is represented in SNF as follows:

$$\left.\begin{array}{l} \phi \Rightarrow \neg \mathit{done}(a_2) \\ \phi \Rightarrow (w \vee \mathit{done}(a_1)) \end{array}\right\} \text{ present time rules}$$

$$\left.\begin{array}{l} w \Rightarrow \bigcirc(\neg \mathit{done}(a_2)) \\ w \Rightarrow \bigcirc(w \vee \mathit{done}(a_1)) \end{array}\right\} \text{ step rules}$$

The present time rules informally specify that if the left-hand side (lhs) of the rule holds in a state, then the right-hand side (rhs) must also hold in that state. Step rules represent that if the lhs holds, the rhs should hold in the next ($\bigcirc$) state. For reasons of simplicity, in this paper we consider the "next-fragment" of LTL as represented by present time and step rules. This means that we do not include rules that express that something should hold sometime in the future. The advantage of present time and step rules is that they can be evaluated over individual transitions. This does mean that the formalism discussed here cannot represent all norms encoded in LTL, but we believe it includes a large useful subset.

In the translation process from LTL to SNF, an auxiliary atom w was introduced above which can be read as 'waiting'. A new atom is introduced whenever a complex temporal operator (e.g., before) is translated into SNF. These atoms do not refer to the environment state as atoms in At, but are introduced for technical reasons in order to facilitate model building for the temporal formula. Their role is to defer, from state to state, the satisfaction of some formula e.g., $\mathit{done}(a_1)$ above – so in every state either $\mathit{done}(a_1)$ is true or the system is still waiting for a_1 to be done (w). While w is true $\mathit{done}(a_2)$ is prohibited because it must take place after $\mathit{done}(a_1)$. Agent actions do not directly affect auxiliary atoms, and therefore we use a separate set At_{aux} to denote the set of auxiliary atoms, where $\mathsf{At} \cap \mathsf{At}_{aux} = \emptyset$.

We use $\mathit{lit}(\mathsf{At})$ and $\mathit{lit}(\mathsf{Act})$ to denote the set of positive and negative literals over At and Act, respectively. We use *propositional literals* to refer to literals containing atoms from At, *auxiliary atoms* to refer to atoms from At_{aux}, and *action literals* to refer to literals from Act. To clearly distinguish action literals from propositional literals, we denote the former as $\mathit{done}(a)$ or $\neg\mathit{done}(a)$. We define present time and step rules as follows.

DEFINITION 1. *(Present Time Rules) A present time rule has the form* $\bigwedge l \Rightarrow \bigvee l'$ *where* $l \in \mathit{lit}(\mathsf{At})$ *and* $l' \in \mathit{lit}(\mathsf{Act}) \cup \mathsf{At}_{aux}$.

DEFINITION 2. *(Step Rules) A step rule has the form* $\bigwedge l \Rightarrow \bigcirc(\bigvee l')$ *where* $l \in \mathit{lit}(\mathsf{At} \cup \mathsf{Act}) \cup \mathsf{At}_{aux}$ *and* $l' \in \mathit{lit}(\mathsf{Act}) \cup \mathsf{At}_{aux}$.

Step rules correspond to conditional norms that oblige or forbid the execution of actions. For example, the step rule $p \Rightarrow \bigcirc(\mathit{done}(a))$ ($\bigcirc\neg\mathit{done}(a)$) can be read as a norm that expresses that if p holds, the agent is obliged (forbidden) to execute a next. Our language of norms is, for example, comparable to the normative language of the norm-aware agent framework of [29]. Our language is more restrictive than [29] in that we only allow actions and no states in the consequent of norms. It is more expressive in that we allow disjunctions of actions (and auxiliary atoms) in the consequent, and we have present time rules. LTL has also been used to specify norms for the MOISE organizational modelling language [33]. Our normative language differs from temporal deontic logics (e.g., [14, 6]) in that norms are not expressed explicitly. Temporal deontic logics are aimed at reasoning *about* which norms are in force, while our normative language directly expresses the behavior that agents should exhibit.

4. NORMATIVE AGENT SEMANTICS

In this section we define the normative agent semantics that formally defines how we integrate norms and agent deliberation. The semantics is a declarative variant of the semantics of [32], with a number of changes needed for providing a definition and characterization of strong norm compliance. We summarize these at the end of the section. The semantics is defined with respect to a set of norms N. If the set N changes at run-time because new norms are adopted, the agent can immediately adapt its behavior according to the semantics. We do not define such an update mechanism because it is not required for studying strong norm compliance.

[1]This means we can exclude multiply defined norms from consideration – e.g., $\neg smoking$ and $refrain\_from\_smoking$.

We use N to denote a set of norms in SNF and use $N_{PTR} \subseteq N$ and $N_{SR} \subseteq N$ to denote the sets of present time and step rules of N ($= N_{PTR} \cup N_{SR}$), respectively. In order to represent which auxiliary atoms hold and to represent the action that has been executed[2], we extend the states $s \in S$ to *normative states* $n = \langle s, aux, a\rangle$ with $aux \subseteq \mathsf{At}_{aux}$ and $a \in \mathsf{Act}$. We use $\lfloor$ to denote the projection of a normative state transition to a transition in *Dec*, i.e., $(\langle s, aux, a\rangle \longrightarrow \langle s', aux', a'\rangle)\lfloor \; = s \xrightarrow{a'} s'$.

We define that a normative state $n = \langle s, aux, a\rangle$ satisfies a formula $\phi = \bigwedge l$, denoted as $n \models \phi$, iff *(i)* all propositional literals (both positive and negative) from ϕ follow from s (according to entailment relation $\models_{\mathcal{L}_{\mathsf{At}}}$ introduced above), *(ii)* all auxiliary atoms from ϕ are contained in *aux*, and *(iii)* all positive action literals from ϕ are equal to a and all negative action literals are unequal to a. Satisfaction for a formula $\phi = \bigvee l$ is defined similarly by replacing "for all" by "there exists".

One advantage of the transformation of temporal formulas into present time and step rules is that satisfaction of these rules can be evaluated on single normative transitions $n \longrightarrow n'$. A transition satisfies a present time rule if it is satisfied in n and n', and a step rule $\phi \Rightarrow \bigcirc(\psi)$ is satisfied if it is the case that when ϕ holds in n, ψ holds in n'. We formally define satisfaction of a present time rule ($\models_{PTR}$) in a normative state as follows:

$$n \models_{PTR} (\phi \Rightarrow \psi) \quad \text{iff} \quad n \models \phi \Rightarrow n \models \psi$$

. We define satisfaction of present time ($\models_{PTR}$) and step rules ($\models_{SR}$) over a normative transition as follows:

$$n \longrightarrow n' \models_{PTR} (\phi \Rightarrow \psi) \quad \text{iff} \quad n, n' \models (\phi \Rightarrow \psi)$$
$$n \longrightarrow n' \models_{SR} (\phi \Rightarrow \bigcirc\psi) \quad \text{iff} \quad n \models \phi \Rightarrow n' \models \psi$$

We write $n \longrightarrow n' \models N$ to denote that a normative transition $n \longrightarrow n'$ satisfies all norms in N.

In defining a normative agent semantics one can make a range of choices that result in different semantics as well as computational properties. To guide and motivate the choices we make in this paper, we introduce a number of properties for such semantics. These are not the only desirable properties one may consider for such a semantics, and they do not uniquely characterize the semantics. However, by making these properties and the semantic choices explicit, we obtain a better understanding of the space of possibilities and its impact on norm compliance. The semantics in [32] was proven to satisfy the first two of the following properties.

- *Preservation of agent semantics:* If the set of norms $N = \emptyset$, then the generated normative traces correspond to those of the agent semantics.
- *Weak norm compliance:* Transitions derived by the transition rules should satisfy norms.
- *Preservation of norm compliance:* If a norm-compliant transition is possible according to $\mathcal{T}$, there should be a norm-compliant transition in the semantics[3]. Intuitively the semantics does not prevent the agent from executing a transition that is norm-compliant. This property is required for our characterization of strong norm compliance (Section 5.2).

In this paper we aim for a computational semantics that is "local", i.e., that takes into account only the current normative state in deriving a transition. The advantage of such a semantics over one that performs lookahead (and backtracking) to derive transitions is that it is computationally less intensive (see [25] for an example of a lookahead semantics to satisfy maintenance goals). A consequence is, however, that normative conflicts cannot always be prevented, which impacts norm compliance as we will see.

4.1 Semantics

Our normative agent semantics has two transition rules: one that allows us to derive transitions from *Dec* that are norm-compliant, and where no such transition exists one that allows the agent to take any norm-compliant action. The norm-constrained decision rule is introduced to satisfy preservation of the agent semantics. Its guard ensures that a transition can only be derived if it complies with the norms. In [32] the guard was used to construct the next state, while here we provide a simpler declarative constraint.

DEFINITION 3. *(Norm-Constrained Decision Rule)*

$$\frac{(n \longrightarrow n')\lfloor \in \mathit{Dec} \quad n \longrightarrow n' \models N}{n \longrightarrow n'}$$

This rule does not suffice to satisfy the preservation of norm compliance. Consider a step rule $\phi \Rightarrow \bigcirc \mathit{done}(b)$ and that in state $n = \langle s, aux, a\rangle$ we have $n \models \phi$, $\mathcal{T}(b, s) = s'$ and no other norms are applicable. In this case there is a norm-compliant transition to $n' = \langle s', \emptyset, b\rangle$, but unless $s \xrightarrow{b} s' \in \mathit{Dec}$, the transition cannot be derived. Thus we need a transition rule that allows *insertion* of actions into the agent semantics.

We also need to ensure that an agent always has an executable action that it can insert and that is not forbidden by norms. To this end we introduce an action $\texttt{wait} \notin \mathsf{Act}$ and extend $\mathcal{T}$ to include this action such that for all $s \in S : \mathcal{T}(\texttt{wait}, s) = s$. Thus `wait` does not change the agent state but, importantly, it will allow auxiliary atoms to change. Consider, for example, step rules $w_0 \Rightarrow \bigcirc(\neg \mathit{done}(a)), \phi \Rightarrow \bigcirc(w_1)$ and $w_1 \Rightarrow \bigcirc(\mathit{done}(a))$, and an agent in state $n \models \phi \wedge w_0$ – so the norms state that in this state the agent should not do a next but should do it the time step after that. Also assume that a is the only action. In this case the agent would not be able to make a transition through actions from Act, but it can make a norm compliant transition to $n' \models w_1$ through `wait` (which will remove w_0 from the auxiliary atoms and add w_1 – thus waiting until it is allowed to perform a). In n' it may then execute a if the action is enabled, in order to satisfy the third step rule. Without `wait`, no transitions would be possible. As we will see below, this would mean that the agent is not strongly compliant. We require that the transition rule for action insertion is only executed when the norm-constrained decision rule cannot be applied. This is needed for preservation of the agent semantics. Without this constraint the agent may derive transitions that do not appear in *Dec* when $N = \emptyset$.

DEFINITION 4. *(Action Insertion Rule) Let* $n = \langle s, aux, a\rangle$, $\mathcal{T}(a', s) = s'$, *and* n' *be of the form* $\langle s', aux', a'\rangle$[4].

$$\frac{\begin{array}{c} n \longrightarrow n' \models N \\ \forall n''.\, ((n \longrightarrow n'')\lfloor \in \mathit{Dec} \Rightarrow n \longrightarrow n'' \not\models N) \end{array}}{n \longrightarrow n'}$$

[32] proposes a stronger variant of the action insertion rule which only allows the insertion of actions that appear positively in the rhs of step rules. The idea is that the agent will only insert actions that

[2] This is needed to evaluate applicability of present time and step rules, since they may contain conditions about which action has been executed.

[3] This includes vacuous norms, in fact such norms may be required, see our discussion of `wait`.

[4] Note that is is allowable for n to equal n' if such a transition otherwise complies with the norms.

it may, at some point, be required to take. Such a rule provides a semantics that satisfies preservation of the agent semantics without requiring that it can be applied only if the norm-constrained decision rule is inapplicable. However, the stronger rule makes it impossible to make norm-compliant transitions in situations where any transition in *Dec* is forbidden but no action is required according to step rules, as in the example above for $\phi \Rightarrow w_1$. This semantics would therefore violate preservation of norm compliance.

The transition relation $\longrightarrow$ on normative states n is the smallest relation induced by the two transition rules defined above.

We define an agent $A = \langle S_0, \mathit{Dec}(\mathsf{Act}, S, \mathcal{T})\rangle$ where $S_0 \subseteq S$ is a non-empty set of initial states and *Dec* is an abstract agent decision mechanism. We define the normative agent semantics for an agent as the set of all normative traces that can be derived from the initial states using the normative transition system. To this end we transform initial agent states into initial normative states.

DEFINITION 5. *(Initial Normative States) Let s be a state and N a set of norms. An initial normative state of s, denoted as $\mathit{init}(s)$, is a tuple $\langle s, aux, \epsilon\rangle$ where aux is a set of auxiliary atoms for which $\langle s, aux, \epsilon\rangle \models N_{PTR}$, and $\epsilon \notin \mathsf{Act}$ is a special "empty" action.*

There are multiple initial states, for example representing different ways in which present time rules with more than one auxiliary atom in their rhs can be satisfied. An initial normative state does not always exist for any initial agent state – it may not be possible to create one that satisfies present time rules. The following proposition captures when an initial normative state exists.

PROPOSITION 1. *We use lit^- to denote the set of negative literals. Let N be a set of norms where for each $r \in N_{PTR}$ it holds that r has a disjunct $l' \in lit^-(\mathsf{Act}) \cup \mathsf{At}_{aux}$ in its right hand side. Then for any state $s \in S$ there is an initial normative state $\mathit{init}(s)$.*

Proof: Consider a present time rule $\phi \Rightarrow \psi$. If $s \not\models \phi$, the result trivially holds. If $s \models \phi$, either there is a negative action literal in ψ which holds in all $\mathit{init}(s)$ or there is an auxiliary atom w in ψ in which case there is an $\mathit{init}(s) = \langle s, aux, \epsilon\rangle$ where $w \in aux$. □

A trace in the normative agent semantics is a sequence of normative states where each consecutive state can be derived from the previous state through the transition rules defined above. If no transition can be derived then a special action `stop` is inserted that changes neither the agent state nor the auxiliary atoms. `stop` is thus different from `wait`, which allows changes to auxiliary atoms.

DEFINITION 6. *(Normative Agent Semantics)*
Let N be a set of norms and $A = \langle S_0, \mathit{Dec}(\mathsf{Act}, S, \mathcal{T})\rangle$ be an agent. A trace from state s in the normative transition system, typically denoted by t and referred to as a normative trace*, is an infinite sequence of normative states $n_0, n_1, n_2, \ldots$ such that n_0 is some $\mathit{init}(s)$, and for each $i \geq 0$ the transition $n_i \longrightarrow n_{i+1}$ is in the transition relation on normative states (Definitions 3 and 4), or no transition from $n_i = \langle s_i, aux_i, a_i\rangle$ can be derived and $n_{i+1} = \langle s_i, aux_i, \mathtt{stop}\rangle$ where $\mathtt{stop} \notin \mathsf{Act}$.*
We define $S^N(s)$ as the set of all normative traces under set of norms N starting in an initial state $\mathit{init}(s)$. This set is empty if there is no initial normative state for s. The normative agent semantics $[\![A]\!]_N$ of an agent A under set of norms N is the set of associated normative traces starting in an initial state of A, i.e.,

$$[\![A]\!]_N = \bigcup_{s \in S_0} S^N(s)$$

COROLLARY 1. *For each trace $t \in [\![A]\!]_N$, we have that the initial state of t satisfies N_{PTR}. Thus $[\![A]\!]_N = \emptyset$ iff all initial normative states violate present time rules N_{PTR}.*

Proof: Immediate from the definition of an initial normative state (Definition 5) and S^N (Definition 6). □

In contrast with [32], the normative agent semantics in this paper only yields infinite traces. The former yields finite traces if the agent cannot progress or would violate a norm. Besides yielding a more elegant semantics, it is required for investigating strong norm compliance which is defined over infinite traces.

4.2 Properties

We show that the normative agent semantics satisfies the three properties we introduced above. Preservation of the agent semantics is up to prefixes for finite traces of the agent. That is, if $N = \emptyset$ and for a state s there is no transition in *Dec*, then the normative semantics allows the agent to execute any action from Act[5].

PROPOSITION 2. *(Preservation of agent semantics) If A is an agent and $N = \emptyset$ then for each trace $t \in [\![A]\!]_N$, the maximal prefix of t for which each transition $(n \longrightarrow n) \restriction \in \mathit{Dec}$, is a trace of A.*

Proof: Immediate from the fact that Definition 4 can only be applied if Definition 3 does not. If N is empty then norms do not impose constraints on the derivation of transitions from *Dec*. □

We define that a trace weakly complies with a set of norms if all non-stop transitions comply with norms.

DEFINITION 7. *(Weak norm compliance) A normative trace t* weakly complies *with a set of norms N iff all non-stop transitions $n \longrightarrow n'$ (i.e., where n' is of the form $\langle s', aux', a'\rangle$ and $a' \neq \mathtt{stop}$) in t satisfy N. An agent A weakly complies with set of norms N iff for all $t \in [\![A]\!]_N$ we have that t weakly complies with N.*

Defining weak norm compliance by referring only to non-stop transitions is justified because stop occurs iff the transition does not comply with norms. The inclusion of `wait` ensures that this proposition holds because it prevents the semantics deriving a `stop` transition just because no executable actions are available.

PROPOSITION 3. *(Stop signals norm violation) Let A be an agent and N a set of norms. Then a transition $n \longrightarrow n'$ on a trace $t \in [\![A]\!]_N$ is a stop transition (i.e., of the form $\langle s, aux, a\rangle \longrightarrow \langle s, aux, \mathtt{stop}\rangle$) iff $n \longrightarrow n' \not\models N$.*

Proof: Follows because a stop transition can only be derived iff no norm compliant transition exists by Definition 6. □

PROPOSITION 4. *The normative agent semantics is weakly norm compliant. That is, for any agent A and set of norms N, it holds that A weakly complies with N.*

Proof: The transition rules of Definitions 3 and 4 only allow the derivation of transitions that comply with norms. Weak compliance only concerns those transitions. □

PROPOSITION 5. *(Preservation of norm compliance) Let A be an agent, N a set of norms and $n = \langle s, aux, a\rangle$ a state on a trace $t \in [\![A]\!]_N$. If there is a normative state $n' = \langle s', aux', a'\rangle$ such that $\mathcal{T}(a', s) = s'$ and $n \longrightarrow n' \models N$ then there is a normative state n'' such that $n \longrightarrow n''$ is a norm compliant transition in t.*

[5]The semantics could be adapted so that, in this situation, only `wait` transitions were be selected if no transition from *Dec* were available, but we do not do that here for reasons of simplicity.

Proof: If $n \longrightarrow n'$ can be derived by Definition 3 we are done. Otherwise if $\mathcal{T}(a', s) = s'$ it can be derived by Definition 4 which encompasses all other norm compliant transitions in $\mathcal{T}$. □

In summary, we have obtained the following insights regarding the definition of a semantic framework for socially adaptive agents targeted at strong norm compliance:

- Preservation of norm compliance requires a weaker form of action insertion than proposed in [32] because satisfying norms may require progression without explicitly requiring the execution of an action.
- A consequence of using a weaker form of action insertion is that preservation of the agent semantics can only be proven up to maximal prefixes. That is, if $N = \emptyset$ and according to *Dec* the agent would terminate at some point, according to the normative agent semantics the agent would continue executing actions if possible. In [32] it was shown that the stronger action insertion rule yields exact preservation of the agent semantics, i.e., not only up to maximal prefixes.
- Defining weak norm compliance over infinite traces requires distinguishing norm compliant transitions that do not change the agent state (e.g., `wait`) from those that violate norms (`stop`). Intuitively, this highlights that there is a difference between refraining from doing an action in order to avoid norm violation (if certain actions are forbidden) and aborting the execution of the agent. The former is ok from the perspective of strong norm compliance, while the latter is problematic.

5. STRONG NORM COMPLIANCE

Weak norm compliance expresses that *if* the agent can derive a transition through the transition rules of Definitions 3 or 4, the transition is norm compliant. We argue that this notion is too weak to model what one would intuitively consider to be a norm compliant agent. Consider a robot in a care facility that is required by the family to leave the patient's room immediately when the family enters, and it is required by the physician to hold a device immediately when s/he asks it to do so. Now assume the family comes in and the physician asks it to hold a device. The robot can stop and be weakly norm compliant. However, from the perspective of the family and the physician, if the robot stops it violates both norms.

We introduce a strong notion of norm compliance to reflect this intuition. We define that an agent strongly complies with norms when all transitions on all traces of the agent satisfy the norms, and there is at least one such trace. The set of traces is empty iff all initial states of the agent violate present time rules (Corollary 1).

DEFINITION 8. *(Strong Norm Compliance)*
A normative trace t strongly complies *with a set of norms N iff all transitions $n \longrightarrow n'$ in t satisfy N, i.e., $n \longrightarrow n' \models N$. An agent A strongly complies with a set of norms N iff there is a transition $t \in [\![A]\!]_N$ and for all $t \in [\![A]\!]_N$, t strongly complies with N.*

COROLLARY 2. *Strong compliance implies weak compliance, i.e., if an agent A strongly complies with a set of norms N then A weakly complies with N.*

In contrast with weak norm compliance, not all agents are strongly norm compliant with any set of norms. In our example above where N contained $\mathit{family} \Rightarrow \bigcirc(\mathit{done}(\mathit{leave}))$ and $\mathit{requestHold} \Rightarrow \bigcirc(\mathit{done}(\mathit{hold}))$, when both *family* and *requestHold* are satisfied, no norm compliant transition can be derived and the semantics introduces the `stop` transition. This transition does not comply with these step rules, and thus the robot is not strongly norm compliant.

Weak and strong norm compliance differ only with respect to the evaluation of stop transitions. Thus the traces of strongly norm compliant agents do not have stop transitions. This corresponds with the intuition that to be strongly norm compliant, agents should not be allowed to abort to prevent norm violation.

In this section we explore semantic characteristics of strong norm compliance from three perspectives. We provide a logical characterization of when a strongly norm compliant agent exists (Section 5.1), we provide a semantic characterization of strong norm compliance (Section 5.2), and based on this show that certain subclasses of norms ensure strong norm compliance (Section 5.3).

5.1 LTL Satisfiability

In this section we investigate how strong norm compliance relates to satisfiability of particular LTL expressions. Intuitively, strong norm compliance corresponds to satisfaction of norms expressed as LTL formulas. We investigate whether our semantics and notion of strong norm compliance indeed satisfy this property.

To this end, we define how normative states in our semantics can be translated to LTL states, and based on this how normative traces can be translated to LTL traces. LTL states consist of a valuation over the propositions in the language over which formulas are defined. In our case this means that for each atom, auxiliary atom, and action, either the atom or its negation is part of the state. Normative states can then be constructed from LTL states straightforwardly.

DEFINITION 9. *(Normative trace to LTL trace and vice versa) The function $\mathit{ns2ltl}(n)$ for some normative state $n = \langle s, \mathit{aux}, a\rangle$ yields an LTL state consisting of the set of literals l where $l \in \mathit{lit}(\mathsf{At})$ and $s \models l$, auxiliary atoms w for $w \in \mathit{aux}$, and negated auxiliary atoms $\neg w$ for $w \in \mathsf{At}_{\mathit{aux}} \setminus \mathit{aux}$, positive action literal $\mathit{done}(a)$ as well as $\neg\mathit{done}(a')$ for any $a' \in \mathsf{Act}$ with $a' \neq a$. The function $\mathit{nt2ltl}(t)$ transforms a normative trace t that does not include `stop` to a normative LTL trace by applying ns2ltl to each state on the trace. The space of normative LTL traces consists of sequences of states defined over At, $\mathsf{At}_{\mathit{aux}}$, and actions $\mathsf{Act} \cup \{\texttt{wait}\}$. Similarly, the functions $\mathit{ltls2n}(n)$ and $\mathit{ltlt2n}(t)$ transform an LTL state and trace to a normative state and trace.*

Because states are complete, for any $\mathit{ns2ltl}(n)$ and atom $p \in \mathsf{At}$ we have that either $p \in \mathit{ns2ltl}(n)$ or $\neg p \in \mathit{ns2ltl}(n)$. This is required for construction of an action transition function $\mathcal{T}$ from a trace.

One may expect that strong norm compliance corresponds to satisfiability of the set of norms N. However, it is not enough to consider only the set of norms. Our semantics adds a number of constraints on traces: no action has been executed in initial states (*init*); precisely one action at a time can be executed, which means that only one atom of the form $\mathit{done}(a)$ can hold in a state (*Single*); and at least one must hold (*Do*). Finally, the action `wait` does not change the state expressed over atoms (*Wait*). Below we describe these properties as LTL formulas.

1. $\mathit{init} \equiv \big(\bigwedge_{a \in \mathsf{Act}} \neg\mathit{done}(a)\big) \wedge \neg\mathit{done}(\texttt{wait})$
2. $\mathit{Single} \equiv \Box\big(\bigwedge_{a_1, a_2 \in \mathsf{Act}} \neg\big(\mathit{done}(a_1) \wedge \mathit{done}(a_2)\big)\big)$
3. $\mathit{Do} \equiv \Box\big(\neg\mathit{init} \rightarrow \bigvee_{a \in \mathsf{Act} \cup \{\texttt{wait}\}} \mathit{done}(a)\big)$
4. $\mathit{Wait} \equiv \Box\big(\bigwedge_{p \in \mathsf{At}} (p \wedge \bigcirc(\mathit{done}(\texttt{wait})) \rightarrow \bigcirc(p)\big)$
5. $N_{\mathit{spec}} \equiv \bigwedge_{r \in N} \Box(r)$

We prove that a formula representing the norms as well as the constraints on initial states and execution of actions is satisfiable iff there is an agent that strongly complies with these norms.

THEOREM 1. *(Satisfiable norms) Let N be a set of norms defined over a set of atoms* At*, set of auxiliary atoms* At_{aux}*, and actions* Act*. We then have that init $\wedge$ Single $\wedge$ Do $\wedge$ Wait $\wedge$ N_{spec} is satisfiable iff there is an action transition function $\mathcal{T}$ and set of initial states S_0 such that an agent $A = \langle S_0, Dec(\mathsf{Act}, S, \mathcal{T})\rangle$ strongly complies with N.*

Proof: (Sketch) ($\Rightarrow$) Assume *init $\wedge$ Single $\wedge$ Do $\wedge$ Wait $\wedge$ N_{spec}* is satisfiable. We can construct the appropriate agent from a trace that satisfies this formula using *ltls2n* to define the transitions in $\mathcal{T}$.

($\Leftarrow$) Because some agent, A, strongly complies with norms we know that there is a strongly norm compliant trace $t \in [\![A]\!]_N$ and thus a strongly compliant normative LTL trace *nt2ltl*(t). This trace satisfies *Single $\wedge$ Do $\wedge$ Wait $\wedge$ N_{spec}*. □

5.2 Characterizing Strong Norm Compliance

In this section we provide a characterization of strong norm compliance by showing that an agent strongly complies with norms iff there are no states n on any trace of the agent that are in conflict with respect to the agent's norms and transition function. This yields a deeper understanding of the notion of strong norm compliance by identifying precisely when a problem occurs, and it provides the basis for identifying subsets of norms for which all agents are strongly norm compliant.

We say that a normative state is in conflict when all possible ways in which the agent could potentially progress to a next state are in conflict. To capture what it means to "potentially progress to a next state", we introduce the notion of a Potential Normative Future (PNF). The idea of a PNF is that it captures two states that *could* be paired by the transition system in a way that satisfies all those parts of the norms that do not explicitly restrict the actions that may be taken. A PNF is a tuple $\langle s', aux', A\rangle$ where s' and aux' represent the agent state and auxiliary atoms of the potential next state, and A is a set of action literals that represents the constraints that norms place on the actions that may be executed for this potential next state. The definition of a PNF is independent of $\mathcal{T}$. It implicitly assumes that any state can transition to any other.

DEFINITION 10. *(Potential Normative Future) Let N be a set of norms and $n = \langle s, aux, a\rangle$ a normative state. A potential normative future (PNF) of n is a tuple $\langle s', aux', A\rangle$ where $s' \in S$ is a state, $aux' \subseteq \mathsf{At}_{aux}$ is a set of auxiliary atoms and $A \subseteq lit(\mathsf{Act})$ is a set of action literals such that:*

1. *$\forall r = \phi \Rightarrow \psi \in N_{PTR}$: if $s' \models \phi$ then $\exists l \in \psi : l \in aux'$ or $\exists l \in \psi : l \in A$, and*
2. *$\forall r = \phi \Rightarrow \bigcirc\psi \in N_{SR}$: if $n \models \phi$ then $\exists l \in \psi : l \in aux'$ or $\exists l \in \psi : l \in A$*

Note that A may contain additional action literals not required by the norms. This is required for the proof of Lemma 1.

As an example consider the states s_1, s_2 such that $s_1 \models \phi$ and the norms we generated for $\Box(\phi \rightarrow (done(a_1)\ \mathsf{before}\ done(a_2)))$ in section 3.2. Then the potential normative futures for some normative state $\langle s_1, \{w\}, a\rangle$ include $\langle s_1, \{w\}, \{\neg done(a_2)\}\rangle$, $\langle s_1, \emptyset, \{done(a_1), \neg done(a_2)\}\rangle$ and $\langle s_2, \{w\}, \{\neg done(a_2)\}\rangle$. They also include "impossible" futures such as $\langle s_1, \{w\}, \{done(a_2), \neg done(a_2)\}\rangle$ – the only requirement is that all literals *required* by the norms be present.

We now define a notion of a conflicted normative state. We identify three possible causes of conflict: (i) the norms require the agent to perform an action that it cannot perform (*practical conflict*), (ii) the norms require the agent to do an action that is forbidden by another (*prohibition conflict*), or (iii) the norms require the agent to do multiple actions at the same time (*obligation conflict*). The latter two are together called a *normative conflict*. A PNF is in conflict if it is in practical or normative conflict.

DEFINITION 11. *(Normative State in Conflict) Let N be a set of norms, $\mathcal{T}$ be a transition function, and $n = \langle s, aux, a\rangle$ a normative state, and let pnf $= \langle s', aux', A\rangle$ be a PNF of n wrt. N.*

- *We say that pnf is in* practical conflict *with $\mathcal{T}$ iff there is an action a such that $done(a) \in A$ and there is no $done(a) \in A : \mathcal{T}(a, s) = s'$.*
- *We say that pnf is in* normative conflict *iff $\exists a, b : a \neq b$ and $done(a), done(b) \in A$ (obligation conflict), or $\exists a : done(a), \neg done(a) \in A$ (prohibition conflict).*

We say that a PNF is in conflict with $\mathcal{T}$ if it is in practical conflict with $\mathcal{T}$ or in normative conflict. We say that a normative state n is in conflict *wrt. set of norms N and action transition function $\mathcal{T}$ iff all PNFs of n with respect to N which contain positive action literals are in conflict with $\mathcal{T}$.*

Where $\mathcal{T}$ and N are obvious from the context we will say that, respectively, a PNF is in conflict or a normative state is in conflict.

COROLLARY 3. *Let pnf $= \langle s', aux', A\rangle$ be a PNF of n with respect to N. Then the following holds: if there is no action a such that $done(a) \in A$, then pnf is not in conflict with any $\mathcal{T}$.*

To prove the theorem that an agent strongly complies with norms iff none if its traces contain conflicting normative states, we first prove a lemma that shows that a transition $n \longrightarrow n'$ complies with norms iff there is a non-conflicting PNF of n.

LEMMA 1. *(Norm-compliance and non-conflicting PNFs) Let N be a set of norms, n a norm compliant state, and $n \longrightarrow n'$ with $n' = \langle s', aux', a'\rangle$ be a transition for an agent A with transition function $\mathcal{T}$. Then $n \longrightarrow n' \models N$ iff there is a PNF $\langle s', aux', A\rangle$ of n that contains a positive action literal and that is not in conflict.*

Proof: (Sketch) ($\Rightarrow$) We construct a non-conflicting PNF, $\langle s', aux', A \cup \{done(a')\}\rangle$, by examining the RHS of all applicable normative rules. If the RHS contains an auxiliary atom it is placed in aux' otherwise it is placed in A. It is simple to prove that this PNF is not in conflict.

($\Leftarrow$) We assume there is a potential normative future *pnf* $= \langle s', aux', A\rangle$ for $n = \langle s, aux, a\rangle$, which is not in conflict and which contains a positive action literal, $done(a')$. Since $done(a') \in A$ then $\mathcal{T}(a', s) = s'$ We show that $n \longrightarrow \langle s', aux', a'\rangle \models N$ which follows simply from the fact that *pnf* is not in conflict. □

We prove that an agent strongly complies with norms iff there are no conflicting states.

THEOREM 2. *An agent $A = \langle S_0, Dec(\mathsf{Act}, S, \mathcal{T})\rangle$ strongly complies with a set of norms N iff there is no normative state n on some trace $t \in [\![A]\!]_N$ that is in conflict.*

Proof: ($\Rightarrow$) Assume that an agent A strongly complies with a set of norms N. This means that all transitions $n \longrightarrow n'$ on all traces $t \in [\![A]\!]_N$ satisfy N. Then by Lemma 1 we know that n has a PNF that contains a positive action literal and is not in conflict, which means that n is not in conflict.

($\Leftarrow$) Assume that there is no normative state n on some trace $t \in [\![A]\!]_N$ that is in conflict. Then for all states n there is a PNF $\langle s', aux', A\rangle$ of n that contains a positive action literal and is not in conflict. Then by Lemma 1 we know that there is an $n' = \langle s', aux', a'\rangle$ such that $n \longrightarrow n' \models N$. By Proposition 5

we know that there is a transition, $n \longrightarrow n''$ on a trace $t \in [\![A]\!]_N$ and $n \longrightarrow n'' \models N$. Therefore there is a norm-compliant transition from all states n on all traces t, and if there is a norm-compliant transition from some state n then all transitions from n on all traces are norm compliant (since `stop` means no norm-compliant transition was possible). Hence A strongly complies with N. □

5.3 Safe Norms

In this section we analyze a number of subsets of norms for which any agent A is strongly norm compliant. This means by Theorem 2 that there can be no normative state of any A that is in conflict. We call these "safe" norms. The first proposition considers step rules with only negative action literals in the rhs.

PROPOSITION 6. *Let N be a set consisting only of step rules of the form $\phi \Rightarrow \bigcirc\neg done(a)$. Then any agent strongly complies with N.*

Proof: Immediate from Corollary 3. □

Proposition 7 concerns step rules that oblige the execution of an action in the next state. Such norms cannot give rise to a conflict when the lhs of the norm implies the precondition of the action (ought implies can), which means that whenever the norm is applicable, the action can be executed. To prevent two actions being obliged at the same time, the lhs of these rules are defined to be mutually exclusive. This is formalized in the following proposition.

PROPOSITION 7.
Let $\mathcal{T}$ be an action transition function and let N be a set consisting only of step rules of the form $\phi \Rightarrow \bigcirc done(a)$ such that:

- *for all $\phi \Rightarrow \bigcirc done(a) \in N$: if $s \models \phi$, then $\mathcal{T}(a, s)$ is defined (ought implies can), and*
- *for any two norms $\phi \Rightarrow \bigcirc done(a) \in N$ and $\phi' \Rightarrow \bigcirc done(a') \in N$ we have $\models \phi \Rightarrow \neg\phi'$ (mutually exclusive lhs).*

Then we have that any agent A with action transition function $\mathcal{T}$ strongly complies with N.

Proof: We have to show that none of the states on traces of A are in conflict. This is simple to show from the fact that at most one rule is applicable in any state. □

We note that the implication does not hold in the other direction. If an agent strongly complies with norms consisting of step rules of the form $\phi \Rightarrow \bigcirc done(a)$, this does not mean that these have to follow ought implies can nor that the lhs have to be mutually exclusive. For example, if the agent can avoid states where the lhs of these rules hold, then they are never applicable. If the lhs of these rules can be satisfied in initial states, which means they should not include positive action literals, and the initial states of the agent $A = \langle S_0, \langle \mathsf{Act}, S, \mathcal{T}\rangle\rangle$ are complete (i.e., $S_0 = S$), then the proposition holds in both directions.

The next proposition considers norms of the form introduced in Section 3.2 where an action has to be executed before another one.

PROPOSITION 8. *Let N be the SNF rules corresponding to a set of norms of the form $\phi \Rightarrow (done(a_1)$ before $done(a_2))$. Then any agent A strongly complies with N.*

Proof: The SNF rules for norms of this form were presented in Section 3.2. Consider a state n where both ϕ and w hold for all rules of this form, i.e., all norms are applicable. If there is no such state, similar reasoning applies for subsets of N. The question is now whether a conflict occurs in such a state. These norms can be satisfied (i.e., a PNF can be constructed accordingly) by not executing actions a_2 and satisfying w, because even if all actions from Act are included in these norms, the agent can execute `wait` to get to the next state in a norm compliant way. Thus a non-conflicting PNF $\langle s', aux', A\rangle$ of n is one where the action set A consists of negative action literals for all actions a_2, aux' is the set of auxiliary atoms w occurring in these rules, and s' is the result of executing some executable action from $\mathsf{Act} \cup \{\texttt{wait}\}$ such that no norms are violated (and we know this exists due to `wait`). Therefore n is not in conflict, and we are done. □

We note that while any agent with these norms is strongly norm compliant, it may be so because it continues to execute `wait` and thus postpone the execution of $done(a_2)$. If we require that a_2 *is* executed eventually, then assumptions on $\mathcal{T}$ have to be introduced to ensure that a_2 can be executed. These "sometime" rules that require that an action is eventually executed take us out of the next fragment of LTL, and so are not considered in this paper.

6. CONCLUSION AND FUTURE WORK

If we want to ensure that an agent can follow norms in a range of different scenarios where norms may change at run-time, we cannot pre-program all such possible normative behaviours at design time. For this reason we propose a formal semantic framework that specifies an execution mechanism for socially adaptive agents that can adapt their behavior to comply with newly adopted norms at run-time.

Whether an agent can comply with norms is determined by the interplay between execution semantics, norms, and agent capabilities.[6] To arrive at a fundamental understanding of how the combination of these elements affects norm compliance we have provided a formal specification for them, and in doing so have made choices in defining the normative agent semantics, the language of norms and our notion of compliance. We have argued why these choices were made, and we have shown how these choices resulted in satisfaction of relevant properties of the resulting framework.

In summary, this paper provides the following main contributions: i) a set of properties that a normative agent semantics should satisfy, ii) a declarative *normative agent semantics* with proofs that the properties are satisfied, iii) a definition of *strong norm compliance* with an exploration of its semantic characteristics. In particular we provide a logical characterization, a semantic characterization, and an exploration of safe norm sets.

In future work we plan to explore other choices for defining a framework and how they affect norm compliance. In particular, extensions of the language of norms, such as larger fragments of SNF, and alternative notions of norm compliance, such as the *bounded* norm compliance corresponding to bounded rationality. We also aim to investigate the effect of complying with norms on goal achievement (see also [21]), and studying how an agent can comply with norms with minimal effort. Finally, we will explore how this framework can be integrated with techniques for reasoning about norm violation, working towards a comprehensive framework for normative agents.

REFERENCES

[1] N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In *Proceedings of the 11th International*

[6] Eventually we will also need to take into account environment characteristics and actions of other agents and study their effect on norm compliance. However, before we can embark upon studying such complex interactions we need a proper understanding of the fundamentals of norm compliance.

Conference on Autonomous Agents and Multiagent Systems (AAMAS'12), pages 1057–1064. IFAAMAS, 2012.

[2] G. Andrighetto, M. Campenni, F. Cecconi, and R. Conte. The complex loop of norm emergence: A simulation model. In *Simulating Interacting Agents and Social Phenomena: The second world congress*, volume 7 of *Agent-Based Social Systems*, pages 19–35. Springer, 2010.

[3] G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[4] G. Boella and R. Damiano. An architecture for normative reactive agents. In *Proceedings of the 5th Pacific Rim International Workshop on Intelligent Agents and Multi-Agent Systems (PRIMA'02)*, volume 2413 of *LNCS*, pages 1–17, 2002.

[5] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[6] J. Broersen and J. Brunel. Preservation of obligations in a temporal and deontic framework. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, pages 1108–1110. ACM, 2007.

[7] J. Broersen, S. Cranefield, Y. Elrakaiby, D. Gabbay, D. Grossi, E. Lorini, X. Parent, L. W. N. van der Torre, L. Tummolini, P. Turrini, and F. Schwarzentruber. Normative reasoning and consequence. In G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre, editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 33–70. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[8] N. Bulling, M. Dastani, and M. Knobbout. Monitoring norm violations in multi-agent systems. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 491–498. IFAAMAS, 2013.

[9] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm acceptance. In *Proceedings of the 5th International Workshop on Agent Theories, Architectures, and Languages (ATAL'98)*, volume 1555 of *LNCS*, pages 99–112. Springer, 1999.

[10] S. Cranefield, M. Winikoff, and W. Vasconcelos. Modelling and monitoring interdependent expectations. In S. Cranefield, M. Riemsdijk, J. Vázquez-Salceda, and P. Noriega, editors, *Coordination, Organizations, Institutions, and Norms in Agent System VII*, volume 7254 of *LNCS*, pages 149–166. Springer, 2012.

[11] N. Criado, E. Argente, P. Noriega, and V. Botti. Human-inspired model for norm compliance decision making. *Information Sciences*, 245:218–239, 2013.

[12] D. D'Aprile, L. Giordano, V. Gliozzi, A. Martelli, G. Pozzato, and D. T. Dupré. Verifying business process compliance by reasoning about actions. In J. Dix, J. Leite, G. Governatori, and W. Jamroga, editors, *Computational Logic in Multi-Agent Systems*, volume 6245 of *Lecture Notes in Computer Science*, pages 99–116. Springer, 2010.

[13] F. Dignum, V. Dignum, J. Thangarajah, L. Padgham, and M. Winikoff. Open agent systems??? In *Proceedings of the 8th International Workshop on Agent-Oriented Software Engineering (AOSE'07)*, volume 4951 of *LNCS*, pages 73–87. Springer, 2008.

[14] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 5, pages 336–346. IEEE, 1997.

[15] A. Elgammal. *Towards a comprehensive framework for business process compliance*. PhD thesis, 2012.

[16] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.

[17] M. S. Fagundes, H. Billhardt, and S. Ossowski. Normative reasoning with an adaptive self-interested agent model based on Markov Decision Processes. In *Proceedings of the 12th Ibero-American Conference on AI (IBERAMIA'10)*, volume 6433 of *LNCS*, pages 274–283. Springer, 2010.

[18] M. Fisher. A normal form for temporal logics and its applications in theorem-proving and execution. *Journal of Logic and Computation*, 7(4):429–456, 1997.

[19] M. Fisher. Agent deliberation in an executable temporal framework. *Journal of Applied Logic*, 9(4):223–238, 2011.

[20] M. Fisher and A. Hepple. Executing logical agent specifications. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Tools and Applications*, pages 1–27. Springer, Berlin, 2009.

[21] G. Governatori, F. Olivieri, S. Scannapieco, and M. Cristani. Designing for compliance: Norms and goals. In F. Olken, M. Palmirani, and D. Sottara, editors, *5th International Symposium on Rule-Based Modeling and Computing on the Semantic Web (RuleML'11)*, pages 282–297. Springer, 2011.

[22] G. Governatori and S. Shazia. The journey to business process compliance. pages 426–454. IGI Global, 2009.

[23] H. Groefsema and D. Bucur. A survey of formal business process verification: From soundness to variability. In *International Symposium on Business Modeling and Software Design*, pages 198–203, 2013.

[24] J. Hansen, G. Pigozzi, and L. van der Torre. Ten philosophical problems in deontic logic. In G. Boella, L. van der Torre, and H. Verhagen, editors, *Normative Multi-agent Systems*, number 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[25] K. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In *Proceedings of the fifth International Workshop on Declarative Agent Languages and Technologies (DALT'07)*, volume 4897 of *LNAI*, pages 86–103. Springer, 2007.

[26] F. López y López, M. Luck, and M. d'Inverno. Constraining autonomy through norms. In *The First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 674–681. ACM, 2002.

[27] F. López y López, M. Luck, and M. d'Inverno. Normative agent reasoning in dynamic societies. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 732–739. IEEE Computer Society, 2004.

[28] F. M. Maggi, M. Westergaard, M. Montali, and W. M. van der Aalst. Runtime verification of ltl-based declarative process models. In S. Khurshid and K. Sen, editors, *Runtime*

Verification, volume 7186 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2012.

[29] F. Meneguzzi and M. Luck. Norm-based behaviour modification in BDI agents. In *Proceedings of the eighth international joint conference on autonomous agents and multiagent systems (AAMAS'09)*, pages 177–184, Budapest, 2009.

[30] M. Montali, P. Torroni, F. Chesani, P. Mello, M. Alberti, and E. Lamma. Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundamenta Informaticae*, 102(3-4):325–361, 2010.

[31] S. Panagiotidi and J. Vázquez-Salceda. Norm-aware planning: Semantics and implementation. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, pages 33–36. IEEE, 2011.

[32] M. B. van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks. Agent reasoning for norm compliance: a semantic approach. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 499–506. IFAAMAS, 2013.

[33] M. B. van Riemsdijk, K. V. Hindriks, C. M. Jonker, and M. Sierhuis. Formalizing organizational constraints: A semantic approach. In *Proceedings of the nineth international joint conference on autonomous agents and multiagent systems (AAMAS'10)*, pages 823–830. IFAAMAS, 2010.