

Bridging Social and Technical Compliance in Autonomous Systems through a Standardized Pattern Catalog

Luca Brodo

luca.brodo@hshl.de

Hochschule Hamm-Lippstadt
Lippstadt, NRW, Germany

Lukas Walter

lukas.walter@hshl.de

Hochschule Hamm-Lippstadt
Lippstadt, NRW, Germany

Giuseppe Scalora

giuseppe.scalora@hshl.de

Hochschule Hamm-Lippstadt
Lippstadt, NRW, Germany

Stefan Henkler

stefan.henkler@hshl.de

Hochschule Hamm-Lippstadt
Lippstadt, NRW, Germany

Abstract

As autonomous systems move into human-centered environments, their correctness can no longer be defined in purely technical terms. Beyond safety and liveness, these systems must comply with *social norms*: the implicit, context-dependent rules that govern acceptable conduct, such as yielding to pedestrians or avoiding noise near residential areas. Temporal logics such as TCTL and CTL are well established for technical properties, but engineers have no systematic way to verify social norms alongside them: encoding such norms in (T)CTL by hand is error-prone, hard to trace back to requirements, and not reusable. We address this gap with a standardized catalog of reusable observer patterns spanning both families of social norm: *regulative* norms, which constrain behavior (permission, prohibition, obligation, exceptions, role-based authority, and graded compliance), and *constitutive* norms, which create institutional facts. Each pattern couples a natural-language template, a (T)CTL formalization, and an observer automaton for integration into existing workflows. We evaluate the catalog on smart-farming and autonomous-transport scenarios, showing how engineers formalize social requirements, resolve norm conflicts, and quantify compliance–performance trade-offs.

CCS Concepts

• **Software and its engineering** → **Extra-functional properties.**

Keywords

Social specifications, social compliance, formal verification

ACM Reference Format:

Luca Brodo, Giuseppe Scalora, Lukas Walter, and Stefan Henkler. 2026. Bridging Social and Technical Compliance in Autonomous Systems through a Standardized Pattern Catalog. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026)*, October 04–09, 2026, Málaga, Spain. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3837062.3838903>

1 Introduction

Formal verification is an established technique for ensuring correctness and reliability in software-intensive and safety-critical systems [7, 15]. In particular, model checking enables the automated validation of system behavior against formally specified requirements. The verification process typically follows three steps [12, 13]: (i) elicitation of natural-language requirements, (ii) their formalization into formal specifications, and (iii) automated verification of the system model. In the context of safety-critical cyber-physical systems (CPS), this pipeline is most commonly instantiated using timed automata models [5], where properties are expressed in untimed or timed computation tree logic (TCTL [4] and CTL [7, 15], respectively) and verification is carried out using UPPAAL [8].

This pipeline is well-established for *technical* properties such as collision avoidance (safety) and task completion (liveness), and has been applied across many safety-critical systems [29, 37, 38]. However, as autonomous systems move into human-centered environments, correctness can no longer be purely technical: beyond safety and liveness, they must also comply with *social norms*, the rules governing acceptable conduct in human environments. Ignoring such norms can erode public trust and hinder adoption [3, 18, 21, 42], and even incur regulatory and legal liability: the European AI Act, for one, demands trustworthy, human-centric AI whose societal impact exceeds narrow technical compliance [1]. Crucially, normativity is not monolithic. The literature distinguishes two fundamental kinds of norm [10, 43, 44]: *regulative* norms, which constrain antecedently possible behavior, and *constitutive* norms, which *create institutional facts*, statuses, and powers through rules of the form “*X counts as Y in context C*”. Both kinds are highly relevant to autonomous systems, which must not only avoid prohibited behaviors but also respect the institutional contexts in which they operate.

Yet verifying social norms remains an open challenge. Unlike technical properties, they are implicit, culturally situated, and vary across regions, and engineers have no established methodology for specifying and verifying them alongside technical properties. Existing work addresses fragments rather than the whole pipeline, is decoupled from the timed-automata infrastructure engineers actually use, and lacks tool support comparable to UPPAAL. And although social norms *can* be expressed in CTL and TCTL [3, 21, 42], doing so is error-prone [20, 28, 49], ad hoc, and unsupported, yielding formulas that are hard to trace to requirements and not reusable.



This work is licensed under a Creative Commons Attribution 4.0 International License. *MODELS Companion 2026, Málaga, Spain*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2903-4/2026/10
<https://doi.org/10.1145/3837062.3838903>

We close this gap with a standardized catalog of reusable observer patterns spanning both regulative and constitutive social norms. Each pattern offers a tripartite interface: a natural-language template guiding requirements elicitation, a precise (T)CTL formalization, and an observer automaton ready for established model checkers. By separation of concerns [40], each norm is a dedicated observer that monitors behavior without altering it, so social properties are co-verified alongside safety and liveness in one workflow. To our knowledge, this is the first catalog to cover both norm families, letting engineers go from a stakeholder requirement to a verified property without constructing temporal formulas from scratch, switching tools, or bridging formalisms.

This paper is structured as follows. It covers the motivating scenario (Section 2), related work (Section 3), the pattern catalog (Section 5) and its completeness (Section 6), the evaluation (Section 7), and limitations (Section 8).

2 Motivating Example

Consider a fleet of autonomous tractors (ATs) that carry goods between outlying fields and a coordinator-managed processing hub, sharing peri-urban roads with human field workers, residents, and ordinary traffic. Verifying such a deployment technically, with collision freedom and timely completion as key criteria, is necessary but no longer sufficient: to be socially accepted, each AT must also honor the tacit expectations of the people around it. Stakeholder interviews in projects such as [31] voice concrete anxieties about *disruptive* behavior, such as a driverless tractor holding up a line of commuters or grinding through a residential street at dinnertime. When a worker steps onto a track, a well-behaved AT *may* slow to let them pass; this is welcome, but never required (**R1**); it must *not* enter the residential zone after 5 pm (**R2**); and when vehicles queue behind it, it must pull over within a bounded time t (**R3**). Such norms are defeasible, not absolute: the quiet-hours rule is overridden when the AT carries perishable, time-critical harvest (**R4**). Context also reshapes *which* norms apply, inside the work zone around field workers the AT must integrate courtesy into stricter yielding and speed norms (**R5**), and some expectations are graded: it should not be too noisy near homes, its social cost scaling with proximity rather than a hard threshold (**R6**).

At the hub, access is supervised by a human site coordinator whose clearance counts as authorization for an AT to enter (**R7**) and confers a transient institutional status, e.g. priority carrier, until revoked (**R8**); crucially, only the empowered coordinator may issue it, and no AT may self-authorize entry (**R9**). Each expectation is precise enough to reason about formally, yet to this day each must today be hand-crafted from scratch, with no assurance the formula faithfully captures the norm and no way to co-verify it alongside the technical properties.

3 Related Work

Specification patterns are a proven device for taming specification complexity. Gruhn and Laue [28] introduce patterns for timed properties; Dong et al. [19] and Cheng et al. [36] provide timed-automata patterns and structured-English grammars; the classic Dwyer et al. catalog [20] and Vogel et al. [49] give comprehensive (real-time) pattern catalogs for SPIN/NuSMV and UPPAAL using observer

and debt patterns [30]. These catalogs target technical timing and behavioral properties. Our contribution adapts this mature methodology to social properties, pairing natural-language templates with (T)CTL semantics and observer automata.

The formalization of regulative norms in multi-agent systems has been studied extensively. OWL-POLAR [45] and ODRL [34] specify permissions, prohibitions, and obligations over digital assets but do not address temporal constraints or norm-lifecycle tracking [42]. Roshankish and Fornara’s T-Norm model [42] offers a comprehensive regulative taxonomy [18] designed as a benchmark for formal models, but does not target verification. SLEEC DSL [21] captures normative ambiguity via defeaters and fallbacks, reasoning about specification-level well-formedness through FOL* satisfiability rather than system-level model checking. Ågotnes et al. [2] generalize CTL to a Normative Temporal Logic; Alechina et al. [3] classify norms by whether they affect states, actions, transitions, or paths; Stratulat et al. [46] translate permissions and obligations into time-bounded deontic constraints; and Hjelmblom [32] interprets norms as constraints on labeled transition systems. These works address what norms mean but none provides reusable, pre-formalized patterns that drop directly into a model-checking workflow.

Constitutive norms have a rich but separate literature. Boella and van der Torre [10, 11] import the regulative/constitutive distinction into normative multi-agent systems, assigning constitutive norms the roles of classifying brute facts, defining institutional actions, and conferring powers. The counts-as relation is formalized as modal/conditional logics by Jones and Sergot [35] (institutionalised power) and taxonomized by Grossi et al. [26, 27] and Grossi and Jones [24]; these are reasoning logics with no model checker, observers, or engineering workflow. Operational frameworks make constitutive constructs first-class but verify them outside temporal model checking: InstAL [16, 41] compiles institutions (counts-as, empowerment, permission) to Answer Set Programming; norm-governed computational societies use the Event Calculus and the action language C+ [6]. The closest prior art is FIEVeL [47, 48], which pairs Searlean status functions with CTL model checking in SPIN, but as a bespoke language verifying ad-hoc soundness properties, with no natural-language templates and no reusable observer-pattern library. The deontic+temporal hybrids that do model-check, such as Normative Temporal Logic [2] and SLEEC/N-Check [21], remain explicitly regulative.

Every ingredient exists somewhere, but no artifact unifies (i) natural-language templates for counts-as, status-function, and institutional norms, (ii) (T)CTL observer-automaton semantics for each, and (iii) integration into one standard model-checking pipeline. Generic pattern catalogs [20, 49] supply the methodology but no constitutive content; constitutive formalisms supply the content but no reusable, observer-based, standard-tool catalog. Our catalog occupies this empty cell, porting the mature pattern-to-observer-to-model-checker methodology onto the established constitutive-norm categories and integrating both norm families into one workflow.

4 Background

We target *networks of timed automata* (NTA), the formalism underlying UPPAAL [8, 9]: finite automata with real-valued clocks composed over channels and shared bounded variables. A *state* is a

triple of locations, variable valuation, and clock zone, explored as a finite symbolic space. NTA is standard practice for timed cyber-physical systems, adding no modeling burden beyond the observer.

We specify norms in (T)CTL, pairing a *path quantifier* (A, all paths; E, some path) with a *temporal operator* (X next, F eventually, G globally, U until): $AG\ p$ is an invariant, $AF\ p$ an inevitability, $EF\ p$ reachability. TCTL adds real-time upper bounds ($AF_{\leq t}\ p$, $AG_{\leq t}\ p$). In UPPAAL, $A[\]=AG$, $E\<=>EF$, $A\<=>AF$ (X is undefined over dense time and omitted); timed bounds are not native syntax, so a clock-bearing observer encodes the deadline and the bound reduces to an ordinary $A[\]/A\<$ check.

We choose (T)CTL for three reasons. (i) It needs *branching-time* quantification: permission asserts a compliant path *exists* ($EF\ P$), not expressible with linear-time logics, whereas prohibition and obligation are universal (AG , AF); CTL’s possible/necessary (E/A) split matches the deontic permitted/mandatory one. (ii) Mature zone-based checkers (UPPAAL [8]) and strategy-synthesis extensions (Stratego [17], TIGA [14]) decide (T)CTL, so observers need no translation. (iii) STL targets robustness of *continuous* signals, not this branching distinction; graded compliance is instead recovered via weighted observers and cost optimization within the same UPPAAL(-Stratego) workflow (Section 5.1).

5 Social Norm Pattern Catalog

Observer patterns simplify the specification of formal constraints, removing the need to hand-craft temporal-logic formulas. A norm is encoded as an observer automaton [28, 49]: an additional timed automaton composed with the system that passively monitors it through shared variables and synchronization, without constraining or altering its behavior. The observer’s locations represent the lifecycle of the monitored norm as seen by the monitor: an initial/idle location q_0 (the norm’s triggering context has not occurred), an active/pending location $\bar{q}$ (the context S has occurred and the deontic condition P is being awaited or watched), an accepting location q_a (the permitted/obligatory condition has been observed), and an error location q_e (the norm has been violated). Verifying a norm reduces to a reachability question over these distinguished locations, e.g., “ q_e is never reachable” for a prohibition or obligation.

Each pattern is presented as a tripartite entry: a natural-language template, a (T)CTL formalization, and an observer automaton. Following the foundational dichotomy of normative systems [10, 43, 44], the catalog is organized into two families: regulative norms, which constrain antecedently possible behavior (Section 5.1), and constitutive norms, which create institutional facts, statuses, and powers (Section 5.2). We adopt separation of concerns [40] so each observer is a reusable, independently verifiable component [28]. Throughout, x is a real-time clock and t a threshold; S is a triggering context, P the regulated condition, and Q an exception.

5.1 Regulative Norm Patterns

Regulative norms directly constrain observable behavior [18] and are the category most frequently surfaced in stakeholder requirements [42]. We capture them with six patterns; their observers all share the lifecycle structure above and differ only in which location

is distinguished and how time is constrained (Figure 2 shows the representative timed case).

Permission Observer Pattern. Social norms first define what is socially permitted. Since permitting one behavior does not forbid others, the observer need only check that a permitted condition can be reached, that is, some path on which it eventually holds and which moves the observer to an accepting state q_a . Norms read “**P is socially permissible**” and are encoded with the CTL existential quantifier $EF\ P$, with the observer shown in Figure 1. While Governatori [23] questions temporal readings of permission, Alechina et al. [3] and Ågotnes [2] support the existential reading we adopt.

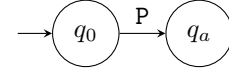


Figure 1: Permission Observer

Example 5.1. Along its route, an AT *may* slow to let a crossing field worker pass (**R1**); this is a courtesy, not a duty, so we require only that a compliant path *exists*: $EF\ StoppedForPed$, verified on the AT Transport model as $E\<> StoppedForPed$ (Section 7).

Prohibition Observer Pattern. Dual to permission, the prohibition pattern ensures a forbidden state/action never occurs: globally, “**P is socially prohibited**” is $AG(\neg P)$, with the observer reaching an error state q_e if P holds. A context-scoped variant, “**P is socially prohibited when S** [for t],” has untimed (Figure 2a) and timed (Figure 2b) forms:

$$AG(S \Rightarrow AG(\neg P)), \quad AG(S \Rightarrow AG_{\leq t} \neg P). \quad (1)$$

The timed form forbids P within t of S via the guard $x \geq t$ and the

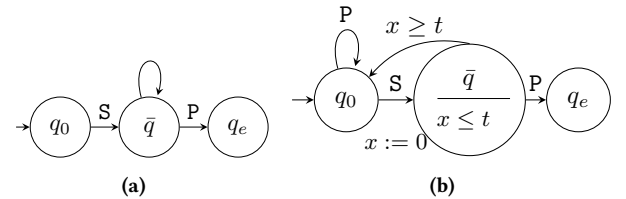


Figure 2: Prohibition Observer Patterns

invariant of $\bar{q}$; treating prohibitions as invariants follows Stratulat et al. [46].

Example 5.2. That same AT must *not* enter the residential zone after 5 pm (**R2**) ($S=resZone \wedge after5pm$, $P=driving$), a timed prohibition verified on AT Transport as $A[\] ! obs. Error$.

Obligation Observer Pattern. The obligation pattern requires that a behavior follow a situation: “**P is socially obligatory when S** [in t].” Its semantics use AF , capturing a deadline rather than an invariant [2, 3, 46]:

$$AG(S \Rightarrow AF(P)), \quad AG(S \Rightarrow AF_{\leq t}(P)). \quad (2)$$

The observer cycles between q_0 and $\bar{q}$, with a timed version adding an error state reached if P does not hold within t , as shown in Figure 3.

Example 5.3. When a queue of vehicles forms behind the slow AT on the shared road, it is obliged to pull over (bound $t=3$ in the timed variant) **(R3)** ($S=queued, P=PullOver$), verified on AT Transport as $A \langle \rangle \dots PullOver$.

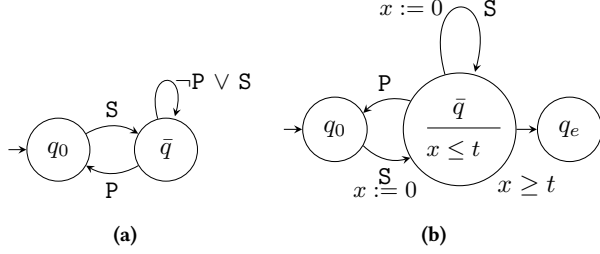


Figure 3: Obligation Observer Patterns

Exception Observer Pattern. Exception norms enforce a behavior but admit explicit, higher-priority overrides, modeling defeasible norms “**P is obligatory/prohibited after S except if Q.**” The exception Q enters the antecedent: prohibition becomes $AG(S \Rightarrow Q \vee AG(\neg P))$ and obligation $AG((S \wedge \neg Q) \Rightarrow AG(P))$. The observer is shown in Figure 4. The condition Q is exactly where engineers encode priority: a safety- or mission-critical Q deterministically overrides the social norm.

Example 5.4. The quiet-hours prohibition **(R2)** is defeasible: it is overridden when the AT carries perishable harvest **(R4)**, so $Q=perishable$ enters the antecedent. The Pest-Control Drone shares this shape (no-spray overridden by a declared pest emergency), verified as $control: A \langle \rangle (!obs.Err)$.

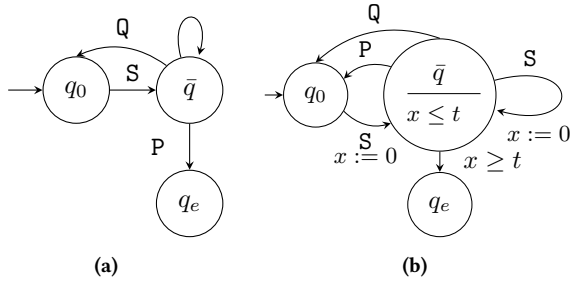


Figure 4: Exception Observer Patterns

Power-Inducing Observer Pattern. Some norms govern transitions between roles that determine which norms apply, rather than regulating behavior directly. Following [22], roles capture a component’s expected behavior, can be decided at run time, and are redefinable while preserving correctness. The pattern expresses “**when P, the system’s social role is B**” (Figure 5). It is most useful in composition, supplying role-dependent preconditions for obligations, prohibitions, and exceptions, thereby letting authority-compliance norms [39] enter autonomous decision-making. As we show in

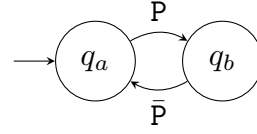


Figure 5: Power Observer Patterns

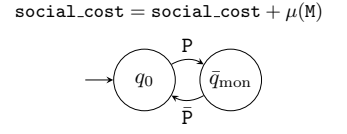


Figure 6: Fuzzy Prohibition Observer

Section 5.2, this pattern is the behavioral face of the constitutive Status-Function norm.

Example 5.5. Near field workers, an AT switches from STANDARD to DIFFERENTIAL **(R5)**, activating stricter yielding and speed norms. We realize this on the pedestrian model, where a vehicle detecting a pedestrian assumes the DIFFERENTIAL role that gates the applicable regulative norms.

Fuzzy Prohibition Pattern. Many social expectations are graded: “a tractor should not be too noisy near homes” scales with the perceived noise **(R6)**. We quantify the degree of violation as a continuous social cost: “the **social cost** increases by $\mu(M)$ when P .” The core is a membership function $\mu(M) \in \mathbb{R}_{\geq 0}$ mapping a system metric M (distance, noise, speed) to a normalized violation degree ($\mu=0$ is full compliance); e.g. for proximity, $\mu=0$ beyond 50m, $\mu=1$ within 10m, linear in between.

The observer (Figure 6) is a weighted automaton that accumulates `social_cost` via the update `social_cost += μ(M)` whenever P holds. This turns verification from reachability into optimization: using UPPAAL Stratego [17] or TIGA [14], an integral part of our approach for graded norms (Section 7), the goal becomes finding a strategy minimizing `technical_cost + w · social_cost`, where w weights social compliance against efficiency. This lets engineers explore the trade-off between being “correct” and being “socially acceptable.”

5.2 Constitutive Norm Patterns

The patterns above are all regulative: they constrain antecedently possible behavior. Yet, as Searle observed, normativity has a second, more fundamental kind. Constitutive norms do not regulate pre-existing activity; they create the very possibility of new forms of behavior and bring institutional facts into being, through rules of the canonical form “ X counts as Y in context C ” [43, 44]. Autonomous systems increasingly act within institutions, such as traffic-management authorities, delegated permissions, market and auction mechanisms, where what matters is not only what an agent does but what its action counts as: a manager’s signal counts as an authorization; detecting a pedestrian assigns a vehicle a deferential status; only an empowered controller can validly grant a slot. A catalog that claims to support social compliance must therefore address constitutive norms, not only regulative ones.

The normative-systems literature converges on three major categories of constitutive norm, which we adopt as the basis for our patterns: counts-as / classification [24, 26, 44], status-function (role) assignment [10, 44, 47], and institutional (normative) power [10, 33, 35]. Crucially for our purposes, each category admits a behavioral, state-based reading that is checkable by an observer over the system

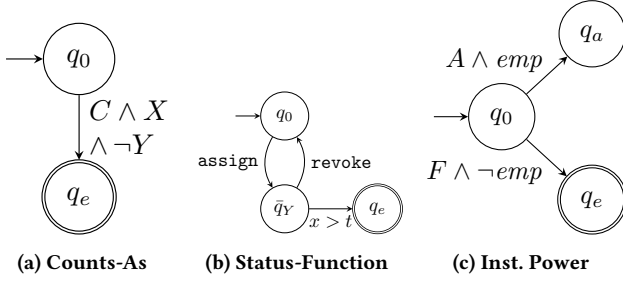


Figure 7: Observers for the constitutive-norm patterns.

model. We make this operationalization explicit for each pattern, and we are equally explicit about its boundaries (Section 6.1). The three observer automata are shown in Figure 7.

Count-as Observer Pattern. The most basic constitutive form classifies a brute fact or event as an institutional one within a context. Norms of this kind read In context C , X **counts as** Y . Read as contextual classification [25], the institutional fact Y must hold whenever the brute condition X holds in C . This is a maintenance invariant, encoded as $AG((C \wedge X) \Rightarrow Y)$. The observer (Figure 7a) reaches its violation location q_e exactly when $C \wedge X \wedge \neg Y$ becomes reachable. The pattern composes with the regulative catalog: the derived fact Y can serve as the trigger S or condition P of a regulative norm, so that classification activates downstream obligations or prohibitions.

Example 5.6. At the hub, a coordinator’s clearance *counts as* entry authorization (**R7**); we evaluate the structurally identical Traffic Intersection, where a granted slot counts as authorization to cross: $AG(\text{SlotGranted} \Rightarrow \text{hasGrantedSlot})$ (Section 7).

Status-Function Observer Pattern. A status-function norm imposes on an entity an institutional status (a role) it could not have by physical nature alone, carrying an associated bundle of deontic positions [44, 47]. Its template is “ X **is assigned status**, Y [until revoke | for t].” The status is established by a recognizing event and persists until revoked; its lifecycle is captured by $AG(\text{assign} \Rightarrow \text{status_Y})$, $A[\text{status_Y} \ W \ \text{revoke}]$, with an optional bounded-validity constraint $AG(\text{status_Y} \Rightarrow x \leq t)$. The observer (Figure 7b) holds the status in q_Y between assign and revoke and flags a bounded-validity violation. This pattern generalizes the Power-Inducing Observer Pattern: where Pattern 5 captures the behavioral consequence of holding a role, the Status-Function pattern captures the constitutive act that assigns it and constrains its validity; the role then gates the applicable regulative norms by composition.

Example 5.7. At the hub, the coordinator assigns an AT a transient status (e.g. priority carrier) that persists until revoked (**R8**). We evaluate the same lifecycle on the pedestrian model: detecting a pedestrian *assigns* the DIFFERENTIAL status, valid until it has crossed, gating the crossing-time limit (obs10) (Section 7).

Institutional Power Observer Pattern. An institutional power norm confers on a designated agent, acting in a specified role, the competence to bring about an institutional fact by performing a specified act: the empowered agent’s act counts as the creation of the fact [10, 35]. This is distinct from mere permission and from physical possibility (cf. Hohfeld’s power/liability [33]). Its template is

An agent **empowered** as R , by doing A , **brings about** F . The template is instantiated with four complementary properties, all discharged by ordinary reachability/invariance (UPPAAL has no next operator): (i) exercisability, $EF(\text{emp} \wedge A \wedge F)$; (ii) effect-establishment, $AG((A \wedge \text{emp}) \Rightarrow F)$; (iii) groundedness, $AG(F \Rightarrow \text{requested})$; and (iv) no spurious effect, $AG(\neg \text{emp} \Rightarrow \neg F)$. The observer (Figure 7c) establishes F on an empowered act and reaches q_e if an institutional effect F arises without one. Notably, the “no spurious effect” property is exactly what a “no unsolicited grants” prohibition was implicitly approximating in the regulative catalog; the Institutional Power pattern names it as a constitutive property and supplies its positive companions (exercisability, effect-establishment, and groundedness), which a prohibition cannot express.

Example 5.8. Only the empowered site coordinator may grant hub clearance; an AT cannot self-authorize entry (**R9**). We evaluate this on the Traffic Intersection, where only the empowered manager grants a slot, verifying exercisability, effect-establishment, groundedness, and no-spurious-effect (Section 7.6).

6 Completeness and Comparison

We analyze the catalog’s completeness over both norm families (Section 6.1) and compare it to existing frameworks (Section 6.2).

6.1 Completeness, Scope, and Boundaries

A pattern catalog should cover a well-defined space of norms without gaps. We ground completeness in independent taxonomies for each family, so the claim does not rest on our own categorization.

Any regulative norm is characterized by its deontic modality (require, permit, forbid), its activation condition S , and its defeasibility [2, 3, 18, 42]. Our six patterns cover all combinations: permission, prohibition, and obligation address the modalities; the trigger S provides conditional activation; the Exception Observer Pattern handles discrete defeasibility and the Fuzzy Prohibition Pattern graded defeasibility; the Power-Inducing Observer Pattern covers role-dependent activation. We substantiate this by mapping the catalog to the T-Norm taxonomy [42], chosen as one of the most comprehensive classifications (four norm types with sub-types plus three exception categories) and designed as a benchmark for comparing formal models. Table 2 shows every sub-type is expressible by a single pattern or composition, and that the catalog extends beyond the taxonomy via graded compliance, which no existing classification covers.

For the constitutive family we adopt the categorization the literature converges on, attributable to specific sources: counts-as / classification (Searle’s “ X counts as Y in C ” [44], taxonomized by Grossi et al. [24, 26]), status-function assignment (Searle’s status functions [44], made verifiable by Viganò and Colombetti [47]), and institutional power (Hohfeld’s power/liability [33] as institutionalised power by Jones and Sergot [35], conferred to roles by Boella and van der Torre [10]). Boella and van der Torre [11] characterize the roles of constitutive norms in MAS by classifying brute facts, defining institutional actions, and conferring powers, which our three patterns (Count-as Observer Pattern, Status-Function Observer Pattern, Institutional Power Observer Pattern) cover in their state-based, verifiable reading. The institutional facts derived by counts-as and the powers exercised compose with the regulative

Table 1: Comparison across normative frameworks spanning both norm families. Reg. = regulative; Const. = constitutive; MC = model checking; Obs. = observer-based; Pat. = reusable specification patterns; NL = natural-language templates; WF = fits a standard engineering/model-checking workflow. RA = Refinement-Analysis [12, 13]; ✓ = full, ~ = partial/indirect, – = none.

Approach	Reg.	Const.	Logic	MC	Obs.	Pat.	NL	WF	RA
Our Catalog	✓	✓	CTL/TCTL	✓	✓	✓	✓	✓	~
T-Norm [42]	✓	–	operational	–	–	–	✓	–	–
SLEEC / N-Check [21]	✓	–	FOL* / LTL	~	–	–	✓	~	–
Normative Temporal Logic [2]	✓	–	NTL (CTL)	~	–	–	–	–	–
Counts-as logic [24, 35]	–	✓	modal	–	–	–	–	–	–
InstAL [16, 41]	~	✓	ASP	–	~	~	–	~	–
Event Calculus societies [6]	✓	✓	Event Calc. / C+	–	–	~	–	~	–
FIEVeL [47, 48]	–	~ (status)	CTL (SPIN)	✓	–	~	–	✓	–
Generic patterns [20, 49]	–	–	LTL/CTL/TCTL	✓	✓	✓	✓	✓	~

Table 2: Mapping of the T-Norm regulative taxonomy [42] to catalog patterns.

Norm Type	Pattern(s)	(T)CTL
<i>Type 1: Activation condition</i>		
Event/State-activated	Patterns 2–4 via S	$AG(S \Rightarrow \dots)$
Combined (event+state)	Patterns 2–4, S conj.	$AG((e \wedge s) \Rightarrow \dots)$
<i>Type 2: Debtor specificity</i>		
Specific debtor	Patterns 2, 3	Standard instantiation
General (role-based)	Pattern 5 + Patterns 2, 3	Role gates norm set
<i>Type 3: Temporal constraints</i>		
Deadline obligation	Pattern 3 (timed)	$AG(S \Rightarrow AF_{\leq t} P)$
Timed/Permanent prohib.	Pattern 2	$AG(S \Rightarrow AG_{\leq t} \neg P)$
Standing/Repetitive obl.	Pattern 3 (+4)	Cycling obs.; clock reset
<i>Type 4: Normative power</i>		
Explicit / implicit power	Pattern 5 + 2, 3	Role transition
<i>Exceptions / Beyond taxonomy</i>		
Exception (activation/rel.)	Pattern 4	$AG(S \Rightarrow Q \vee \dots)$
Graded compliance	Pattern 6 (Fuzzy)	Optimization objective

patterns (a derived fact Y becomes a regulative trigger S), so the two families interlock rather than sit side by side.

We are explicit about what falls outside a finite-state observer. For counts-as, only the resulting classification invariant is checkable; the proper-classificatory reading (classification genuinely due to the institution, not logical subsumption [25]) quantifies over derivations and is out of scope. For institutional power, the static state-of-affairs effects are checkable, but the full Hohfeldian liability/immunity structure and the unbounded run-time introduction of new institutional categories are only partially captured. These boundaries delimit the catalog: it operationalizes the behavior- and institution-governing core of social normativity, not its open-ended, contestable periphery.

6.2 Comparison to other approaches

Table 1 compares the catalog with prominent frameworks across both families. The pattern is striking: regulative frameworks (T-Norm, SLEEC, NTL) ignore constitutive norms; constitutive formalisms (counts-as logics, InstAL, Event-Calculus societies) define them richly but do not deliver reusable, observer-based, standard-tool model checking; the one constitutive model-checking effort,

FIEVeL, is bespoke (no templates, no reusable pattern library, no regulative coverage); and the generic pattern catalogs supply exactly the template/observer/model-checker methodology but contain no notion of social or constitutive norms. Only our catalog spans both families with natural-language templates, (T)CTL observer realizations, and integration into one model-checking workflow. Concretely, this yields end-to-end traceability from an informal requirement to a verifiable observer and result, and avoids the inter-formalism “semantic gap.” Other frameworks retain complementary strengths outside this scope, such as T-Norm’s lifecycle tracking, SLEEC’s richer surface syntax, InstAL’s non-monotonic reasoning, and the abductive and runtime analyses of Event-Calculus institutions; recent stable-model approaches also couple constitutive norms with checking, though not via CTL/TCTL observers. Our contribution is the unification, not the replacement, of these lines.

7 Evaluation

We evaluate the catalog on seven UPPAAL models spanning smart farming and autonomous transport. Our goals are to show that the patterns apply across diverse scenarios, capture realistic social complexity, and remain computationally tractable. We address four research questions:

- RQ1:** To what extent can the patterns be reused across scenarios?
- RQ2:** What is the verification performance, and how does the approach scale with system size and the number of norms?
- RQ3:** How does the framework support co-verification of hard safety constraints with soft, defeasible, and fuzzy social norms requiring prioritization and trade-offs?
- RQ4:** Can the catalog quantitatively measure the impact of social norms on system performance?

Experimental setup. All models are networks of timed automata verified with UPPAAL 5.0.0 (breadth-first search, conservative space optimization) on an AMD EPYC 7452 workstation (Linux, 64 GB RAM). Reachability properties ($A[], E\Diamond$) use the symbolic model checker; graded norms and trade-off analysis use UPPAAL Stratego [17] for strategy synthesis followed by statistical model checking (200 runs). We report CPU time and resident memory from UPPAAL’s per-query summary, taking the median of three runs

(state counts are deterministic). All models, queries, and scripts are publicly available.¹

7.1 Use cases and norm mapping

Table 3 lists, for each system, its social norms, their source, the pattern used to formalize each, the verification query, and the outcome. The *Collaborative Farm* explores close-quarters human–robot collaboration (a drone balancing harvesting efficiency against a worker’s safety and comfort). The *Pest Control Drone* adds an environmental constraint (a no-spray zone) and quantifies the energy cost of compliance. *Highway Merging* tests tiered priorities (a hard obligation to yield to an emergency ambulance vs. a soft courtesy to yield to a non-emergency vehicle). The *Traffic Intersection* models multi-agent coordination for safety and fairness, and its pedestrian extension introduces role-dependent prioritization of pedestrian right-of-way over throughput. The table doubles as a record of how each informal requirement is instantiated as a concrete pattern, query, and result, directly answering reviewers’ request for end-to-end traceability.

7.2 RQ1: Reusability

We consider a pattern *successfully reused* when (i) a single pattern, instantiated only by binding its parameters (S, P, Q, t, μ), formalizes social norms of *different content in different domains*, and (ii) the resulting observer and query verify correctly against the corresponding model. Reuse in our framework is manual but *systematic*: the natural-language template and the parameter binding mediate the mapping from informal requirement to formal observer, and the correctness of that mapping is validated by the very model-checking result: a passing safety query confirms the observer detects the intended violations, and an intentionally failing courtesy query (Table 3, “opt.”) confirms the observer correctly captures a defeasible norm.

By this criterion, every pattern is reused across at least two domains. The Prohibition Observer Pattern formalizes both “a drone must not operate too close to its human collaborator” (Collaborative Farm) and “stable, non-ambiguous signaling” (Traffic Intersection); the Obligation Observer Pattern formalizes both yielding to an emergency vehicle (Highway Merging) and re-engaging after a back-off (Traffic Intersection); the Exception Observer Pattern formalizes the no-spray-zone norm overridden by a pest emergency. We emphasize that pure collision avoidance is a *technical safety* property, co-verified but not itself a social norm; the social norms here concern human comfort, courtesy, fairness, and right-of-way (Table 3). Because patterns abstract the *structure* of a norm from its content, an instantiated observer is reusable in any context sharing that structure.

7.3 RQ2: Performance and Scalability

Table 4 reports verification cost per use case. Reachability checks on the farming and single-vehicle transport models complete in single-digit to low-tens of milliseconds over hundreds to a few thousand states, using ≈ 18 MB. The pedestrian intersection is markedly larger, with a 2.7-million-state space verified in ≈ 31 s with 228 MB, yet remains tractable on commodity hardware, demonstrating that the

¹<https://github.com/lucabrodoshl/SocialNormPatternCatalogue>

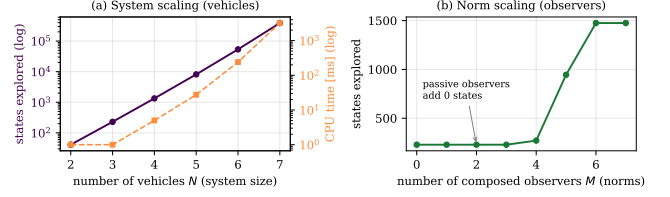


Figure 8: Scaling on the Traffic Intersection model.

approach scales to genuinely complex multi-agent scenarios with roles. Notably, on that model the cost of verifying `obs1` and `obs2` is identical (≈ 31 s), because the state space is dominated by the system, not by the observer being checked.

We separate two scaling dimensions (Figure 8). *System scaling*: varying the number of vehicles N , the reachable state space grows exponentially ($N=2$: 41 to $N=7$: 387,815 states; 1 ms to 3.2 s), the intrinsic state-space-explosion cost of concurrency [8], independent of our contribution. *Norm scaling*: composing M observers with a fixed system, the first three add *no* states (passive monitors), and only clock-bearing observers add a small bounded increment ($M=6$: 1,475 states, 7.5 ms). Since each observer is non-intrusive and properties can be verified independently [12], per-property cost is effectively *independent of the number of norms*, the architectural payoff of separation of concerns. This also yields *modeling scalability*: adding or adapting a norm edits one observer without touching the agent model, localizing re-verification.

7.4 RQ3: Conflict Resolution and Trade-offs

We apply the three prioritization mechanisms of Section 6.1. *Role-based hierarchy*: in the pedestrian intersection, a vehicle assuming the `DIFFERENTIAL` role is bound by stricter norms obligating it to prioritize pedestrian courtesy over crossing efficiency; `A[] !obs10`. Error confirms that the system correctly grants a socially-aware vehicle a longer crossing time while still constraining a `STANDARD` one. *Explicit exceptions*: in the Pest Control Drone, the no-spray prohibition is overridden by a declared pest emergency, letting a critical task supersede a social constraint. *Cost-based trade-offs*: in Highway Merging we model the tension between throughput and the courtesy of yielding to a non-emergency vehicle. We synthesize two strategies with UPPAAL Stratego and measure the accumulated social cost (Figure 9) over 200 SMC runs. The baseline *All-Cars* strategy, free to ignore the courtesy, incurs a mean social cost of **4.9**; the *Socially-Aware* strategy, which enforces yielding, incurs **7.1**. This ability to impose a normative priority and then *measure* its systemic consequence is the core of our approach to conflict resolution.

7.5 RQ4: Behavioral Impact of Social Norms

The framework quantifies compliance cost. *Social cost* is the accumulated membership-function value (Section 5.1): whenever the social condition holds, the weighted observer adds $\mu(M)$ to `social_cost`; here μ counts each un-yielded courtesy as one unit, so the metric is the expected number of social “slights” per run (Highway Merging: 4.9 vs. 7.1). The *Pest Control Drone* shows the energy cost of environmental compliance: a no-spray-respecting strategy lowers

Table 3: Use cases: social norms, their source, the catalog pattern used, the verification query, and the outcome. Technical safety properties (e.g. collision avoidance) are co-verified but listed separately from social norms. “opt.” marks defeasible norms whose query is *intended* to fail, demonstrating optionality.

Use case	Social norm (informal)	Source	Pattern	Query	Result
AT Transport	Pull over when vehicles queue behind No driving in residential zone after 5 pm May slow for pedestrian at crosswalk	Stakeholder [31] Community / noise [31] Driving courtesy	Timed Obligation	$A \langle \dots \text{PulledOver} \rangle$	✓
			Timed Prohibition	$A[] \text{!obs.Error}$	✓
			Permission	$E \langle \text{StoppedForPed} \rangle$	✓
Collaborative Farm	Signal before resuming after a halt Reduce speed when approaching worker	HRI predictability Psychological comfort	Timed Obligation	$A[] \text{!obs2.Error}$	✓
			Prohibition	$A[] \text{!obs4.Error}$	✓
Highway Merging	Yield to emergency ambulance (sirens on) Yield to non-emergency vehicle (courtesy) Cost of not yielding (inconvenience)	Traffic law / life-safety Traffic politeness Social friction	Obligation	$A[] \text{!S0.Error}$	✓
			Permission	$A[] \text{!S02.Error}$	opt.
			Fuzzy Prohibition	$E[\dots](\text{max: social_cost})$	✓
Traffic Intersection	Timed back-off after denial (fairness) Re-engage after back-off period Stable, non-ambiguous signaling	Fairness / efficiency Liveness / no gridlock Predictability	Timed Prohibition	$A[] \text{!obs4alt.Error}$	✓
			Obligation	$A[] \text{!obs6.Error}$	✓
			Timed Prohibition	$A[] \text{!obs2.Error}$	✓
+ Pedestrian	Yield to crossing pedestrian (deferential role) Role-dependent crossing-time limit	Pedestrian safety Context-dependent norm	Power-Inducing + Obl. Timed Prohibition	$A[] \text{!obs8.NotPass}$ $A[] \text{!obs10.Error}$	opt. ✓
Pest Control Drone	Do not spray near organic farm	Environmental	Prohibition + Exception	$\text{control:} A \langle \text{!obs.Err} \rangle$	✓

Table 4: Verification performance per use case (representative reachability query; UPPAAL 5.0.0, median of 3 runs).

Use case (NTA)	States Explored	CPU time	Memory
AT Transport	696	3 ms	18 MB
Collaborative Farm	3,239	12 ms	19 MB
Highway Merging	2,161	7 ms	18 MB
Traffic Intersection	1,475	9 ms	18 MB
Traffic + Pedestrian	2,744,563	31.2 s	228 MB
Pest Control Drone	(synth.+SMC)	32–141 ms	23 MB

the expected minimum power reserve from **18.84** (mission-only) to **13.31** (compliant), a $\approx 29\%$ smaller safety margin, as the drone must loiter to avoid the farm. The *Collaborative Farm* compares three drone behaviors by final energy: a non-social baseline finishes highest, a “slow-down” drone is a moderate compromise, and a “wait-for-human” drone pays the most, idling without progress. Across all three, compliance carries a measurable, strategy-dependent cost, letting engineers compare alternatives (*wait* vs. *slow down*) and choose the best efficiency/compliance balance for a deployment.

7.6 Constitutive Norms in Practice

A central claim of this work is that constitutive norms are not merely theoretical: they are already *latent* in autonomous-systems models, and our patterns make them explicit and verifiable in the same workflow. We demonstrate this on the Traffic Intersection, whose manager embodies an *institutional power*: only the manager may grant a crossing slot, and a grant *counts as* authorization to cross. The regulative catalog could express only the negative “no unsolicited grants” as an opaque prohibition (obs3); the Institutional Power Observer Pattern names this as a constitutive property and adds its positive companions. All four properties verify with the same tool at negligible cost: the power is *exercisable* ($E \langle \text{manager.SlotGranted} \rangle$; 8 states, < 1 ms); the grant *establishes* the institutional fact ($AG(\text{SlotGranted} \Rightarrow \text{hasGrantedSlot})$; 1,475 states,



Figure 9: Highway Merging: distribution of accumulated social cost under the baseline All-Cars strategy (mean 4.9) vs. the Socially-Aware strategy (mean 7.1).

7 ms); the fact is *grounded* in a request ($AG(\text{hasGrantedSlot} \Rightarrow \text{requested})$; 7 ms); and there is *no spurious effect* ($A[] \text{!obs3.Error}$; 8 ms). The same invariant “a granted slot *counts as* authorization” instantiates the Count-as Observer Pattern. The Status-Function Observer Pattern appears in the pedestrian extension: detecting a pedestrian *assigns* a vehicle the DEFERENTIAL status, valid until the pedestrian has crossed, after which the role-dependent crossing-time norm (obs10) applies, the constitutive act that the Power-Inducing Observer Pattern previously realized only behaviorally. Thus the constitutive patterns integrate into the very same model-checking workflow and incur the same negligible cost as the regulative ones, on a model not built for them.

8 Discussion and Limitations

Our catalog deliberately targets social norms, both regulative and constitutive, that can be operationalized over observable states and

events; it does not, and does not claim to, capture the full tacit, contextual, and culturally variable subtlety of human normativity. The engineering value of the catalog lies precisely in giving the practically relevant class of behavior- and institution-governing norms a reusable, traceable, verifiable form. Contextual variability is handled by *parameterization* (thresholds (t), proximity bounds, membership functions (μ), and roles are instantiated per deployment, so a norm that differs across regions is captured by re-instantiation rather than by a new pattern) but norms grounded in genuinely subjective judgment (e.g. “intent” or “aggressiveness”) require first defining monitorable proxies, which our framework supports but does not automate.

(T)CTL makes highly non-linear conditional logic cumbersome to encode as observers (a deliberate trade of declarative convenience for verifiability); the framework relies on an *observability* assumption (norm conditions must map to monitorable states/events); pattern *selection* is currently manual, with automated guidance left to future work; and we address a-priori verification against a static norm set, not run-time norm evolution. The constitutive boundaries of Section 6.1 apply additionally.

Our use cases span two domains and varied norm types but are author-constructed; a controlled user study would more directly establish practical benefit. Performance figures are UPPAAL- and hardware-specific, but the relative trends (system-driven explosion vs. near-flat observer overhead) are formalism-level. SMC results are statistical estimates over 200 runs; reported means and trade-off directions are stable across seeds.

9 Summary and Outlook

We presented a standardized catalog of reusable observer patterns that bridges informal social requirements and their formal verification in autonomous systems, covering *both* fundamental kinds of norm: regulative norms that constrain behavior, and constitutive norms such as counts-as classification, status-function assignment, and institutional power, which create the institutional facts within which autonomous systems increasingly act. Each pattern provides a tripartite interface: a natural-language template, a (T)CTL formalization, and an observer automaton, enabling engineers to co-verify social and technical properties within established model-checking workflows, without hand-crafting temporal formulas, switching tools, or bridging formalisms. We grounded the completeness of each family in an independent taxonomy, were explicit about the boundaries of finite-state observability, and positioned the catalog against both regulative and constitutive frameworks, showing it uniquely unifies templates, (T)CTL observer semantics, and standard-tool model checking. Our evaluation over seven UPPAAL models reported verification time, memory, and state-space scaling, showing that per-property cost is dominated by the system and effectively independent of the number of norms, demonstrated reuse, conflict resolution, and quantitative trade-off analysis, and verified constitutive norms (institutional power, counts-as, status) directly on a model that was not built for them, at negligible cost.

Several directions remain open. Within the catalog’s boundaries, the proper-classificatory reading of counts-as, the full Hohfeldian liability/immunity structure, and run-time introduction of new institutional categories call for richer foundations than finite-state

observers. More broadly, the framework targets a-priori verification against static norms; dynamic norm evolution and run-time adaptation are the most significant open challenge. Controlled user studies would establish the catalog’s practical benefit and reveal usability barriers in pattern selection, which automated tooling could lower. Taken together, this work shows that the institutional fabric of social compliance, not only its behavioral surface, can be brought within reach of the verification workflows engineers already use.

Acknowledgments

Supported by the Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia. Co-funded by the European Union. Grant number EFRE-20800498.

References

- [1] 2024. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>. Official Journal of the European Union, L series, 12 July 2024.
- [2] Thomas Ågotnes, Wiebe van der Hoek, Juan A. Rodríguez-Aguilar, Carles Sierra, and Michael Wooldridge. 2009. *A Temporal Logic of Normative Systems*. Springer Netherlands, Dordrecht, 69–106. doi:10.1007/978-1-4020-9084-4_5
- [3] Natasha Alechina, Brian S. Logan, and Mehdi M. Dastani. 2018. Modeling Norm Specification and Verification in Multiagent Systems. *FLAP* 5 (2018), 457–490. <https://api.semanticscholar.org/CorpusID:197538079>
- [4] R. Alur, C. Courcoubetis, and D. Dill. 1993. Model-Checking in Dense Real-Time. *Information and Computation* 104, 1 (1993), 2–34. doi:10.1006/inco.1993.1024
- [5] Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235. doi:10.1016/0304-3975(94)90010-8
- [6] Alexander Artikis, Marek J. Sergot, and Jeremy V. Pitt. 2009. Specifying Norm-Governed Computational Societies. *ACM Transactions on Computational Logic* 10, 1 (2009), 1:1–1:42. doi:10.1145/1459010.1459011
- [7] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.
- [8] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. 1996. UPPAAL—a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON Workshop on Hybrid Systems III: Verification and Control: Verification and Control* (New Brunswick, NeW Jersey, USA). Springer-Verlag, Berlin, Heidelberg, 232–243.
- [9] Johan Bengtsson and W. Yi. 2004. Timed automata: Semantics, algorithms and tools. *Timed Automata: Semantics, Algorithms and Tools* 3098 (01 2004), 87–124.
- [10] Guido Boella and Leendert van der Torre. 2004. Regulative and Constitutive Norms in Normative Multiagent Systems. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*. AAAI Press, Whistler, Canada, 255–265.
- [11] Guido Boella and Leendert van der Torre. 2006. Constitutive Norms in the Design of Normative Multiagent Systems. In *Computational Logic in Multi-Agent Systems (CLIMA VI) (Lecture Notes in Computer Science, Vol. 3900)*, Francesca Toni and Paolo Torroni (Eds.). Springer, 303–319. doi:10.1007/11750734_17
- [12] Luca Brodo, Giuseppe Scalora, and Stefan Henkler. 2026. Property Refinement in Linear Temporal Logic: Formal Semantics and Algorithms for Software Verification. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering (Montreal, Canada) (FSE ’26)*. ACM. <https://conf.researchr.org/home/fse-2026> To appear.
- [13] Luca Brodo, Giuseppe Scalora, and Stefan Henkler. 2026. What Do We Really Need? Identifying Minimal Sufficient Temporal Properties via Refinement. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion ’26)*. ACM, Rio de Janeiro, Brazil, 1–2. doi:10.1145/3774748.3795661 To appear in 2026.
- [14] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. 2005. Efficient On-the-Fly Algorithms for the Analysis of Timed Games. In *CONCUR 2005 – Concurrency Theory*, Martin Abadi and Luca de Alfaro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 66–80.
- [15] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 1999. *Model Checking*. MIT Press.
- [16] Owen Cliffe. 2007. *Specifying and Analysing Institutions in Multi-Agent Systems Using Answer Set Programming*. Ph. D. Dissertation. University of Bath.
- [17] Alexandre David, Peter Gjø Jensen, Kim Guldstrand Larsen, Marius Mikućionis, and Jakob Haahr Taankvist. 2015. Uppaal Stratego. In *Tools and Algorithms for the Construction and Analysis of Systems*, Christel Baier and Cesare Tinelli (Eds.).

- Springer Berlin Heidelberg, Berlin, Heidelberg, 206–211.
- [18] Frank Dignum. 1999. Autonomous agents and norms. *Artificial Intelligence and Law* 7 (03 1999), 69–79. doi:10.1023/A:1008315530323
 - [19] Jin Song Dong, Ping Hao, Shengchao Qin, Jun Sun, and Wang Yi. 2008. Timed Automata Patterns. *IEEE Transactions on Software Engineering* 34, 6 (2008), 844–859. doi:10.1109/TSE.2008.52
 - [20] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. Patterns in Property Specifications for Finite-State Verification. In *Proceedings of the 21st International Conference on Software Engineering (ICSE 1999)*. ACM, 411–420. doi:10.1145/302405.302672
 - [21] Nick Feng, Lina Marsso, Sinem Getir Yaman, Yesugen Baatartogtokh, Reem Ayad, Victoria Oldemburgo De Mello, Beverley Townsend, Isobel Standen, Ioannis Stefanakos, Calum Imrie, Genaina Nunes Rodrigues, Ana Cavalcanti, Radu Calinescu, and Marsha Chechik. 2024. Analyzing and Debugging Normative Requirements via Satisfiability Checking. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. ACM, 1–12. doi:10.1145/3597503.3639093
 - [22] Holger Giese, Daniela Schilling, Matthias Tichy, Sven Burmester, Wilhelm Schäfer, and Stephan Flake. 2003. Towards the Compositional Verification of Real-Time UML Designs. In *Proceedings of the 9th European Software Engineering Conference held jointly with the 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE '03)*. ACM, 38–47.
 - [23] Guido Governatori. 2015. Thou shalt is not you will. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law (San Diego, California) (ICAIL '15)*. Association for Computing Machinery, New York, NY, USA, 63–68. doi:10.1145/2746090.2746105
 - [24] Davide Grossi and Andrew J. I. Jones. 2013. Constitutive Norms and Counts-as Conditionals. In *Handbook of Deontic Logic and Normative Systems, Volume 1*, Dov Gabbay, John Horty, Ron van der Meyden, Xavier Parent, and Leon van der Torre (Eds.). College Publications, London, 407–441.
 - [25] Davide Grossi, John-Jules Ch. Meyer, and Frank Dignum. 2006. Classificatory Aspects of Counts-as: An Analysis in Modal Logic. *Journal of Logic and Computation* 16, 5 (2006), 613–643. doi:10.1093/logcom/exl027
 - [26] Davide Grossi, John-Jules Ch. Meyer, and Frank Dignum. 2006. Counts-as: Classification or Constitution? An Answer Using Modal Logic. In *Deontic Logic and Artificial Normative Systems (DEON 2006) (Lecture Notes in Computer Science, Vol. 4048)*. Springer, 115–130. doi:10.1007/11786849_11
 - [27] Davide Grossi, John-Jules Ch. Meyer, and Frank Dignum. 2008. The Many Faces of Counts-as: A Formal Analysis of Constitutive Rules. *Journal of Applied Logic* 6, 2 (2008), 192–217.
 - [28] Volker Gruhn and Ralf Laue. 2006. Patterns for Timed Property Specifications. *Electronic Notes in Theoretical Computer Science* 153, 2 (2006), 117–133. doi:10.1016/j.entcs.2005.10.035
 - [29] Xu Guo, Hsiao-Hua Lin, Kenro Yatake, and Toshiaki Aoki. 2014. An UPPAAL Framework for Model Checking Automotive Systems with FlexRay Protocol. In *Formal Techniques for Safety-Critical Systems (FTSCS 2013) (Communications in Computer and Information Science, Vol. 419)*. Springer, 51–66. doi:10.1007/978-3-319-05416-2_4
 - [30] K. Havelund, K. G. Larsen, and A. Skou. 1999. Formal verification of a power controller using the real-time model checker UPPAAL. In *ARTS (LNCS, Vol. 1601)*. Springer, 277–298.
 - [31] S. Henkler, K. Rother, and Achim Rettberg. 2022. 5G-LANDWIRTSCHAFT-ML. <https://www.hshl.de/forschung-unternehmen/forschungsprojekte/forschungsprojekte-im-themenfeld-eingebettete-systeme/5g-landwirtschaft-ml/> Grant No. 005-2108-0039, funded by Ministerium für Wirtschaft, Industrie, Klimaschutz und Energie des Landes Nordrhein-Westfalen.
 - [32] Magnus Hjelmblom. 2014. Normative Positions within Norm-Regulated Transition System Situations. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Vol. 3. 238–245. doi:10.1109/WI-IAT.2014.173
 - [33] Wesley Newcomb Hohfeld. 1913. Some Fundamental Legal Conceptions as Applied in Judicial Reasoning. *Yale Law Journal* 23, 1 (1913), 16–59.
 - [34] Renato Iannella, Michael Steidl, Stuart Myles, and Victor Rodríguez-Doncel. 2018. ODRL Vocabulary & Expression 2.2. W3C Recommendation, <https://www.w3.org/TR/odrl-vocab/>. Accessed: 2026-08-11.
 - [35] Andrew J. I. Jones and Marek Sergot. 1996. A Formal Characterisation of Institutionalised Power. *Logic Journal of the IGPL* 4, 3 (1996), 427–443.
 - [36] Sascha Konrad and Betty H. C. Cheng. 2005. Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering (St. Louis, MO, USA) (ICSE '05)*. Association for Computing Machinery, New York, NY, USA, 372–381. doi:10.1145/1062455.1062526
 - [37] Per Lange Laursen, Van Anh Thi Trinh, and Anne Elisabeth Haxthausen. 2020. Formal Modelling and Verification of a Distributed Railway Interlocking System Using UPPAAL. In *Leveraging Applications of Formal Methods, Verification and Validation: Applications (ISoLA 2020) (Lecture Notes in Computer Science, Vol. 12478)*. Springer, 415–433. doi:10.1007/978-3-030-61467-6_27
 - [38] Marius Mikučionis, Kim Guldstrand Larsen, Jacob Illum Rasmussen, Brian Nielsen, Arne Skou, Steen Ulrik Palm, Jan Storbak Pedersen, and Poul Hougaard. 2010. Schedulability Analysis Using Uppaal: Herschel-Planck Case Study. In *Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2010) (Lecture Notes in Computer Science, Vol. 6416)*. Springer, 175–190. doi:10.1007/978-3-642-16561-0_21
 - [39] Stanley Milgram. 1963. Behavioral Study of Obedience. *Journal of Abnormal and Social Psychology* 67, 4 (1963), 371–378. doi:10.1037/h0040525
 - [40] Harold Ossher and Peri Tarr. 2002. *Multi-Dimensional Separation of Concerns and the Hyperspace Approach*. Springer US, Boston, MA, 293–323. doi:10.1007/978-1-4615-0883-0_10
 - [41] Julian Padget, Emad ElDeen Elakehal, Tingting Li, and Marina De Vos. 2016. InstAL: An Institutional Action Language. In *Social Coordination Frameworks for Social Technical Systems*. Law, Governance and Technology Series, Vol. 30. Springer, 101–124. doi:10.1007/978-3-319-33570-4_6
 - [42] Soheil Roshankish and Nicoletta Fornara. 2024. How to Formalize Different Types of Norms in Multi-agent Systems: A Methodology Focused on the T-Norm Model. *SN Computer Science* 5, 6 (2024), 749. doi:10.1007/s42979-024-03052-4
 - [43] John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.
 - [44] John R. Searle. 1995. *The Construction of Social Reality*. Free Press, New York.
 - [45] Murat Sensoy, Timothy Norman, Wamberto Vasconcelos, and Katia Sycara. 2011. OWL-POLAR: A framework for semantic policy representation and reasoning. *Web Semantics: Science, Services and Agents on the World Wide Web* 12–13 (01 2011). doi:10.1016/j.websem.2011.11.005
 - [46] Tiberiu Stratulat, F. Clerin-Debart, and Patrice Enjalbert. 2001. Temporal reasoning: An application to normative systems. *Proceedings of the International Workshop on Temporal Representation and Reasoning*, 41–47. doi:10.1109/TIME.2001.930696
 - [47] Francesco Viganò and Marco Colombetti. 2007. Specification and Verification of Institutions Through Status Functions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II (COIN 2006) (Lecture Notes in Computer Science, Vol. 4386)*. Springer, 115–129. doi:10.1007/978-3-540-74459-7_8
 - [48] Francesco Viganò and Marco Colombetti. 2008. Model Checking Norms and Sanctions in Institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems III (COIN 2007) (Lecture Notes in Computer Science, Vol. 4870)*. Springer, 316–329. doi:10.1007/978-3-540-79003-7_23
 - [49] Thomas Vogel, Marc Carwehl, Genaina Nunes Rodrigues, and Lars Grunske. 2023. A property specification pattern catalog for real-time system verification with UPPAAL. *Information and Software Technology* 154 (Feb. 2023), 107100. doi:10.1016/j.infsof.2022.107100