Algorithmic Ethics: Formalization and Verification of Autonomous Vehicle Obligations

COLIN SHEA-BLYMYER and HOUSSAM ABBAS, Oregon State University

We develop a formal framework for automatic reasoning about the obligations of autonomous cyber-physical systems, including their social and ethical obligations. Obligations, permissions and prohibitions are distinct from a system's mission, and are a necessary part of specifying advanced, adaptive AI-equipped systems. They need a dedicated deontic logic of obligations to formalize them. Most existing deontic logics lack corresponding algorithms and system models that permit automatic verification. We demonstrate how a particular deontic logic, Dominance Act Utilitarianism (DAU) [23], is a suitable starting point for formalizing the obligations of autonomous systems like self-driving cars. We demonstrate its usefulness by formalizing a subset of Responsibility-Sensitive Safety (RSS) in DAU; RSS is an industrial proposal for how self-driving cars should and should not behave in traffic. We show that certain logical consequences of RSS are undesirable, indicating a need to further refine the proposal. We also demonstrate how obligations can change over time time, which is necessary for long-term autonomy. We then demonstrate a model-checking algorithm for DAU formulas on weighted transition systems, and illustrate it by model-checking obligations of a self-driving car controller from the literature.

CCS Concepts: • **Computer systems organization** → **Robotic control**; • **Computing methodologies** → *Modeling methodologies*; *Model verification and validation*; **Knowledge representation and reasoning**; *Robotic planning*; • **Theory of computation** → *Verification by model checking*; *Logic and verification.*

Additional Key Words and Phrases: Deontic logic, Autonomous vehicles, Model checking, Responsibility-Sensitive Safety, Dominance Act Utilitarianism.

ACM Reference Format:
Colin Shea-Blymyer and Houssam Abbas. 2020. Algorithmic Ethics: Formalization and Verification of Autonomous Vehicle Obligations. 1, 1 (April 2020), 24 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The need for embodied Cyber-Physical Systems (CPS) that are fully autonomous, update their own objectives and interact with us in our daily lives is more obvious today than ever. To cite one example, the Covid-19 pandemic has highlighted the need for nursing robots that can check on patients in high-risk situations, self-driving vehicles that deliver essential goods to people who cannot get them, and companion robots that understand and adapt to different living situations like those of elderly people or incapacitated persons. We refer to these different types of systems as *human-scale CPS* : embodied CPS that interact with humans and their environment, and are perceived as being reasonably intelligent and autonomous. The common thread to all of these applications is that the autonomous CPS is seen as just another agent in our environment, and our interactions with it assume a wide range of social expectations

Authors' address: Colin Shea-Blymyer, sheablyc@oregonstate.edu; Houssam Abbas, houssam.abbas@oregonstate.edu, Oregon State University.

built through our interactions with other humans. Indeed, the success of these systems depends on their respect for these social norms of interaction, and more particularly on the robot's respect for *ethical guidelines* that are as necessary as they are ambiguous. These obligations are distinct from the CPS' mission, which is, for example, to go from A to B without collisions. Obligations place constraints on how the CPS achieves its mission, and might be violated. Safety and performance are no longer sufficient criteria for a successful CPS design: ethical, and more generally social, obligations must be formalized, verified, and where possible, enforced. In this work we tackle the challenges of formalizing ethical obligations in a useful and interpretable way, analyzing the properties of these obligations, and automatically verifying that a system has given obligations.

In the fields of Artificial Intelligence (AI) and Logic, the formalization of ethical and social obligations dates back at least to Mally [28, 30], with most of the focus going towards developing the 'right' logics and simulation-based studies of normative systems. While these logics are interpretable, they lack system models of the agents under obligation. In [19], limited ethical requirements are modeled in the costs and/or constraints of a classical optimal control problem. Precisely defining the costs and constraints that embed ethics into a control problem is challenging, and providing a high-level interpretation of what a particular choice logically entails is generally not possible. E.g. how does the behavior change qualitatively if a slack variable is increased or a weight is decreased? In CPS, The formalization, verification, and enforcement of ethical and social obligations has not been adequately tackled. This paper develops a formal framework and tool for the analysis of the ethical obligations of human-scale CPS, with applications in self-driving cars.

The formal verification and control of CPS safety and performance has relied on *alethic temporal logics*, like Linear Temporal Logic [34], to express behavioral specifications of system models. Alethic logic is the logic of *necessity and possibility*: for example, if p is a predicate, $\Box p$ says that p is true in every accessible world - that is, p is necessary. Possibility is then formalized as $\Diamond p := \neg\Box\neg p$: saying that p is possible is the same as saying that it is not the case that $\neg p$ is necessary. And so on. The best known instantiation of this in Verification is LTL [31], in which an accessible world is a moment in the linear future. Thus $\Box p$ formalizes 'p is true in every future moment', and $\Diamond p$ formalizes 'p is true in some future moment'. It is natural to want to leverage alethic logics and associated tools to formalize and study CPS obligations as well. However, it has been understood for over 70 years that *the logic of obligations is different from that of necessity* [32]: applying alethic logic rules to obligation statements can lead to conclusions that are intuitively paradoxical or undesirable. Consider the following statements:

A. The car *will* eventually change lanes: this is a statement about necessity. It says nothing about whether the car plays an active role in the lane change (e.g., perhaps it will hit a slippery patch), or whether it *should* change lanes.
B. The car *can* change lanes: this is a statement about ability. The car might be able to do something, but does not actually do it.
C. The car *sees to it that* it changes lanes: this is a statement about agency. It tells us that the car ensures that it changes lanes. [1] I.e., it is an active agent in the lane change.
D. The car *ought* to change lanes: this is a statement about obligation. The car, for example, might fail to meet its obligation, either by choice or because it can't change lanes.

These are qualitatively different statements and there is no a priori equivalence between any two of them. The logic we adopt should reflect this: its operators and inference rules should model these aspects *in the logic*, without having to add new atomic propositions for every new concept and situation that occurs to us. Alethic logics like LTL cannot do so.[2]

[1]The 'see to it' phraseology is very common in Logic and we use it in this paper.
[2]Anderson and Kanger attempted a reduction of obligations to alethic logic. See [32, Section 3] for a discussion.

We now give a simple but fundamental example, drawn from [32], illustrating this inability of alethic logic. One might be tempted to formalize obligation using the necessity operator $\Box$: that is, formalize 'The car ought to change lane' by $\Box$`change-lane`. However, in alethic logic, $\Box p \implies p$: if p is necessarily true then it is true. If we use $\Box$ for obligation this reads as **Obligatory** $p \implies p$: this inference is clearly unacceptable because agents sometimes violate their obligations so some obligatory things are not true. This leads to the question of what an agent ought to do when some primary obligations are violated. I.e. the study of statements of the form **Obligatory** $p \wedge \neg p \implies \ldots$. This is not possible if obligation is formalized using $\Box$ in pure alethic logic, since $\Box p \wedge \neg p \implies q$ is trivially true for *any* p and q.

Deontic logic [18] has been developed specifically to reason about obligations, starting with von Wright [42]. It is used in contract law, including software contracts, and is an active area of research in Logic-based AI [26]. There are many flavors of deontic logic [22]. In this paper, we adopt the logic of Dominance Act Utilitarianism (DAU) developed by Horty [23] because it explicitly models all four aspects above: necessity, agency, ability and obligation. We first extend DAU to formalize the obligations of human-scale CPS with complex missions. We then formalize a subset of Intel's Responsibility-Sensitive Safety, or RSS, in DAU [39]. RSS proposes a set of rules to be followed by self-driving cars to avoid collisions. To promote 'naturalistic driving', RSS places an *obligation* to avoid aggressive driving while giving *permission* to drive assertively. Using our DAU formalization of RSS, we demonstrate that RSS allows a car to facilitate an accident in traffic, clearly an undesirable position; this points to the need to further refine the RSS proposal.

We develop the first model-checking algorithm for DAU formulas, to determine whether a system model has a given obligation or not. We implemented the model-checker and present results on a self-driving car controller. An obligation constitutes a constraint on the CPS controller, and can be integrated into the controller's objective; thus designing obligations and checking them is conceptually akin to reward shaping in Reinforcement Learning [43].

When studying an autonomous CPS' obligations, it is also necessary to analyze how these obligations change over time, as a result of the agent's choices [13]. For example if I ought to visit an ill relative today or tomorrow, and I don't visit them today, then it's reasonable to say that tomorrow, my residual obligation is to visit them. It is important that the formal conclusions yielded by the logic match such intuitive conclusions, in order to build trust in human-scale CPS. We prove obligation propagation patterns for obligations expressed in DAU.

Our contributions in this paper are to[3]:

(1) formalize the obligations of RSS in DAU, and highlight the subtle decisions that need to be made when developing a rigorous specification;
(2) derive undesirable consequences of the RSS obligations, pointing to the need for further refinements of RSS;
(3) demonstrate patterns for temporal propagation of obligations in DAU, allowing evaluation of obligations inheritance;
(4) develop a model-checking algorithm of DAU specifications that allows us to establish whether a system has a given obligation or not; and
(5) implement the model-checker and demonstrate its use on a self-driving car from the litterature.

Paper Organization. Section 2 defines DAU. Section 3 gives a first case study: the formalization of a subset of RSS rules in DAU, and some of their logical consequences. Section 4 proves propagation patterns that hold in DAU. Section 5 gives a model-checking algorithm for absolute and conditional DAU statements. Section 6 demonstrates the use of the

[3]A preliminary conference version of this work appeared in [40]. This paper adds the analysis of temporal propagation (item 3 above), improves the RSS formalization significantly and formalizes assertive driving — a model of a social permission (in item 2), adds a model-checking algorithm for conditional obligations, as the original can not find histories that satisfy a condition (in item 4) and implements both model-checkers and demonstrates their use (item 5).

model-checker on a highway driving controller from the literature. Related work is reviewed in Section 7, and Section 8 concludes the paper.

2 DOMINANCE ACT UTILITARIANISM

We adopt the logic of Dominance Act Utilitarianism (DAU) developed by Horty [23] because it explicitly models all four aspects listed in the Introduction: necessity, agency, ability and obligation. It includes a temporal logic as a component so we can describe temporal behaviors essential to system design, and it uses branching time, essential for modeling uncontrollable environments. It has an intrinsic computational structure which makes it appealing for CPS verification and control purposes: the agent's obligations are derived from maximizing utility, so DAU can be viewed as the deontic logic of utility maximization in non-deterministic systems. As such, it gives a *logical interpretation* to the behavior of systems that maximize utility, such as [19]. This section summarizes the main aspects of DAU developed in [23].

Syntax. Let *Agents* be a finite set of agents, which represent, for example, the cars in traffic. A DAU formula is obtained as follows:

$$A := \phi \mid \neg A \mid A \wedge A \mid [\alpha\, cstit\colon A] \mid [\alpha\, dstit\colon A] \mid \odot\, [\alpha\, cstit\colon A] \mid \odot\, ([\alpha\, cstit\colon A]/\phi) \mid XA$$

where $\alpha \in Agents$, $\wedge, \neg$ are the usual boolean connectives, and ϕ is a formula in the logic CTL*. We use CTL* to specify the CPS' mission and to describe states of affairs in the world. We give the informal description of CTL* operators and refer the reader to [17] for formal semantics: the temporal operator $\Box$ means Always (now and in every future moment along this trace), $\Diamond$ means Eventually (now or at some future moment along this trace), and $\mathcal{R}$ means Release: $\phi\mathcal{R}\psi$ means that either ψ always holds, or it does not hold at some future moment and sometime before then ϕ holds. The path quantifier $\forall$ means For all paths, and $\exists$ means There exists a path. The DAU-specific operators informally mean the following: $[\alpha\, cstit\colon A]$ is the agency operator and says that α sees to it, or ensures, that A is true; $[\alpha\, dstit\colon A]$ is a variant on $[\alpha\, cstit\colon A]$; $\odot[\alpha\, cstit\colon A]$ is the obligation modality and says that α ought to ensure that A is true; finally, $\odot([\alpha\, cstit\colon A]/\phi)$ says that under the condition ϕ, α ought to ensure that A is true. The rest of this section gives the formal semantics of these deontic operators.

Branching time. Let *Tree* be a set of *moments* with an irreflexive, transitive ordering relation $<$ such that for any three moments m_1, m_2, m_3 in *Tree*, if $m_1 < m_3$ and $m_2 < m_3$ then either $m_1 < m_2$ or $m_2 < m_1$. There is a unique *root moment* which we denote by 0. A *history* is a maximal linearly ordered set of moments from *Tree*: intuitively, it is a branch of the tree that extends infinitely into the future. Given a moment $m \in Tree$, the set of histories that go through m is $H_m := \{h \mid m \in h\}$. See Fig. 1. We will frequently refer to moment/history pairs m/h, where $m \in Tree$ and $h \in H_m$.

Definition 2.1. [23, Def. 2.2] With *AP* a set of atomic propositions, a *branching time model* is a tuple $\mathcal{M} = (Tree, <, v)$ where *Tree* is a tree of moments with ordering $<$ and v is a function that maps moments m in $\mathcal{M}$ to sets of atomic propositions from 2^{AP}, the set of subsets of *AP*.[4]

In this paper, to retain a uniform satisfaction relation like [23], we will speak of formulas holding or not at an m/h pair and write $\mathcal{M}, m/h \models \phi$, where it is always the case that $h \in H_m$. When the formula is in CTL* there should be no confusion as a CTL* path formula is evaluated along h and a state formula is evaluated at m. Given a DAU statement A,

[4]In the DAU formulation of [23], v maps m/h pairs, rather than moments m, to subsets of AP. This is more general but disagrees with the common usage of atomic propositions in CPS Verification, so we adopt this more classical definition of v. The ideas of this paper are best explained without such (currently) unnecessary generalities.

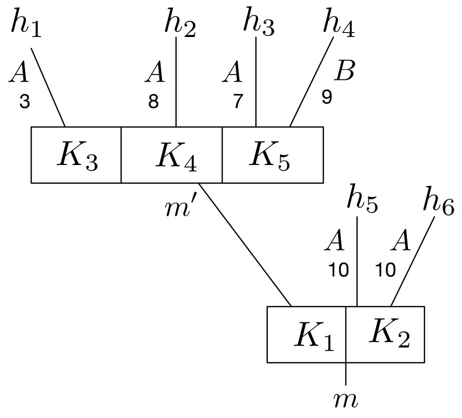


Fig. 1. A utilitarian stit model for an agent α illustrating the main DAU definitions. Moments $m < m'$ with sets of histories $H_m = \{h_1, \ldots, h_6\}$ and $H_{m'} = \{h_1, \ldots, h_4\}$. Each moment is marked with the actions available to α at that moment: $Choice^m_\alpha = \{K_1, K_2\}$ and $Choice^{m'}_\alpha = \{K_3, K_4, K_5\}$. Action $K_2 = \{h_5, h_6\}$ and $K_4 = \{h_2\}$. Each history is marked with the formula(s) that it satisfies at m and with its value $Value(h)$, e.g., m/h_1 satisfies A and has value 3. $m/h_5 \models [\alpha\ cstit\colon A]$ since $Choice^m_\alpha(h_5) = K_2$, and both h_5 and h_6 satisfy A. On the other hand, $m/h_1 \not\models [\alpha\ cstit\colon A]$ since $Choice^m_\alpha(h_1) = K_1 = \{h_1, h_2, h_3, h_4\}$ and h_4 does not satisfy A. $Optimal^m_\alpha = \{K_2\}$ so $m/h_5 \models \odot[\alpha\ cstit\colon A]$. $Optimal^{m'}_\alpha = \{K_4, K_5\}$ and so α has no obligations at m' since there is no formula ϕ s.t. $|\phi|_{m'} \supseteq K_4 \cup K_5$ (See Def. 2.4). Finally, $m/h_5 \models [\alpha\ dstit\colon A]$ because $K_2 \subset |A|_m$ and $H_m \neq |A|_m = \{h_1, h_2, h_3, h_5, h_6\}$.

the *proposition* it expresses at moment m is the set of histories where it holds starting at m

$$|A|^{\mathcal{M}}_m := \{h \in H_m \mid \mathcal{M}, m/h \models A\} \tag{1}$$

Where there's no risk of ambiguity, we drop $\mathcal{M}$ from the notation, writing $|A|_m$, $m/h \models A$, etc.

Choice. Consider an agent $\alpha \in Agents$. Formally, at m, an action K is a subset of H_m: this is the subset of histories that are still realizable after taking the action. At every moment m, α is faced with a choice of actions which we denote by $Choice^m_\alpha$. So $Choice^m_\alpha \subset 2^{H_m}$. See actions in Fig. 1. $Choice^m_\alpha$ must obey certain constraints given in the Supplementary material. In what follows, $Choice^m_\alpha$ is assumed finite for every α and m.

Agency. Agency is defined via the 'Chellas sees to it' operator *cstit*, named after Brian Chellas [16]. Intuitively, an agent *sees to it*, or ensures, that A holds at m/h if it takes an action K s.t., whatever other history h' could've resulted from K, A is true at m/h' as well. I.e., the non-determinism does not prevent α from guaranteeing A.

Definition 2.2 (Chellas cstit). [23, Def. 2.7] With agent α and DAU statement A, let $Choice^m_\alpha(h)$ be the unique action that contains h. Then

$$\mathcal{M}, m/h \models [\alpha\ cstit : A] \text{ iff } Choice^m_\alpha(h) \subseteq |A|^{\mathcal{M}}_m$$

If $K \subseteq |A|_m$ we say K *guarantees* A. See Fig. 1. A *deliberative stit* operator is also defined, which captures the notion that an agent can only truly be said to do something if it also has the choice of not doing it. See Fig. 1.

Definition 2.3 (Deliberative stit). [23, Def. 2.8] With agent α and DAU statement A,

$$\mathcal{M}, m/h \models [\alpha\ dstit\colon A] \text{ iff } Choice^m_\alpha(h) \subseteq |A|^{\mathcal{M}}_m \text{ and } |A|^{\mathcal{M}}_m \neq H_m$$

Operators *cstit* and *dstit* are not interchangeable and fulfill complementary roles.

Optimal actions. To speak of an agent's obligations, we will need to speak of 'optimal actions', those actions that bring about an ideal state of affairs. Let $Value : H_0 \to \mathbb{R}$ be a *value function* that maps histories of $\mathcal{M}$ to *utility values* from the real line $\mathbb{R}$. This value represents the utility associated by all the agents to this common history. Given two sets of histories Z and Y , we order them as

$$Z \leq Y \text{ iff } Value(h) \leq Value(h') \quad \forall\, h \in Z, h' \in Y \tag{2}$$

Let $State^m_\alpha := Choice^m_{Agents\setminus\{\alpha\}}$ be the set of *background states* against which α's decisions are to be evaluated. These are the choices of action available to other agents. Given two actions K, K' in $Choice^m_\alpha$, $K \preceq K'$ iff $K \cap S \leq K' \cap S$ for all $S \in State^m_\alpha$. That is, K' dominates K iff it is preferable to it regardless of what the other agents do (known as *sure-thing reasoning*). Strict inequalities are naturally defined. Optimal actions are given by

$$Optimal^m_\alpha := \{K \in Choice^m_\alpha \mid \nexists K' \in Choice^m_\alpha \text{ s.t. } K \prec K'\} \tag{3}$$

$Optimal^m_\alpha$ is non-empty in models with finite $Choice^m_\alpha$ [23, Thm. 4.10].

Dominance Ought. We are now ready to define Ought statements, i.e., obligations. Intuitively we will want to say that at moment m, agent α *ought to see to it* that A iff A is a necessary condition of all the histories considered ideal at moment m. This is formalized in the following *dominance Ought operator*, which is pronounced "α ought to see to it that A holds".

Definition 2.4 (Dominance Ought). With α an agent and A an obligation in a model $\mathcal{M}$,

$$\mathcal{M}, m/h \models \odot[\alpha\, cstit\colon A] \text{ iff } K \subseteq |A|^{\mathcal{M}}_m \text{ for all } K \in Optimal^m_\alpha \tag{4}$$

See Fig. 1 for examples. The dominance ought satisfies a number of intuitive logical properties; we refer the reader to [23, Ch. 4]. The dual of the Ought is (weak, a.k.a. negative) Permission:

$$\mathsf{P}[\alpha\, cstit\colon A] := \neg \odot [\alpha\, cstit\colon \neg A]$$

The intuitive meaning of permission is that α can ensure A without violating any obligations. Moreover, having a permission does not imply that one actually sees to it that A is true. This is quite different from $\Diamond A$, which simply says that A actually happens, and from $\exists[\alpha\, cstit\colon A]$ which says that α can ensure A, neither of which refers to obligations.

Conditional obligation. It is often necessary to say that an obligation is imposed only under certain conditions. Let X be a proposition, i.e. $X = |\phi|_m$ for some ϕ. The choice of actions available to α at m under the condition that X holds is defined as $Choice^m_\alpha/X := \{K \in Choice^m_\alpha \mid K \cap X \neq \emptyset\}$. This is the right definition because non-determinism might make it impossible to have $K \subseteq X$ (i.e., an action that guarantees X), but future actions might still ensure the finally realized history will satisfy X. Thus in Fig. 1 $Choice^m_\alpha/B = \{K_1\}$. Conditional dominance is then defined by comparing only histories that satisfy ϕ: for two actions K, K' from $Choice^m_\alpha$, $K \preceq_X K'$ iff $K \cap S \cap X \leq K' \cap S \cap X$ for all $S \in State^m_\alpha$. The *conditionally optimal actions* are then

$$Optimal^m_\alpha/X := \{K \in Choice^m_\alpha/X \mid \nexists K' \in Choice^m_\alpha/X \text{ s.t. } K \prec_X K'\} \tag{5}$$

Finally, where A is an obligation and ϕ a formula in the underlying temporal logic, the conditional Ought is defined by

$$\mathcal{M}, m/h \models \odot([\alpha\, cstit\colon A]/\phi) \text{ iff } K \subseteq |A|^{\mathcal{M}}_m \;\forall K \in Optimal^m_\alpha/|\phi|^{\mathcal{M}}_m. \tag{6}$$

We note that conditional obligation is *not* the same as $\phi \implies \odot[\alpha\, cstit : \phi]$. Conditional obligation only compares ϕ-satisfying histories, while this latter formula still compares all histories.

Terminology abuse. In what follows, histories that belong to optimal actions will be called optimal.

3 CASE STUDY IN MODELING: RESPONSIBILITY-SENSITIVE SAFETY FOR SELF-DRIVING CARS

Responsibility-Sensitive Safety, or RSS, is a proposal put forth by Intel's Mobileye division [39]. It proposes rules or requirements that, if followed by all cars in traffic, would lead to zero accidents. RSS attempts to promote a natural way of driving by drawing the line between acceptable assertive driving, and unacceptable aggressive driving. We consider these notions, of assertive vs. aggressive driving, to be fundamentally *social* because they refer to what a particular society accepts. Thus we may say that RSS places an *obligation* to avoid aggressive driving while giving *permission* to drive assertively. The RSS proposal is expressed in the language of continuous-time dynamical systems and ordinary differential equations, but the rules to be followed are not formalized logically, so it is not possible to reason about them or derive their logical consequences. This work complements the dynamical equations-based presentation of RSS in [39] with a deontic logic formalism. We have three objectives in doing so: demonstrating the usefulness of DAU in a real use case; highlighting the ambiguities implicit in such proposals, which would go unnoticed without formalization; and automating the checking of logical consistency and deriving of conclusions. We first present the RSS rules in natural language (Section 3.1), then their formalization (Section 3.2), and finally we analyze the rules' logical consequences. Three important points must be made:

(A) The formalization does not depend on the dynamical equations that govern the cars because we wish our conclusions to be independent of these lower-level concerns. This is consistent with the standard AV control architecture where a logical planner decides what to do next ('change lanes' or 'turn right') and a lower-level motion planner executes these decisions. Our logical analysis concerns the logical planner.

(B) We are not trying to formalize general traffic laws [38] or driving scenarios, which is outside the scope of this paper. We are only formalizing the RSS rules.

(C) Every formalization, in any logic, can always be refined. We are not aiming for the most detailed formalization; we aim for a useful formalization.

3.1 The RSS rules

The rules for Responsibility-Sensitive Safety are [39]:

RSS1. Do not hit someone from behind.

RSS2. Do not cut-in (to a neighboring lane) recklessly.

RSS3. Right-of-way is given, not taken.

RSS4. Be careful of areas with limited visibility.

RSS5. If you can avoid an accident without causing another one, you must do it.

RSS6. To change lanes, you do not have wait forever for a perfect gap: i.e., you do not have to wait for a gap large enough to get into even when the other car, already in the lane, maintains its current motion.

RSS6 is derived directly from the following in [39, Section 3]: "the interpretation [of the duty-of-care law] should lead to [...] an agile driving policy rather than an overly-defensive driving which inevitably would confuse other human drivers and will block traffic [...]. As an example of a valid, <u>but not useful</u>, interpretation is to assume that in order to be "careful" our actions should not affect other road users. Meaning, if we want to change lane we should find a gap large enough such that if other road users continue their own motion uninterrupted we could still squeeze-in without a collision. Clearly, for most societies this interpretation is over-cautious and will lead the AV to block traffic and be non-useful." Note that, consistently with points (A)-(C) above, this is stated without any reference to dynamics or specific scenarios. The RSS authors are concerned that overly cautious driving might lead to unnatural traffic, so RSS aims to allow cars to move a bit assertively, and defines correct reactions to that.

Note finally that RSS4 is explicitly formulated in terms of obligations and ability. However, we will not study RSS4 and RSS5 as they are currently too vague for formalization.

3.2 Formalization of RSS Rules

Formalizing RSS1. Let ϕ denote 'collision with car ahead of me'. A plausible formalization of RSS1 is then

$$RSS1. \ \odot [\alpha\, cstit\colon \neg\phi]$$

That is, α ought to see to it that there is no collision with a car ahead of it. A positive aspect of this formalization is that if at some m, a rear-end collision is inevitable, then $RSS1$ ceases to hold: $\forall\phi \implies \neg \odot [\alpha\, cstit\colon \neg\phi]$. This provides an automatic and interpretable update of control objectives. In a deployed system, an automatic proof engine could update which obligations hold and which don't, based on the current situation [3]. An alternative formalization is

$$RSS1r. \ \odot [\alpha\, cstit\colon \neg[\alpha\, dstit\colon \phi]]$$

This says that α should see to it that it does not deliberately ensure an accident ϕ. This form of obligation is called *refraining* [24]: α *refrains* from hitting anyone from behind. If a rear-end collision is inevitable at some m, then $RSS1r$ still holds (unlike $RSS1$) *and is trivially satisfied.* This might be computationally cheaper than having to use a proof engine to tell us that the obligation no longer holds.

In the general case, some actions at m guarantee a collision, some guarantee no collision, and the rest don't guarantee either: the future could evolve either way. If we are interested in guaranteeing no collision over a long horizon, then, because of non-determinism, it is unlikely that any action in the present moment can guarantee that. In such a case $RSS1$ will be violated repeatedly in a rather trivial way; on the other hand, $RSS1r$ is more permissive, since it can be met by taking any optimal action that allows the *possibility* of no collision over the horizon. A lower-level controller, running at a higher rate, could then ensure freedom from collision forever.

Formalizing RSS2. Define formulas, ψ : a non-reckless cut-in, and ψ_r: a reckless cut-in. Then RSS2 is formalizable as

$$RSS2. \odot [\alpha\, cstit\colon \Box(\psi \vee \psi_r \implies \neg\psi_r)].$$

That is, α should see to it that always, if a cut-in happens, then it is a non-reckless cut-in.

Formalizing RSS3. Formalizing this rule requires some care. First, note that RSS3 should probably be amended to say that 'Right-of-way is given, not taken, *and some car is given the right-of-way*' - otherwise, traffic comes to a standstill. We will first focus on formalizing the prohibition (nobody should take the right-of-way), then we will formalize the positive obligation (somebody must be given it).

Let $Agents = \{\alpha, \beta, \gamma, \ldots\}$ be a finite set of agents. Define the atomic propositions $GROW^{\alpha}_{\beta}$: β gives right-of-way to α and p_α: α proceeds/drives through the conflict region. Then $TROW_\alpha := p_\alpha \wedge \neg(GROW^{\alpha}_{\beta} \wedge GROW^{\alpha}_{\gamma} \wedge \ldots)$ formalizes taking the right-of-way: α proceeds without being given the right-of-way by everybody. We could now express the prohibition in RSS3: every α ought to see to it that it does not take the right-of-way:

$$RSS3prohib0. \quad \bigwedge_{\alpha\in Agents} \odot[\alpha\, cstit\colon \neg TROW_\alpha] \tag{7}$$

The difficulty with this formulation is that it could lead to α being obliged to force *everybody else* to give it the right-of-way - something over which, a priori, it has no control. To see this, we need the following.

Proposition 3.1. *Given obligations A and B, $\odot[\alpha\, cstit\colon A\vee B]\wedge(\forall\neg A) \implies \odot[\alpha\, cstit\colon B]$*

In words, if α ought to ensure A or B at m/h, but every available history violates A, then its obligation is effectively to ensure B.

Proof. Assume that $m/h \models \odot[\alpha\, cstit\colon A\vee B]\wedge\forall\neg A$. By definition of the dominance ought, for all $K\in Optimal^m_\alpha, K\subseteq |A\vee B|_m$. And by definition of $|A|_m$ (Eq. 1), $|A\vee B|_m = |A|_m\cup|B|_m$. We also have that $m/h\models\forall\neg A$, i.e., $m/h'\models\neg A$ for all $h'\in H_m$; thus $|A|_m=\emptyset$. Therefore $\forall K\in Optimal^m_\alpha, K\subseteq|B|_m$, which is the definition of $m/h\models\odot[\alpha\, cstit\colon B]$. □

Applied to Eq. (7) with $A=\neg p_\alpha$ and $B=\wedge_{\beta\neq\alpha} GROW^\alpha_\beta$, Thm. 3.1 says that if α is in a situation where it has no choice but to proceed (e.g. as a result of slippage on a wet road, say), then its obligation is to see to it that everybody else gives it the right-of-way, which is unreasonable.

Instead, we adopt a more passive attitude: every agent sees to it that if they are not given the right-of-way, then they do not pass. Letting atomic proposition g_α denote that right-of-way is Granted to α,

$$RSS3prohib. \quad \bigwedge_{\alpha\in Agents} \odot[\alpha\, cstit\colon \Box(\neg g_\alpha \implies \neg p_\alpha)] \tag{8}$$

The positive obligation, that somebody must be given the right-of-way, seems to be a *group obligation*: the *group* must give right-of-way to one of its members. Group obligations are formally defined in [23, Ch. 6]. Then we formalize

$$RSS3pos. \ \odot\, [Agents\ \ cstit\colon \vee_{\alpha\in Agents}\, g_\alpha] \tag{9}$$

This says the group *Agents* has an obligation to give right-of-way to someone, and the only choice is in *who* gets it. Finally, we formalize *RSS3* as the conjunction of *RSS3prohib* and *RSS3pos*.

Formalizing assertiveness and RSS6. This rule says that if the car wants to change lanes, it shouldn't have to wait forever for the perfect gap (otherwise, traffic is stalled). It is one way in which RSS attempts to promote 'assertive driving', a style of driving that tries to obtain right-of-way in a 'polite' way. The key difficulty, of course, is to distinguish between assertive driving, which is acceptable, and aggressive driving, which is not. Deontic logic can help in that regard. *We model assertiveness as a permission to not drive conservatively or defensively.* That is, if χ is a formula that describes conservative driving behavior in a particular context Ω, then driving assertively is the conditional permission

$$\mathrm{P}([\alpha\ cstit\colon \neg[\alpha\, dstit\colon \chi]]/\Omega) \tag{10}$$

This is a *permission*: it does not constitute an obligation to drive assertively. Depending on its reward structure, the agent might choose to drive conservatively after all. Importantly, Eq. (10) states that the agent can drive assertively without violating any obligations it does have.

For RSS6, conservative driving consists in waiting for the perfect gap before passing, that is, waiting until the other car, already in the lane, gives α the right-of-way. Thus we may take $\chi = g_\alpha\mathcal{R}\neg p_\alpha$, where, recall, p_α means 'α proceeds through the conflict region' and g_α means 'α is granted the right-of-way'. Finally, with w_α meaning 'α wants to change lanes' we have

$$RSS6. \quad \mathsf{P}([\alpha\ cstit\colon \neg[\alpha\ dstit\colon g_\alpha \mathcal{R} \neg p_\alpha]]/w_\alpha) \tag{11}$$

3.3 Application: undesirable consequence of RSS star-calculations

One of the main tenets of RSS is that an Autonomous Vehicle (AV) is only responsible for avoiding potential accidents between itself and other cars (so-called 'star calculations'); interactions between 2 other cars are not its concern [39, Remarks 1 and 8]. Yet everyday driving experience makes clear that our actions can be faulted for at least *facilitating* an accident: e.g., by repeated braking, I may cause the car behind me to do the same, leading the car behind *it* to rear-end it. Or I might make a sudden lane change over two lanes, causing the car in the lane next to me to over-react when I speed past it, and collide with someone else. We now show how this intuition is automatically captured by the DAU logic, and that RSS star-calculations lead to undesirable behavior of the AV.

Let $\phi \in \text{CTL}^*$ denote a formula expressing "Accident between two other cars", and the accident is such that α can facilitate it as in the above 2 examples. Then $[\alpha\ dstit : \phi]$ says that α (deliberately) sees to it that the accident happens even though it could avoid doing so; given what we assumed about this accident, this means α facilitates the accident. Then $[\alpha\ dstit\colon \neg[\alpha\ dstit\colon \phi]]$ expresses that α sees to it that it does *not* facilitate the accident: this is a form of refraining. Finally, $[\alpha\ dstit\colon \neg[\alpha\ dstit\colon \neg[\alpha\ dstit\colon \phi]]]$ says that α refrains from refraining, that is, α does not refrain from facilitating the accident (even though it could). The RSS position is that it is OK for α to refrain from refraining [39, Remarks 1 and 8]. However, refraining from refraining is the same as doing. Formally [23, 2.3.3.]

$$[\alpha\ dstit : \neg[\alpha\ dstit : \neg[\alpha\ dstit : \phi]]] \equiv [\alpha\ dstit\colon \phi]$$

This matches our intuition: to not refrain from facilitating an accident even though one could (left-hand side in previous equation) is the same as facilitating it (right-hand side). In other words, under this formalization, the RSS position is tantamount to allowing AVs to facilitate accidents between others - clearly, an undesirable conclusion. This aspect of RSS, therefore, needs refinement to take into account longer-range interactions between traffic participants.

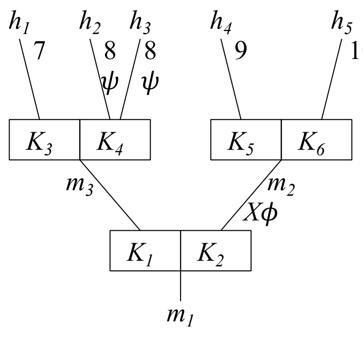
4 OBLIGATION PROPAGATION

Obligations vary over time: the obligation at moment m is the set of necessary conditions (formulas in the tense logic) satisfied by all histories optimal at m, and the set of optimal histories can change from moment to moment. There is thus a need to understand how obligations change over time: for example if the agent does not act optimally at m, does the obligation disappear at the next moment? Or does it persist, perhaps in a modified form? The formal study of obligation propagation is also a way to interpret the temporal evolution of utility-maximizing controllers: as the controller (and the environment) act, obligations change, placing new constraints on the controller.

The following examples show that these questions must be studied formally, since intuition usually fails us. Consider the following tentative propagation pattern, in which ϕ is a CTL* formula:

$$\odot\, [\alpha\ cstit\colon X\phi] \implies X \odot [\alpha\ cstit\colon \phi] \tag{12}$$

This says that an obligation now to ensure that ϕ holds at the next moment implies an obligation at the next moment to ensure that ϕ holds, which sounds plausible. However, it is not valid in DAU. Fig. 2a gives a counter-example: K_2 is optimal at m_1 so $m_1/h_1 \models \odot[\alpha\ cstit\colon X\phi]$; however m_3 is the next moment along h_1 and $m_3/h_1 \not\models \odot[\alpha\ cstit\colon \phi]$, so $m_1/h_1 \not\models X \odot [\alpha\ cstit\colon \phi]$ and Eq. (12) is not valid.



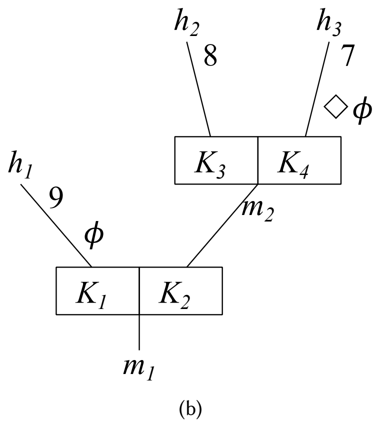


Fig. 2. Counter-examples to tentative obligation propagation patterns. (a) Pattern in Eq. (12); (b) Pattern in Eq. (13)

As a second example, the following tentative pattern says that if the agent has an obligation to ensure that ϕ eventually holds, does not do so now, *but it is still possible to do so at the next moment*, then at the next moment the agent still has an obligation to ensure eventually ϕ:

$$\odot[\alpha\, cstit\colon \Diamond\phi] \wedge \neg[\alpha\, cstit\colon \Diamond\phi] \wedge X\exists[\alpha\, cstit\colon \Diamond\phi] \implies X\odot[\alpha\, cstit\colon \Diamond\phi] \tag{13}$$

Fig. 2b shows a counter-example to this second pattern: we have $m_1/h_2 \models \odot[\alpha\, cstit\colon \Diamond\phi]$, and that $m_1/h_2 \models X\exists[\alpha\, cstit\colon \Diamond\phi]$. If K_2 will be taken, then $m_2/h_2 \not\models \odot[\alpha\, cstit\colon \Diamond\phi]$ because K_3 is optimal at m_2, so Eq. (13) is also invalid.

Both counter-examples exploited the fact that along the m/h pair where the left-hand side is evaluated, the agent acts non-optimally. This suggests that to derive valid temporal propagation patterns, we must assume the agent is acting optimally. So we define the distinguished atomic proposition a^* for this purpose:

$$m/h \models \mathsf{a}^* \text{ iff } Choice^m_\alpha(h) \in Optimal^m_\alpha$$

The following pattern *is* valid in DAU. Let $m^+(h)$ be the moment that follows m in h; e.g., $m_1^+(h_1) = m_3$ in Fig. 2a.

$$\odot[\alpha\, cstit\colon X\phi] \wedge \mathsf{a}^* \implies X\odot[\alpha\, cstit\colon \phi] \tag{14}$$

Proof. The pair m/h satisfies the left-hand side iff $K \subseteq |X\phi|_m$ for all optimal K at m. By a^*, we have that $Choice^m_\alpha(h)$ is optimal, thus $m/h \models X\phi$, which implies that $m^+(h)/h \models \phi$, which is the definition of $m/h \models X\odot[\alpha\, cstit\colon \phi]$. □

Acting optimally is not always enough however. The following valid pattern says that if α ought to see to it that $\Diamond\phi$, acts optimally, but it is impossible to satisfy ϕ now, then at the next moment α still ought to see to it that $\Diamond\phi$. Here, $\forall\neg\phi$ is necessary in the antecedent: the implication fails trivially without it.

$$\odot[\alpha\, cstit\colon \Diamond\phi] \wedge \mathsf{a}^* \wedge \forall\neg\phi \implies X\odot[\alpha\, cstit\colon \Diamond\phi] \tag{15}$$

Finally we present a pattern of obligation propagation which does not require optimal behavior, but which is only satisfied in certain models.

Lemma 4.1. *With ϕ, ψ CTL* formulas, let $\mathcal{M}$ be a stit model which satisfies the following constraint at every moment m: for all actions $K \in Choice^m_\alpha$ s.t. $K \subseteq |\neg\phi|_m$ and which contain a history h s.t. $m^+(h) \models \exists[\alpha\, cstit\colon \psi]$, it holds that all* *optimal actions in* $Optimal_\alpha^{m^+(h)}$ *guarantee* ψ*. Then* in such a model*, the following is satisfied at every index* m/h*.*

$$\odot\,[\alpha\, cstit\colon \phi \vee X\psi] \wedge [\alpha\, cstit\colon \neg\phi] \wedge X\exists[\alpha\, cstit\colon \psi] \implies X \odot [\alpha\, cstit\colon \psi] \tag{16}$$

This says that if α has an obligation to ensure $\phi \vee X\psi$, guarantees $\neg\phi$ now, but next it is still possible to guarantee ψ, then the next obligation is to guarantee ψ.

Proof. Let m/h be an index in $\mathcal{M}$ at which the DAU formula (16) is evaluated. Let K_h be the action to which h belongs, and for brevity, write $m' = m^+(h)$.

Case 1: $h \in \cup_{K \in Optimal_\alpha^m} K$. Then $K_h \subseteq |\phi \vee X\psi|_m = |\phi|_m \cup |X\psi|_m$. By hypothesis, $K_h \subseteq |\neg\phi|_m$ also so $K_h \subseteq |X\psi|_m \setminus |\neg\phi|_m$. By construction, $Optimal_\alpha^{m'} \subseteq H_{m'} \subseteq K_h$ so for every $K^* \in Optimal_\alpha^{m'}$ and every $h' \in K^*$, $m'/h' \models \psi$, which is the definition of $m/h \models X \odot [\alpha\, cstit\colon \psi]$.

Case 2: $h \notin \cup_{K \in Optimal_\alpha^m} K$. From the formula antecedent, we have that $K_h \subseteq |\neg\phi|_m$ and that $m'/h \models \exists[\alpha\, cstit\colon \psi]$. Therefore the model constraint yields that all optimal actions at m' guarantee ψ, which is the definition of $m/h \models X \odot [\alpha\, cstit\colon \psi]$. □

Finally, the proof also establishes the following pattern.

Proposition 4.2. *The following is valid (i.e., satisfied in all models) in DAU:*

$$\odot[\alpha\, cstit\colon \phi \vee X\psi] \wedge [\alpha\, cstit\colon \neg\phi] \wedge \mathsf{a}^* \implies X \odot [\alpha\, cstit\colon \psi]$$

5 MODEL CHECKING DAU

The expressive power of DAU makes the logic a useful tool in the hands of a system designer. The system designer can use DAU to specify the obligations the system ought to have. While DAU derives obligations from stit trees, control engineers often model agents as some kind of automata. How then can we verify that the controller has the obligations the system designer has specified? Note that having an obligation is not the same as meeting that obligation: the obligation is a constraint that might or might not be met. This section's algorithms verify that a system *has* a given obligation, i.e. that it has the given constraints on its behavior.

We can ensure that an agent has an obligation by framing the question as a model checking problem. In this section we cast agents as *stit automata*, and introduce novel algorithms to perform model checking for obligations. All proofs not given here can be found in the supplementary material.

5.1 Stit Automata

For a set S, let S^ω denote the set of infinite sequences $(a_i)_{i\in\mathbb{N}}$ with $a_i \in S$.

Definition 5.1 (Stit automaton). Let AP be a finite set of atomic propositions. A *stit automaton* T is a tuple $T = (Q, q_I, \mathcal{K}, F, \Delta, L, w, \lambda)$, where Q is a finite non-empty set of states, q_I is the initial state, $\mathcal{K}$ is a finite non-empty set of actions, $F \subset Q$ is a set of final states, $\Delta \subset Q \times \mathcal{K} \times Q$ is a finite transition relation such that if (q, K, q') and (q, K', q') are in Δ then $K = K'$, $L : Q \to 2^{AP}$ is a labeling function, $w : \Delta \to \mathbb{R}$ is a weight function, and $\lambda : \mathbb{R}^\omega \to \mathbb{R}$ is an accumulation function.

When dealing with multiple automata, we will sometimes write $T.q_I$, $T.\lambda$, etc, to clarify which automaton is involved. Note that T is a type of non-deterministic weighted automaton. Its unweighted counterpart T^u is a classical transition system, thus for a CTL* formula ϕ, we could model-check whether $T^u \models \phi$. Denote by $\Delta(q)$ the set of outgoing

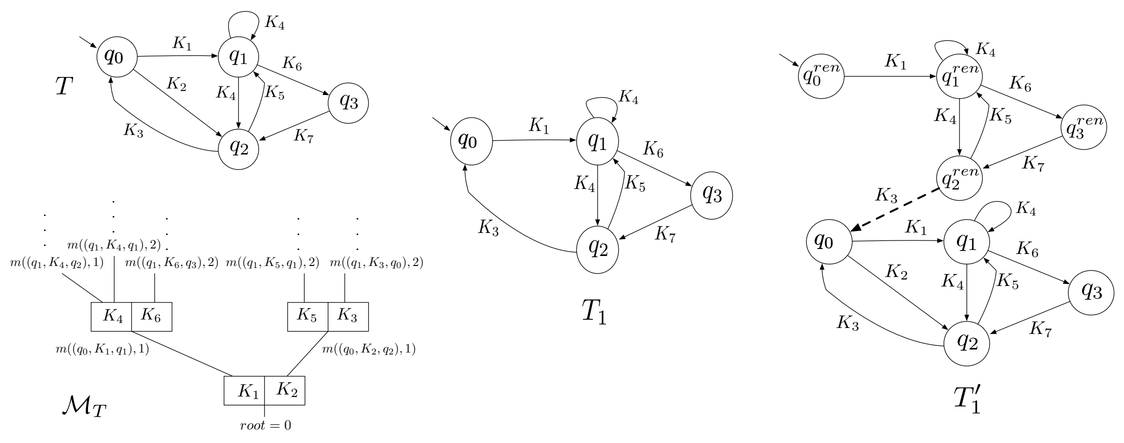


Fig. 3. Left: a stit model generated by executing the stit automaton T (transition weights not shown). Center and right: Automata T_n and T_n' used in Algorithm 1. T_1 only has K_1 as first action, and T_1' is obtained by re-naming states of T_1 and adding a copy of T_1 to it. Executions of T_1' are simply the execution of T that start with K_1.

transitions from q ($\Delta(q) = \{(q, K, q') \in \Delta\}$), by $\mathcal{K}(q) = \{K \in \mathcal{K} \mid \exists (q, K, q') \in \Delta\}$ the set of actions available at q. Examples of λ include the functions min / max, discounted sum and long-run average:

$$\min(\xi) \quad = \quad \min_i w(\xi[i]) \tag{17}$$

$$\mathsf{DiscSum}(\xi) \quad = \quad \sum_{i=0}^{\infty} \gamma^i \cdot w(\xi[i]) \ , 0 < \gamma < 1 \tag{18}$$

$$\mathsf{liminfAvg}(\xi) \quad = \quad \liminf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} w(\xi[i]) \tag{19}$$

Definition 5.2 (Execution). Let T be a stit automaton and q_0 a state in Q. A *q_0-rooted execution ξ* of T is a sequence of transitions of the form $\xi = (q_0, K_0, q_1)(q_1, K_1, q_2) \ldots \in \Delta^\omega$. The corresponding sequence of actions $K_0, K_1, \ldots \in \mathcal{K}^\omega$ is called a *tactic*. The value of execution $\xi = \xi[0]\xi[1]\xi[2]\ldots$, where $\xi[i] \in \Delta$, is defined to be $\lambda(w(\xi[0])w(\xi[1])w(\xi[2])\ldots)$, and abbreviated $\lambda(\xi)$.

Because of non-determinism, a tactic can produce multiple executions. A set of agents is modeled by the product of all individual stit automata, which is itself a stit automaton. (When taking the product, we must define how weights are combined and how to construct the product's accumulation function, which are application-specific considerations.) Therefore the rest of this section applies to stit automata, whether they model one or multiple agents. We will continue to refer to one agent α for simplicity.

From stit automata to stit models. An automaton T, along with a state $q_0 \in Q$, induce a stit model $\mathcal{M}_{T,q_0}$ in the natural way, which we now describe somewhat informally: state q_0 maps to the root moment 0 of $\mathcal{M}_{T,q_0}$. From q_0, T has a choice of actions $\mathcal{K}(q_0)$, which map to the actions available at 0 in $\mathcal{M}_{T,q_0}$. Each action K in $\mathcal{K}(q_0)$ non-deterministically causes one or more transitions, each of which maps to a moment in $\mathcal{M}_{T,q_0}$; all transitions caused by a given K map to moments in histories that originate in the same action K in $\mathcal{M}_{T,q_0}$. And so on from each next state. See Fig. 3 for an example. We let $\eta : \Delta \to \mathit{Tree}$ denote the map from transitions to moments, and lift it to executions in the natural way, i.e., $\eta(\xi) := \eta(\xi[0])\eta(\xi[1])\dots$ By construction, $\eta(\xi)$ is a history in $\mathcal{M}_{T,q_0}$. Its utility $Value(\eta(\xi))$ is the λ-value of the generating execution, i.e., $\lambda(\xi)$. The atoms labeling $\eta((q,K,q'))$ are $v(\eta((q,K,q'))) := L(q')$. The formal construction and proof of the following proposition can be found in the supplementary material.

Proposition 5.3. *The structure $\mathcal{M}_{T,q_0}$ is a utilitarian stit model with finite $Choice_\alpha^m$ for every agent α and moment m.*

Model-checking determines whether a stit automaton, at a given state, satisfies an Ought statement.

Definition 5.4. Given an automaton T, one of its states q, the induced model $\mathcal{M}_{T,q}$ and an obligation A, we say that T, q satisfies $\odot[\alpha\, cstit\colon A]$, written $T, q \models \odot[\alpha\, cstit\colon A]$, iff $\mathcal{M}_{T,q}, 0/h \models \odot[\alpha\, cstit\colon A]$ for an arbitrary history $h \in H_0$.

The history h is arbitrary since the truth of $\odot[\alpha\, cstit\colon A]$ does not depend on the history but only on the moment.

The construction of $\mathcal{M}_{T,q}$ roots all histories at moment 0. However, what if the automaton can only reach state q after i time steps? Then a priori, it might be that whether an Ought holds at q depends on i, because the accumulation function λ can be time-dependent. The following shows that for certain accumulation functions important in practice, the choice of root moment does not matter.

Proposition 5.5. *Given a stit automaton T and an obligation A, let q, q' be states of T s.t. q is reachable from q' in i transitions along an execution ξ. Let h be an arbitrary history of $\mathcal{M}_{T,q}$, let $h' = \eta(\xi)$ be the history that connects $\eta(\xi[0])$ to $\eta(\xi[i-1])$ in $\mathcal{M}_{T,q'}$, and let $m' = \eta(\xi[i-1])$. Then, if λ is discounted sum or long-run average, $\mathcal{M}_{T,q}, 0/h \models \odot[\alpha\, cstit\colon A]$ iff $\mathcal{M}_{T,q'}, m'/h' \models \odot[\alpha\, cstit\colon A]$*

Proof. For clarity, we write $\mathcal{M} = \mathcal{M}_{T,q}$ and $\mathcal{M}' = \mathcal{M}_{T,q'}$, and write $\mathcal{M}.()$ vs $\mathcal{M}'.()$ to disambiguate something in $\mathcal{M}$ vs something in $\mathcal{M}'$. We will show that the trees rooted at $0/h$ in $\mathcal{M}$ and i/h' in $\mathcal{M}'$ have the same structure and that the value ordering of their histories is the same in both models. This implies that the same Oughts hold at both.

The histories of $\mathcal{M}$ are images, under η, of executions that start at q. Because transition $\xi[i-1]$ ends in q, the histories of $\mathcal{M}'$ rooted at $m' = \eta(\xi[i-1])$ are also images of executions that start at q. Therefore, $\mathcal{M}'.H_{m'}$ is identical to $\mathcal{M}.H_0$. In particular they satisfy the same set of CTL* formulas. We refer to this common set of histories as H^*.

Take two arbitrary $h_1, h_2 \in H^*$ and their pre-images ξ_1, ξ_2 by η. By construction, $\xi[k+i] = \xi_1[k] = \xi_2[k]$, $k \geq 0$, and the concatenation $f_j := h'[0]\dots h'[i-1]h_j[0]h_j[1]\dots$ is a 0-rooted history in $\mathcal{M}'$, $j = 1, 2$. If $\lambda = \text{DiscSum}$ then $\mathcal{M}.Value(h_j) = \sum_{k\geq 0}\gamma^k w(\xi_j[k])$, while $\mathcal{M}'.Value(f_j) = \sum_{k=0}^{k=i-1}\gamma^k w(\xi[k]) + \sum_{k\geq 0}\gamma^{i+k}w(\xi_j[k])$. Thus

$$\mathcal{M}.Value(h_1) \leq \mathcal{M}.Value(h_2) \text{ iff } \mathcal{M}'.Value(f_1) \leq \mathcal{M}'.Value(f_2)$$

Thus the histories in H^* are identically ranked in both models, which implies that optimal actions are the same. This, combined with the fact that they satisfy the same formulas, yields the desired conclusion.

Similarly, if λ is liminfAvg, then for $j = 1, 2$,

$$\mathcal{M}'.Value(f_j) = \liminf_{n\to\infty}\frac{1}{n}\left[\sum_{0\leq k\leq i-1} w(\xi[k]) + \sum_{k\geq 0} w(\xi_j[k])\right] = \liminf_{n\to\infty}\frac{1}{n}\sum_{k\geq 0} w(\xi_j[k]) = \mathcal{M}.Value(h_j)$$

So histories of H^* are identically ranked by liminfAvg in both models, yielding the desired conclusion. □

5.2 Model Checking of Unconditional Obligations

The problem of *cstit model checking* is: given a stit automaton T that models an agent α, a state $q \in T.Q$, and a formula A which is either a CTL* formula, or a statement of the form $[\alpha\, dstit\colon \phi]$ or $\neg[\alpha\, dstit\colon \phi]$ where ϕ is a CTL* formula, determine whether $\mathcal{M}_{T,q}, 0/h \models \odot[\alpha\, cstit\colon A]$ for some arbitrary $h \in H_0$.

We restrict the algorithm to statements of the above forms for conciseness of the presentation; DAU formulas with additional nesting levels can be handled by extending the algorithms we present below.

Recalling Definition 2.4, the cstit model checking problem can be broken into two parts: what is the set of optimal actions at H_0 (i.e. $K \in Optimal_\alpha^0$), and do all these optimal actions guarantee the truth of A (i.e. $K \subseteq |A|_0^{\mathcal{M}}$)? If all optimal actions guarantee A then, by Def. 2.4, $\mathcal{M}_T$ has obligation A at $0/h$. Algorithm 1 solves this problem, and is discussed in depth below.

Proposition 5.6. *Algorithm 1 returns True iff $\mathcal{M}, root/h \models \odot[\alpha\ cstit : A]$. It has complexity $O(2\sigma(|T| + c_\lambda + |T| \cdot 2^{|\phi|}))$, where σ is the maximum out-degree from any state in T, c_λ is the cost of computing the minimum and maximum values of a tactic executed on automaton T, $|T|$ is the number of states and transitions in T, and $|\phi|$ is the size of the CTL*formula in A.*

Algorithm 1 begins by considering each action available to the agent at root: $K_n \in Choice_\alpha^0$. For each of these actions, a version T_n' of the automaton T is constructed such that each of its executions is an execution of T starting with action K_n. In this way we can determine the best (u_n) and worst (ℓ_n) possible values of the executions in each action K_n by analyzing the automaton T_n' (this is discussed further in Section 5.2.1). With the range of values $[\ell_n, u_n]$ known for each action K_n, we find those ranges whose u_n is not less than any ℓ_n'. These value ranges are un-dominated. The optimal actions $Optimal_\alpha^0$ are those actions whose corresponding value ranges are un-dominated. This completes the first step of the algorithm: finding the optimal actions at H_0. The second step determines if all optimal actions guarantee A. In this algorithm $\models_{\text{CTL}^*}$ denotes the classical CTL* satisfaction relation. If the obligation is a CTL*formula, then we simply check if every execution of T_n' satisfies the A by checking $\forall A$. If the obligation is a *dstit* statement containing a CTL*formula ϕ, then we must verify two conditions: that not all actions in $Choice_\alpha^0$ guarantee ϕ, so $\exists\neg\phi$, and that every execution of T_n' with $K_n \in Optimal_\alpha^0$ satisfies ϕ.

5.2.1 *Computing Extremal History Utilities.*

In line 8 of algorithm 1, the maximum- and minimum-valued executions of an automaton T_n' must be found. This problem is related to, but distinct from, temporal logic accumulation [10] and quantitative languages [15]. A realistic example of a λ that can be computed is $\lambda = \min$. For instance, if a transition's weight $w((q, K, q'))$ is the time-to-collision when taking that transition, then the value of an execution $\lambda(\xi)$ is the shortest time-to-collision encountered along that execution. The best history, then, is the one with the greatest minimum time-to-collision. To compute $\lambda(\xi)$ for $\lambda = \min$ we proceed as follows. To avoid trivialities assume every cycle in T_n is reachable. Every infinite execution visits one or more cycles. A *simple* cycle is one that does not contain any other cycles. A prefix is a path connecting q_I to a simple cycle, and which does not itself contain a cycle. We call an execution simple if it only loops around one simple cycle forever, possibly after traversing a prefix to get there from q_I. There are finitely many simple cycles, and their prefixes are obtainable using backward reachability, so we can compute the value of every simple execution by taking the min along every connected prefix-cycle pair. The value of a non-simple execution ξ equals the value of some simple execution, since the transition of ξ with minimum weight is also a transition of a simple execution, be it on a simple cycle or a prefix. Thus, the maximum execution value u_n equals the maximum *simple* execution value. Similarly for the minimum execution value ℓ_n.

A second common accumulation function is the discounted sum function in Eq. (17). To find find the histories that carry the highest and lowest values, we cast the automaton as an extreme case of a Markov decision process (MDP).

```
   Data: A stit automaton $T = (Q, q_I, \mathcal{K}, F, \Delta, L, w, \lambda)$, an obligation $A$
   Result: $\mathcal{M}_T, root/h \models \odot[\alpha\, cstit\colon A]$
1  Set $root = 0$
2  Set $Choice^0_\alpha = \{K \in \mathcal{K} \mid (q_I, K, q') \in \Delta$ for some $q'\} = \{K_1, \ldots, K_m\}$
   // First step: find optimal actions at $root$
3  for $1 \leq n \leq m$ do
       /* Construct automaton $T'_n$ s.t. every execution of $T'_n$ is an execution of $T$ starting with
       action $K_n$. See Fig. 3.                                                                  */
4      Create automaton $T_n$ by deleting all transitions $(q_I, K, q')$ with $K \neq K_n$
5      Create a copy $T^{\text{ren}}_n$ of $T_n$
6      Create the automaton $T'_n$ as a union of $T^{\text{ren}}_n$ and $T$, with every transition $(q, K, T^{\text{ren}}_n.q_I)$ in $T^{\text{ren}}_n$ replaced by a
       transition $(q, K, T.q_I)$
8      Compute the max value, $u_n$, and min value, $\ell_n$, of any $T'_n$ tactic starting at $q_I$
9  end
   /* An interval $[\ell_n, u_n]$ is un-dominated if there is no other interval $[\ell'_n, u'_n]$, computed in the
   above for-loop, s.t. $\ell'_n > u_n$                                                              */
11 Find all un-dominated intervals $[\ell_n, u_n]$
13 Set $Optimal^0_\alpha = \{K_n \in Choice^0_\alpha \mid [\ell_n, u_n]$ is un-dominated$\}$
   /* Second step: decide whether all actions $K$ in $Optimal^0_\alpha$ guarantee $A$, i.e., $K \subseteq |A|_{root}$.   */
15 for $K_n \in Optimal^0_\alpha$ do
16     if $A$ is a $CTL^*$ formula then
           /* Does every execution of $T$ starting with $K_n$ satisfy $A$?                          */
17         Use CTL* model-checking to check whether $T'_n \models_{\text{CTL}^*} \forall A$
18         if $T'_n \not\models_{CTL^*} \forall A$                    // Optimal action $K_n$ does not guarantee $A$
19         then
21             return False
22         end
24     else if $A = [\alpha\, dstit\colon \phi]$ with $\phi \in CTL^*$ then
           // This is true iff $H_0 = |\phi|_0$
26         Model-check whether $T \models_{\text{CTL}^*} \forall\phi$
           /* This is true iff $K_n$ guarantees $\phi$, is not equiv. to line 26                    */
27         Model-check whether $T'_n \models_{\text{CTL}^*} \forall\phi$
28         if $T \models_{CTL^*} \forall\phi$ or $T'_n \not\models_{CTL^*} \forall\phi$ then
30             return False
31         end
33     else
           /* Last case: $A = \neg[\alpha\, dstit\colon \phi]$ with $\phi \in$ CTL*. Similar to previous case on line 24 with
              obvious modifications                                                              */
34     end
35 end
37 Return True
```

Algorithm 1: Model checking DAU.

An MDP is a control process modeled in discrete time where actions are chosen by a decision making agent, the outcomes of those actions are stochastic, and each outcome gives the agent some reward [6]. We specify the construction of the MDP_T cast from an automaton T in the supplementary material.

Value iteration is a dynamic programming algorithm used to solve MDPs [36]. Solving an MDP generates a policy for choosing an action at each state that optimizes some reward aggregation function λ. Following this policy from a given state q (called an "optimal policy" and denoted by $\pi^*(q)$) will produce the sequence of state transitions (denoted by $\omega^*(q)$) that maximizes accumulated rewards. The expected accumulated reward for following an optimal policy from $q \in S$ is denoted by $V^*(q)$.

Proposition 5.7. *Given a stit automaton T_n, let T_n^- be a copy of T_n where the edge weights are negated, let MDP_{T_n} be the stit MDP cast from T_n, and let $MDP_{T_n^-}$ be the stit MDP cast from T_n^-. Then, if λ is discounted sum, the extremal values of T_n are $u_n = V^*(q_I)$ in MDP_{T_n} and $\ell_n = -V^*(q_I)$ in $MDP_{T_n^-}$*

5.3 Model Checking Conditional Obligations

The problem of *conditional cstit model checking* is: given a stit automaton T that models an agent α, a state $q \in T.Q$, and a formula A as in Section 5.2 (i.e. A is either in CTL*, or a statement of the form $[\alpha\, dstit\colon \phi]$ or $\neg[\alpha\, dstit\colon \phi]$ where ϕ is in CTL*), and a finite-horizon formula B, determine whether $\mathcal{M}_{T,q}, 0/h \models \odot([\alpha\, cstit\colon A]/B)$ for some $h \in H_0$.

B is a finite horizon condition, meaning that there exists a $\tau \geq 0$ such that every history of length τ either satisfies or violates B. We note that if B is a state formula, then either all q-rooted histories satisfy B or none do. To avoid such trivialities, we only consider conditions that are specified by path formulae. In this section we introduce modifications to algorithm 1 and its proof (in the supplementary material) that reflect this difference in determining optimal actions.

Proposition 5.8. *Algorithm 2 returns True iff $\mathcal{M}, root/h \models \odot([\alpha\, cstit\colon A]/B)$. It has complexity $O(\sigma(|T|+\sigma^\tau |T|^2 2^{|B|} + \sigma^\tau \cdot c_\lambda) + \sigma|T|2^{|\phi|})$, where σ is the maximum out-degree from any state in T, c_λ is the cost of computing the minimum and maximum values of a tactic executed on automaton T, $|T|$ is the number of states and transitions in T, $|\phi|$ is the size of the CTL* formula in A, and $|B|$ is the size of the CTL* formula for the condition.*

Conceptually, getting the histories that satisfy B can be done by brute force: unroll T's executions up to depth τ and retain actions $K_n \in Choice_\alpha^0$ that contain B-satisfying histories. The values of these B-satisfying histories are compared to determine conditionally optimal actions, as per Def. 2.2 and Eq. (6). Once the conditionally optimal actions are determined, the algorithm continues as in Algo. 1.

The actual model-checker constructs incrementally automata $T'_{n,l}$: every such automaton has one initial action K_n, has a single execution up to the horizon τ, and behaves like the original automaton after τ. Its unique execution up to τ satisfies B. Algo. 2 uses these automata to determine the conditionally optimal actions by comparing B-satisfying histories, in the same way that Algo. 1 uses T'_n to compute (unconditionally) optimal actions. Alg. 3 shows how to construct $T'_{n,l}$. Each $T'_{n,l}$ has two components: a "*fragment*" of $|B|$ followed by a copy of T. The *fragment* is obtained by beginning with T'_n, removing all transitions from q_I except for one (q_I, K_n, q'), forming the union between the resulting automaton and a copy of T, and checking this new automaton to see if there exists an execution that accepts B. If it does not, it aborts this branch (line 13). If it does, it sets $q_a = q'$ (that is, we change the state we remove transitions from) and repeats the process of removing transitions, taking the union with T, and checking that the automaton accepts $\exists B$. This process repeats a maximum of τ times, ensuring that the resulting automaton has a single history for τ moments, and accepts B. This final automaton is $T'_{n,l}$.

6 CASE STUDY IN MODEL CHECKING SELF-DRIVING CARS OBLIGATIONS

As discussed in section 5, it is common for a control engineer to model agents as an automaton, and it is natural to want to verify that the automata have some given obligations. The formalizations given thus far are required to

```
Data: A stit automaton $T = (Q, q_I, \mathcal{K}, F, \Delta, L, w, \lambda)$, an obligation $A$, a horizon-limited condition $B$, the condition's
      horizon $\tau$
Result: $\mathcal{M}_T, root/h \models \odot([\alpha\, cstit \colon A]/B)$
1  Set $root = 0$
2  Set $Choice^0_\alpha = \{K \in \mathcal{K} \mid (q_I, K, q') \in \Delta$ for some $q'\} = \{K_1, \ldots, K_m\}$
   // First step: find optimal actions at root
3  for $1 \le n \le m$ do
       /* Construct automaton $T'_n$ s.t. every execution of $T'_n$ is an execution of $T$ starting with
       action $K_n$. This is exactly like lines 4, 5, 6 in Algorithm 1                              */
       /* Generate all automata whose first action is $K_n$ and have one history up to depth $\tau$ ,
       that history satisfies $B$, and after that, it behaves like $T$                              */
5      $\{T'_{n,0}, \ldots, T'_{n,l}\}$ = fragmentStep($T'_n, B, \tau, 1, q_I$)
                                                         // see Algorithm 3 for fragmentStep()
6      for $1 \le i \le l$ do
8          Compute the max value, $u_{n,i}$, and min value, $\ell_{n,i}$, of any $T'_{n,l}$ tactic starting at $q_I$
9      end
10     Set $u_n = \max_i(u_{n,i})$; Set $\ell_n = \max_i(\ell_{n,i})$;
11 end
13 Find all un-dominated intervals $[\ell_n, u_n]$
15 Set $Optimal^0_\alpha/B = \{K_n \in Choice^0_\alpha \mid [\ell_n, u_n]$ is un-dominated$\}$
   /* Once all conditionally optimal actions are found, this algorithm proceeds exactly like
   algorithm 1 starting from line 13                                                              */
```

Algorithm 2: Conditional model checking DAU.

```
Data: A stit automaton $T = (Q, q_I, \mathcal{K}, F, \Delta, L, w, \lambda)$, a horizon-limited condition $B$, the condition's horizon $\tau$, the
      automaton depth $i$, an anchor state $q_a$
Result: The set of stit automata that model fragments of $|B|$
1  Set $\{q_1, \ldots, q_m\} = \{q' \in Q \mid (q_a, K, q') \in \Delta$ for some $q'$ and some $K\}$
   // First step: find condition accepting actions at current root
2  for $1 \le l \le m$ do
       /* Construct automaton $T'_l$ s.t. every execution of $T'_l$ is an execution of $T$ starting with a
       transition to $q_l$.                                                                         */
3      Create automaton $T_l$ by deleting all transitions $(q_a, K, q)$ with $q \neq q_l$
4      Create a copy $T^{\text{ren}}_l$ of $T_l$
5      Create the automaton $T'_l$ as a union of $T^{\text{ren}}_l$ and $T$, with every transition $(q, K, T^{\text{ren}}_l.q_{I,a})$ in $T^{\text{ren}}_l$ replaced by a
       transition $(q, K, T.q_{I,a})$ where $q_{I,a}$ is any state on an execution from $q_I$ to $q_a$
6      if $T'_l \models \exists B$ then
7          if $i < \tau$ then
8              Return fragmentStep($T'_l, B, \tau, i + 1, q_l$);
9          else
10             Return $T'_l$;
11         end
12     else
13         Continue;
14     end
15 end
```

Algorithm 3: fragmentStep(T, B, τ, i, q_a): Recursively generating fragments of $|B|$.

reason about obligations while performing model checking and are a necessary component of our implementation To demonstrate the practical uses of DAU, we developed a software implementation of the model-checking algorithms of Section 5, and applied it to a controller for autonomous driving (adapted from [21]). We check the automaton for relevant CTL* missions, and for obligations and permissions related to the RSS rules.

6.1 Implementation

We implemented our algorithms for model checking obligations in Python, using calls to the nuXmv symbolic model checker [14] to dispatch CTL*model checking. Our implementation regards Stit automata models as directed graphs with edges labeled with action and weight. Operations on the graphs allow us to copy and take unions of automata as needed. The graphs can be translated to MDPs to find an action's extremal history utilities, or to a nuXmv model for CTL* model checking. The source code for our implementation can be found at https://github.com/sabotagelab/MC-DAU.

6.2 Agent Model and Model Checking Results

A hybrid continuous-time controller for autonomous highway driving is presented in [21]. The controller is meant to allow a car to merge onto a highway, and exit when desired. It is shown in [21] that if all cars are equipped with this controller, then no collisions can occur and all cars either merge and exit successfully, or drop-out, meaning that they safely abort the maneuver and go into the doNotEnter state. We modeled this controller in Fig. 4a as a stit automaton, which we will refer to as α. Each state is labeled with the atomic propositions that hold in it, and edges are labeled with α's actions, both of which are self-explanatory. The controller's objective is to ensure safe entry, cruising, and exit; it does not determine *when* to enter or exit. That is determined by a higher-level decision code and is captured here with atoms wantEntry and wantExit. It is important to note that the collision state can be reached from almost every other state: this reflects the understanding that if another agent, β, which is not equipped with this controller, takes a reckless action then it is impossible for α to avoid an accident.

We will state a number of missions, obligations, and permissions, that we might expect this automaton to satisfy, and model-check whether that is indeed the case. If not, we will amend the controller accordingly, thus demonstrating the value of obligation modeling and verification.

Missions. We formulate the following missions in CTL*[5]:

$$\mu_1 = \exists\Diamond(\text{onHighway}) \tag{20}$$

$$\mu_2 = \exists\Diamond_{\leq 4}(\text{reachExit}) \tag{21}$$

$$\mu_3 = \exists\Box(\neg\text{collision}) \tag{22}$$

The existential quantifier is used since, as noted, freedom from collision is not satisfied on all paths.

The automaton depicted in figure 4a satisfies all the missions formulated above. The first mission (μ_1) specifies that the vehicle can eventually enter the highway. The second (μ_2) states that the vehicle can reach the exit lane within four units of time (where $\Diamond_{\leq n}\, p$ means the proposition p must be met in n steps or fewer, and can be put in LTL syntax using Next and a bounded counter variable). The third mission (μ_3) specifies that there is a future where the vehicle never collides.

Obligations. For convenience, we define the 'Collision-Free' subset of states CF := {doNotEnter}.

[5]Arguably, a logic like ATL [2] might be more appropriate here, but our focus is on DAU model-checking.

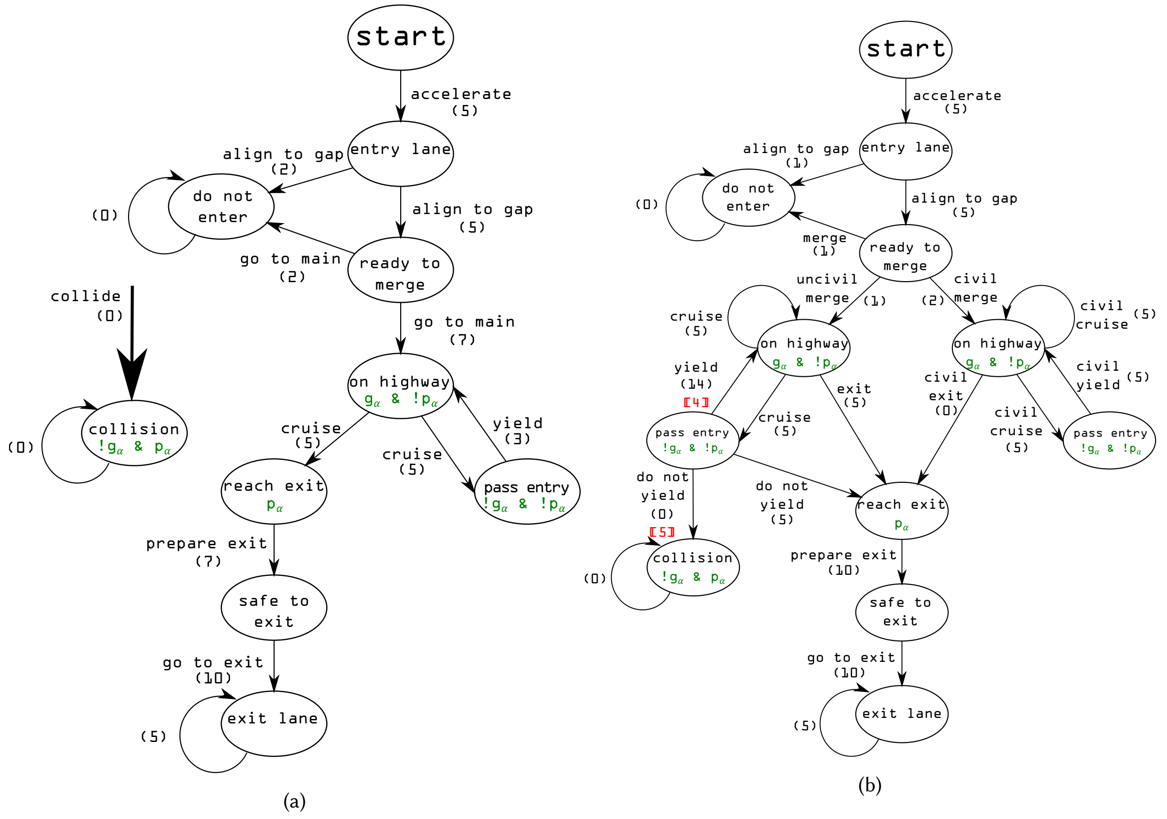


Fig. 4. Highway driving agent automaton. (a) Automaton for one agent α [21]. Each state is labeled with the atomic propositions that hold in it, and each edge is labeled with its weight, and α's actions. Edges without action labels indicate loops in absorbing states. The *collide* action can be taken from any state except doNotEnter, and so is denoted by the large arrow. The proposition g_α means the agent α has been given the right of way, while p_α means that α is proceeding through a conflict region. (b) Modified automaton adds an edge from passEntry to reachExit with action *doNotYield*. It also now models implicitly a second car β, via its actions Drive and D-β_{agg}. Alternative weights are given in red brackets to make the automaton fail the "no next collision" obligation and gain the "aggressive" permission.

No-collision: the role of modeling agency. The natural obligation $\odot[\alpha\, cstit: \Box\neg\text{collision}]$ is expected to fail in all states not in CF since, as pointed above, there is nothing that α *alone* can do to guarantee no collision. Formally, every action of α contains a history which satisfies collision at some moment. The model-checker returns UNSAT in this case, as expected. Perhaps surprisingly, the conditional obligation $\odot([\alpha\, cstit: \Box\neg\text{collision}]/\Box\neg\text{collision})$ also fails in all states not in CF. This obligation says that under the condition that the collision state is never visited, α ought to see to it that there is never a collision - which first sounds almost like a tautology. This is where DAU's ability to model agency proves essential for a proper understanding and formalization of individual obligation. Indeed, recall that in DAU, an agent has an obligation to ensure A only if it can guarantee A regardless of what other agents do (recall sure-thing reasoning and the definition of $State_\alpha^m$ in Section 2). The condition $\Box\neg$collision restricts our value comparisons to those actions that permit the condition to hold (Eq. (6)). However, it is still logically false that α *alone* can ensure no collisions: none of the conditionally optimal actions available to α guarantees no collisions. Avoidance of collisions is still a *group task*, i.e. both α and β must act to guarantee this - we take this up in section 6.3

No collision next. Are there any states not in CF at which the agent has an obligation not to collide next? To answer, we model-check the obligation

$$\odot\, [\alpha\, cstit\colon X\neg \text{collision}] \tag{23}$$

The model-checker confirms that this obligation can not be satisfied from any state not in CF. Since the agent can't guarantee another car won't collide with it, collision is included in the consequence of every action available.

Permission vs Eventually. Suppose the vehicle is actually an ambulance, that occasionally has to be able to exit the highway early. We thus want to give it permission to exit early, without forcing that behavior, and while respecting its obligations. So we model-check the permission

$$\pi_1 = \mathsf{P}[\alpha\ cstit\colon \Diamond_{\leq 4} \text{reachExit}] \tag{24}$$

from the start state (The number 4 is rather arbitrary and is meant to suggest 'early'). The model-checker informs us that the model does have this permission. Indeed, as long as the permission is checked from a state where reachExit is reachable within n steps, the permission

$$\mathsf{P}[\alpha\ cstit\colon \Diamond_{\leq n} \text{reachExit}] \tag{25}$$

will succeed for this automaton.

Assertive vs aggressive. Finally, we model-check the $RSS6$ permission at state passEntry.

$$\pi_2 = \mathsf{P}[\alpha\ cstit\colon \neg[\alpha\, dstit\colon g_\alpha \mathcal{R} \neg p_\alpha]] \tag{26}$$

The model-checker determines that this is satisfied. However, we can show that this is a trivial satisfaction, which holds regardless of the weights. It is due to the fact that all executions of this automaton starting in passEntry satisfy $g_\alpha \mathcal{R} \neg p_\alpha$. On the other hand, consider the following aggressive DAU statement:

$$\pi_3 = \mathsf{P}[\alpha\ cstit\colon [\alpha\, dstit\colon \neg(g_\alpha \mathcal{R} \neg p_\alpha)]] \tag{27}$$

This says that α is permitted to deliberately ensure that its driving is not defensive; morally, this is a less defensible permission. It does not hold because there is no action in this automaton that guarantees $\neg(g_\alpha \mathcal{R} \neg p_\alpha)$.

6.3 Modified automaton.

To draw out the effects of changing weights, we modify the automaton in Fig. 4a to get the automaton in Fig. 4b, which varies in two ways. First, when the vehicle merges onto the highway it may choose to always yield to future traffic (by doing a 'civil merge'), or to allow not yielding (by doing an 'uncivil merge'). Second, another agent β is modeled implicitly, removing most transitions to the collision state. This represents the agent β avoiding collisions with agent α by taking a *drive* action. The remaining transition to collision is taken when α chooses the *do not yield* action and β chooses a determined β aggression (or *D*-β_{agg}) action. We confirmed that this automaton still satisfies the mission formulae μ_1, μ_2, and μ_3.

No collision. With these changes, we can revisit the problem of specifying an obligation to not collide. While the obligation $\odot[\alpha\, cstit\colon \Box\neg\text{collision}]$ still fails from start, it holds (though trivially) from the many states that no longer have a path to collision. On the other hand, the obligation $\odot[\alpha\, cstit\colon X\neg\text{collision}]$ in equation (23) non-trivially holds from the passEntry state adjacent to collision. This is ensured by weighting the *yield* transition relatively heavily — guaranteeing that the *yield* action is the only optimal action.

Permission vs Eventually. Suppose again this is an ambulance that occasionally needs to exit the highway early. The permission P[α *cstit* : $\Diamond_{\leq n}$ reachExit] no longer necessarily holds in states where reachExit is reachable within n steps. We demonstrate this from the onHighway state reached by the *civil merge* action. By making *civil cruise* the optimal action, we guarantee that the optimal histories spend at least one moment in onHighway before moving to reachExit. This yields an ethically difficult position where an insistence on defensive driving negates the permission to exit the highway early, though it might be needed.

Assertive vs. Aggressive. Finally, we model-check again the RSS-type permission in Eq. (26) from passEntry. This does *not* hold in this model, as determined by the model-checker.Similarly, the permission in Eq. (27) does *not* hold because no optimal action guarantees $\neg(g_\alpha \mathcal{R} \neg p_\alpha)$.

However, by changing the weights of this automaton as depicted by the red, bracketed weights in Fig. 4b, we can satisfy permissions π_2 and π_3 at the cost of the "no next collision" obligation in Eq. (23). By ensuring that *do not yield* is an optimal action, we know that not all optimal actions guarantee $g_\alpha \mathcal{R} \neg p_\alpha$ (thus π_2 is satisfied), and we know that there exists an optimal action that guarantees $\neg(g_\alpha \mathcal{R} \neg p_\alpha)$ (thus π_3 is satisfied). As a consequence of *do not yield* being counted as an optimal action, the "no collision next" obligation fails.

7 RELATED WORK

Deontic logic and autonomous systems ethics. The need to encode and study ethical and social obligations for human-scale CPS is well-recognized [26, 27, 41], though little explored technically. This paper follows the logicist program [11] in approaching this problem, within which the deontic family of logics takes pride of place having been created specifically to reason about obligations. Standard Deontic Logic has many well-known paradoxes [22], which have spurred the proposal of alternatives to remedy them [18]. Some variations are used to specify legal and software contracts as in [35]. Alternating-time Temporal Logic (ATL) was proposed in [2] to reason about groups of agents, and used in [12] to reason about strategic obligations, and it will be interesting to connect the modeling of agency between DAU and ATL. Finally, RSS rules have been encoded in Signal Temporal Logic for the purpose of monitoring them over linear traces in [7], but notions of obligation and uncertainty were not investigated.

Temporal propagation. The most relevant work on the propagation of obligations is [13], which takes a near-product of Standard Deontic Logic and LTL to study propagation, and ends up with a semantics that resembles DAU (albeit LTL is linear time). Works that integrate deontic and temporal modalities more generally include [20] (to specify business processes), [37] for interpreted systems, and [1] for contextualized (normed) obligations.

Algorithmic aspects. Most of the work in deontic logic has been concerned with finding the 'right' axioms and inference rules that formalize our intuition about obligations and permissions, with algorithmic aspects receiving comparatively little attention. Broader work in normative multi-agent systems relies on simulation to study, for example, ways in which social norms arise [9]. Decision procedures exist for some logics, like the KED theorem prover for Standard Deontic Logic [4], and the decision procedures in [5]. There are even fewer implemented tools, such as MCMAS, the OBDD-based checker in [29] for the logic of [37], and the implementation of dyadic deontic logic in Isabelle/HOL in [8]. A proof system for a simplified version of DAU has been developed in [3, 33] to determine whether certain obligations follow from others (a 'trusted base'). We propose a model-checker, to determine whether a given automaton has an obligation, by examining directly the values it assigns to its executions. In a deployed system, theorem-proving and model-checking are likely to play complementary roles.

Interpretability. In DAU, an agent that always performs optimal actions is one that always meets its obligations. Therefore, DAU can be viewed, informally, as the logic of utility maximization. As such, it gives a *logical interpretation* to the behavior of controlled systems that maximize long-term utility, such as [19]. This connects DAU to the field of interpretable AI [25], albeit from a non-statistical perspective.

8 CONCLUSIONS

We have discussed and demonstrated the use of Dominance Act Utilitarian deontic logic for the formalization of obligations and permissions for autonomous systems. We investigated the interaction of temporal and deontic modalities to find patterns for temporal propagation of obligations. We expressed self-driving car obligations from RSS in DAU, and found undesirable consequences of these norms. We introduced algorithms to allow system designers to automatically determine if a system has an obligation, and demonstrated an implementation of these algorithms.

In the pursuit of an algorithmic account of a system's obligations, it would be desirable next to synthesize given obligations by automatically adjusting the weights. DAU could also be used in tandem with inverse reinforcement learning to learn the obligations of an agent by observing its behavior. It will also be important to study the inheritance of obligations between groups and individuals, i.e. knowing how the obligation of a group of agents impacts the obligations of agents in that group. Since deontic logic was designed for the study of ethics, this work opens the way for formal ethical analysis of autonomous system design. These considerations will help determine the suitability of DAU, and deontic logic more generally, for the design and verification of autonomous systems.

REFERENCES

[1] Thomas Ågotnes, Wiebe vanÃĆÂăder Hoek, Juan A. Rodríguez-Aguilar, Carles Sierra, and Michael Wooldridge. 2009. *A Temporal Logic of Normative Systems.* Springer Netherlands, Dordrecht, 69–106. https://doi.org/10.1007/978-1-4020-9084-4_5

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time Temporal Logic. *J. ACM* 49, 5 (Sept. 2002), 672–713. https://doi.org/10.1145/585265.585270

[3] Konstantine Arkoudas, Selmer Bringsjord, and Paul Bello. 2005. *Toward Ethical Robots via Mechanized Deontic.* Technical Report. AAAI Fall Symposium on Machine Ethics.

[4] Alberto Artosi, Paola Cattabriga, and Guido Governatori. 1994. KED: A Deontic Theorem Prover. In *on Legal Application of Logic Programming, ICLPâĂŹ94.* 60–76.

[5] Philippe Balbiani, Jan Broersen, and Julien Brunel. 2009. Decision Procedures for a Deontic Logic Modeling Temporal Inheritance of Obligations. *Electronic Notes in Theoretical Computer Science* 231 (2009), 69 – 89. https://doi.org/10.1016/j.entcs.2009.02.030 Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).

[6] Richard Bellman. 1957. A Markovian Decision Process. *Indiana Univ. Math. J.* 6 (1957), 679–684. Issue 4.

[7] Heni Ben Amor, Aviral Shrivastava, Lina Karam, Adel Dokhanchi, Shakiba Yaghoubi, Mohammad Hekmatnejad, and Georgios Fainekos. 2019. Encoding and Monitoring Responsibility Sensitive Safety Rules for Automated Vehicles in Signal Temporal Logic. https://doi.org/10.1145/3359986.3361203

[8] Christoph Benzm̊uller, Ali Farjami, and Xavier Parent. 2018. A Dyadic Deontic Logic in HOL. In *DEON.*

[9] Guido Boella, Leendert van der Torre, and Harko Verhagen. 2006. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory* 12 (10 2006), 71–79. https://doi.org/10.1007

[10] Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. 2014. Temporal Specifications with Accumulative Values. *ACM Trans. Comput. Logic* 15, 4, Article 27 (July 2014), 25 pages.

[11] Selmer Bringsjord, Konstantine Arkoudas, and Paul Bello. 2006. Toward a General Logicist Methodology for Engineering Ethically Correct Robots. *IEEE Intelligent Systems* 21 (07 2006), 38–44. https://doi.org/10.1109/MIS.2006.82

[12] Jan Broersen. 2006. Strategic Deontic Temporal Logic as a Reduction to ATL, with an Application to Chisholm's Scenario. In *Deontic Logic and Artificial Normative Systems*, Lou Goble and John-Jules Ch. Meyer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 53–68.

[13] Jan Broersen and Julien Brunel. 2008. 'What I fail to do Today, I Have to Do Tomorrow': A Logical Study of the Propagation of Obligations. In *Computational Logic in Multi-Agent Systems*, Fariba Sadri and Ken Satoh (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 82–99.

[14] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. 2014. The nuXmv Symbolic Model Checker. In *CAV (Lecture Notes in Computer Science, Vol. 8559)*, Armin Biere and Roderick

Bloem (Eds.). Springer, 334–342.

[15] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. 2008. Quantitative Languages. In *Computer Science Logic*, Michael Kaminski and Simone Martini (Eds.). Springer Berlin Heidelberg, 385–400.

[16] B.F. Chellas. 1968. *The Logical Form of Imperatives.* Department of Philosophy, Stanford University.

[17] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 1999. *Model Checking.* MIT Press, Cambridge, Massachusetts.

[18] Dov Gabbay, John Horty, and Xavier Parent (Eds.). 2013. *Handbook of deontic logic and normative systems.* College Publications.

[19] J. Christian Gerdes and Sarah M. Thornton. 2015. *Implementable Ethics for Autonomous Vehicles.* Springer Berlin Heidelberg, Berlin, Heidelberg, 87–102.

[20] Laura Giordano, Alberto Martelli, and Daniele Theseider Dupré. 2013. Temporal Deontic Action Logic for the Verification of Compliance to Norms in ASP. In *Proc. of the 14th Intl. Conf. on Artificial Intelligence and Law* (Rome, Italy) *(ICAIL '13)*. ACM, New York, NY, USA, 53–62.

[21] Alain Girault. 2004. A hybrid controller for autonomous vehicles driving on automated highways. *Transportation Research Part C: Emerging Technologies* 12, 6 (2004), 421 – 452. https://doi.org/10.1016/j.trc.2004.07.008

[22] Risto Hilpinen and Paul McNamara. 2013. Deontic Logic: A historical survey and introduction.

[23] John Horty. 2001. *Agency and Deontic Logic.* Cambridge University Press.

[24] John F. Horty and Nuel Belnap. 1995. The deliberative stit: A study of action, omission, ability, and obligation. *J. Philos. Logic* (1995), 583âĂŞ644.

[25] Susmit Jha, Tuhin Sahai, Vasumathi Raman, Alessandro Pinto, and Michael Francis. 2019. Explaining AI Decisions Using Efficient Methods for Learning Sparse Boolean Formulae. *J Autom Reasoning* (2019), 1055âĂŞ–1075.

[26] Piotr Kulicki, Robert Trypuz, and Michael P. Musielewicz. 2018. Towards a Formal Ethics for Autonomous Cars. In *Deontic Logic and Normative Systems - 14th International Conference, DEON 2018, Utrecht, The Netherlands, July 3-6, 2018*, Jan M. Broersen, Cleo Condoravdi, Nair Shyam, and Gabriella Pigozzi (Eds.). College Publications, 193–209.

[27] Patrick Lin, Keith Abney, and George A. Bekey (Eds.). 2014. *Robot Ethics: The Ethical and Social Implications of Robotics.* The MIT Press.

[28] Gert-Jan Lokhorst. [n.d.]. MallyâĂŹs Deontic Logic. *The Stanford Encyclopedia of Philosophy (Summer 2019 Edition)* ([n. d.]). https://plato.stanford.edu/archives/sum2019/entries/mally-deontic/.

[29] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. *Intl. Jrnl. on Software Tools for Technology Transfer* 19, 1 (01 Feb 2017), 9–30.

[30] Ernst Mally. 1926. *Grundgesetze des Sollens. Elemente der Logik des Willens.*

[31] Zohar Manna and Amir Pnueli. 1992. *The Temporal Logic of Reactive and Concurrent Systems — Specification.* Springer.

[32] Paul McNamara. 2018. Deontic Logic. *The Stanford Encyclopedia of Philosophy* (Fall 2018).

[33] Yuko Murakami. 2004. Utilitarian Deontic Logic. In *in âĂŸProceedings of the Fifth International Conference on Advances in Modal Logic (AiML 2004.* 288–302.

[34] Amir Pnueli. 1977. The Temporal Logic of Programs. In *Proceedings of the 18th IEEE Symposium Foundations of Computer Science.* 46–57.

[35] Cristian Prisacariu and Gerardo Schneider. 2012. A dynamic deontic logic for complex contracts. *The Journal of Logic and Algebraic Programming* 81, 4 (2012), 458 – 490. Special Issue: NWPT 2009.

[36] Martin L Puterman. 1994. *Markov decision processes : discrete stochastic dynamic programming.* Wiley, New York.

[37] Franco Raimondi and Alessio Lomuscio. 2004. Automatic Verification of Deontic Interpreted Systems by Model Checking via OBDD's. In *Procs. of the 16th European Conf. on Artificial Intelligence.*

[38] A. Rizaldi and M. Althoff. 2015. Formalising Traffic Rules for Accountability of Autonomous Vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems.* 1658–1665.

[39] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2018. On a Formal Model of Safe and Scalable Self-driving Cars. (October 2018). arXiv:1708.06374v6.

[40] Colin Shea-Blymyer and Houssam Abbas. 2020. A Deontic Logic Analysis of Autonomous SystemsâĂŹ Safety. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control* (Sydney, New South Wales, Australia) *(HSCC âĂŹ20)*. Association for Computing Machinery, New York, NY, USA, Article 26, 11 pages. https://doi.org/10.1145/3365365.3382203

[41] A. Thekkilakattil and G. Dodig-Crnkovic. 2015. Ethics Aspects of Embedded and Cyber-Physical Systems. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, Vol. 2. 39–44.

[42] Georg H. von Wright. 1951. Deontic Logic. *Mind* 60, 237 (January 1951).

[43] Eric Wiewiora. 2010. *Reward Shaping.* Springer US, Boston, MA, 863–865. https://doi.org/10.1007/978-0-387-30164-8_731