


A Knowledge-Based Analysis of Intersection Protocols

Kaya Alpturer ✉ 

Princeton University, NJ, USA

Joseph Y. Halpern ✉ 

Cornell University, Ithaca, NY, USA

Ron van der Meyden ✉ 

UNSW Sydney, Australia

Abstract

The increasing wireless communication capabilities of vehicles creates opportunities for more efficient intersection management strategies. One promising approach is the replacement of traffic lights with a system wherein vehicles run protocols among themselves to determine right of way. In this paper, we define the *intersection problem* to model this scenario abstractly, without any assumptions on the specific structure of the intersection or a bound on the number of vehicles. Protocols solving the intersection problem must guarantee safety (no collisions) and liveness (every vehicle eventually goes through). In addition, we would like these protocols to satisfy various optimality criteria, some of which turn out to be achievable only in a subset of the contexts. In particular, we show a partial equivalence between eliminating unnecessary waiting, a criterion of interest in the distributed mutual-exclusion literature, and a notion of optimality that we define called *lexicographical optimality*. We then introduce a framework to design protocols for the intersection problem by converting an *intersection policy*, which is based on a global view of the intersection, to a protocol that can be run by the vehicles through the use of knowledge-based programs. Our protocols are shown to guarantee safety and liveness while also being optimal under sufficient conditions on the context. Finally, we investigate protocols in the presence of faulty vehicles that experience communication failures and older vehicles with limited communication capabilities. We show that intersection protocols can be made safe, live and optimal even in the presence of faulty behavior.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Computer systems organization → Dependable and fault-tolerant systems and networks; Computing methodologies → Reasoning about belief and knowledge

Keywords and phrases Intersection management, Autonomous vehicles, Distributed algorithms, Epistemic logic, Fault tolerance

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.2

Related Version *Full Version*: <https://arxiv.org/abs/2408.09499> [2]

Funding *Kaya Alpturer*: Research supported by AFOSR grant FA23862114029. Work done in part while the author was studying at Cornell University.

Joseph Y. Halpern: Research supported in part by AFOSR grant FA23862114029, NSF grant FMITF 2319186, ARO grant W911NF-22-1-0061, and MURI grant W911NF-19-1-0217.

Ron van der Meyden: The Commonwealth of Australia (represented by the Defence Science and Technology Group) supported this research through a Defence Science Partnerships agreement.

1 Introduction

Traffic lights can slow down traffic significantly, due to their lack of responsiveness to real-time traffic. If vehicles can communicate with each other (which is already quite feasible with today's wireless technology), the door is open for improved protocols, where vehicles can determine right of way among themselves, depending on traffic conditions, and thereby



© Kaya Alpturer, Joseph Y. Halpern, and Ron van der Meyden;
licensed under Creative Commons License CC-BY 4.0

38th International Symposium on Distributed Computing (DISC 2024).

Editor: Dan Alistarh; Article No. 2; pp. 2:1–2:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

significantly increase throughput at an intersection. In this paper, we formally define the *intersection problem*: we assume that agents can communicate with each other via radio broadcasts, and design protocols that take advantage of this communication to allow agents to go through the intersection while satisfying *safety* (no collisions) and *liveness* (every vehicle eventually goes through). In addition, we consider *optimal* protocols, which means, roughly speaking, that the protocol allows as many vehicles as possible to go through the intersection at any given time. Finally, we consider the extent to which we can tolerate communication failures and (older) vehicles that are not equipped with wireless, so cannot broadcast messages. (It turns out that these two possibilities can be dealt with essentially the same way.)

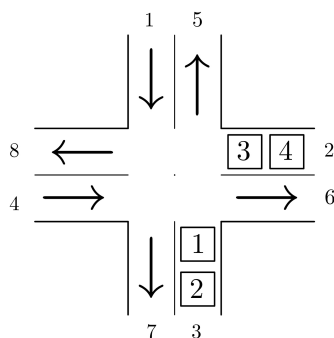
While the inefficiencies of traffic-light-based intersection management have long been recognized [7], prior approaches have mainly focused on specific intersection scenarios [15, 16] or relied on executing leader-election protocols without considering communication failures [9, 10]. Furthermore, the protocols have often been evaluated based on simulations of specific intersections, rather than being proved correct [10, 15]. Given the implications of this problem for traffic safety, as well as its potential for greatly improving energy efficiency and productivity, there is a need for formal guarantees on both correctness and optimality.

To the best of our knowledge, prior work did not consider optimality, especially in the presence of various faults. In designing these protocols, to the extent possible, we want them to be robust to a variety of communication failures, such as contexts with *crash* failures,¹ where an agent may fail by ceasing to participate in the protocol at a given time, and *omission* failures, where arbitrary messages can fail to be broadcast.

Epistemic logic has been shown to provide a high-level abstraction that can be used to design distributed protocols independent of particular assumptions on the communication environment and type of failures [8]. Most analyses of distributed-computing problems that use epistemic logic have used full-information protocols to derive time-optimal algorithms, at the cost of large message size and memory requirements. Given the limitations of wireless networks, it is also desirable to bound the amount of information that needs to be exchanged between agents, while still ensuring that the formal guarantees are still met. To address this, following [1], we separate the part of the protocol that determines what information is exchanged between the agents, and the part that determines what action to take based on the agent’s information. Thus, when we consider optimality, we do so with respect to protocols that limit information exchange in the same way.

We model the intersection problem as the following scenario. There is a (possibly infinite) set of agents $Ag \subseteq \mathbb{N}$. The intersection has ℓ lanes, represented by $\mathcal{L} = \{0, \dots, \ell - 1\}$. The set of lanes is partitioned into a set of lanes $\mathcal{L}_{in} = \{0, \dots, k - 1\}$, where $1 < k < \ell$ by which vehicles approach the intersection, and a set of lanes \mathcal{L}_{out} by which they depart from the intersection. Each lane in \mathcal{L}_{in} has a queue of agents waiting to go through the intersection; at each point in time at most one agent arrives at each of these queues. A *move* through the intersection is represented by a pair $(l_s, l_t) \in \mathcal{L}_{in} \times \mathcal{L}_{out}$. Intuitively, executing (l_s, l_t) means that the agent arrives through lane l_s and departs through lane l_t . The symmetric relation $\mathcal{O} \subseteq (\mathcal{L}_{in} \times \mathcal{L}_{out})^2$ describes which moves of the agents are compatible; $((l_s, l_t), (l'_s, l'_t)) \in \mathcal{O}$ means that both (l_s, l_t) and (l'_s, l'_t) can be executed in the same round. Broadcasts have a limited range, given by $\rho > 0$. We assume that, provided there are no failures, all broadcasts sent by an agent i will be received by all agents that are within a distance ρ of i . The

¹ We follow the distributed-algorithms literature’s interpretation of “crash failure” here: it is not meant to imply a physical collision.



■ **Figure 1** An intersection with $\mathcal{L} = \{1, \dots, 8\}$ where $\mathcal{L}_{in} = \{1, \dots, 4\}$ and $\mathcal{L}_{out} = \{5, \dots, 8\}$. There are currently 4 agents that have arrived in incoming lanes 2 and 3.

problem is then to maximize the rate at which cars move through the intersection while guaranteeing safety (it is never the case that agents with incompatible moves go through the intersection simultaneously) and liveness (all agents that arrive at the intersection eventually move through it). The problem can be thought of as a generalization of distributed mutual exclusion, where the intersection is the critical section.

The rest of the paper is organized as follows: In Section 2, we briefly review the knowledge-based framework of [8]. In Section 3, we modify the information-exchange model of [1] and introduce the sensor model. Section 4 defines models for the adversary which determine the arrival schedule of vehicles and communication failures. Section 5 combines the information-exchange and the adversary model, fully specializing the general model of Section 2 to intersections. Section 6 introduces the various notions of optimality we care about such as eliminating unnecessary waiting and lexicographical optimality. In Section 7, intersection policies are introduced as a global view of the intersection. Section 8 proves a construction that results in an optimal policy even with failures, and explores applications of the construction in two limited-information contexts. Section 9 concludes with a discussion on connections to distributed mutual exclusion. We defer most proofs to the full paper.

2 Reasoning about knowledge

In order to reason about the knowledge of the vehicles in the intersection problem, we use the standard runs-and-systems model [8]. An interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ consists of a system \mathcal{R} , which is a set of runs, and an *interpretation* $\pi : \mathcal{R} \times \mathbb{N} \rightarrow \mathcal{P}(\text{Prop})$. Each *run* $r : \mathbb{N} \rightarrow L_e \times \prod_{i \in Ag} L_i$ describes a particular infinite execution of the system where $r(m)$ is the global state of the system in run r at time m . The global states consist of an environment state drawn from L_e and local states for each agent i drawn from each L_i . The local state of agent i at point (r, m) is denoted $r_i(m)$. We call a run and time pair (r, m) a *point*. The interpretation π describes which atomic propositions hold at each point in a system \mathcal{R} .

We write $\mathcal{I}, (r, m) \models \phi$ if the formula ϕ holds (is satisfied) at point (r, m) in interpreted system \mathcal{I} . A formula ϕ is *valid in an interpreted system* \mathcal{I} , denoted $\mathcal{I} \models \phi$, if ϕ holds at all points in \mathcal{I} ; the formula ϕ is *valid* if it is valid in all interpreted systems. Satisfaction of formulas is inductively defined as follows:

- $\mathcal{I}, (r, m) \models p$ iff $p \in \pi(r, m)$.
- $\mathcal{I}, (r, m) \models \phi \wedge \phi'$ iff $\mathcal{I}, (r, m) \models \phi$ and $\mathcal{I}, (r, m) \models \phi'$.
- $\mathcal{I}, (r, m) \models \neg\phi$ iff $\mathcal{I}, (r, m) \not\models \phi$.

- $\mathcal{I}, (r, m) \models K_i \phi$ iff $\mathcal{I}, (r', m') \models \phi$ for all points (r', m') such that $r_i(m) = r'_i(m')$.
- $\mathcal{I}, (r, m) \models \Diamond \phi$ iff for some $m' \geq m$, $\mathcal{I}, (r, m') \models \phi$.
- $\mathcal{I}, (r, m) \models \bigcirc \phi$ iff $\mathcal{I}, (r, m+1) \models \phi$.

Agent i *knows* a formula ϕ at (r, m) if $\mathcal{I}, (r, m) \models K_i \phi$. Intuitively, agent i knows ϕ if ϕ holds at all points where agent i has the same local state. We say that agent i considers the point (r', m') *possible* at point (r, m) if $r_i(m) = r'_i(m')$. The relation \sim_i is defined as $(r, m) \sim_i (r', m')$ iff $r_i(m) = r'_i(m')$. The formula $\bigcirc \phi$ means that ϕ holds at the next time, and $\Diamond \phi$ means that ϕ holds eventually. In later sections, we formalize how interpreted systems for the intersection problem are specified.

3 Information-exchange protocols

Our framework for modeling limited information exchange is similar to that used by Alpturer et al. [1] to analyze consensus protocols, but we make a number of changes due to the differences in our setting. Here, global states represent not just the result of messages sent between the agents, but also facts about a changing external world, from which the agents obtain sensor readings (e.g., information about their own position and that of nearby vehicles, from GPS, visual, lidar, or radar sensors). We modify the definition of information-exchange protocols from [1] to accommodate these sensor readings. Specifically, assume that we are given a set L_e of environment states. Define a *sensor model* for L_e to be a collection of mappings $\mathcal{S} = \{\mathcal{S}_i\}_{i \in Ag}$, where $\mathcal{S}_i : L_e \rightarrow \Sigma_i$ maps states of the environment to a set Σ_i of possible sensor readings for agent i .

An *information-exchange protocol* \mathcal{E} for agents Ag and sensor model \mathcal{S} is given by the collection $\{\mathcal{E}_i\}_{i \in Ag}$ consisting of a local information-exchange protocol \mathcal{E}_i for each agent i . Each local information-exchange protocol \mathcal{E}_i is a tuple $\langle L_i, Mem_i^{init}, A_i, M_i, \mu_i, \delta_i \rangle$, where

- $L_i = Mem_i \times \Sigma_i$ is a set of local states, where each local state consists of a memory state from a set Mem_i and a sensor reading from Σ_i ;
- $Mem_i^{init} \subseteq Mem_i$ is a set of initial memory states. (Typically, there might be a single initial memory state, containing information such as the agent's identity.)
- M_i is the set of messages that can be sent by agent i ;
- $\mu_i : L_i \times A_i \times \Sigma_i \rightarrow M_i \cup \{\perp\}$ is a function mapping a local state s , an action a , and a sensor reading o to the message to be broadcast (intuitively, $\mu_i(s, a, o) = m$ means that when agent i performs action a in state s and obtains new sensor reading o , the information-exchange protocol broadcasts the message m to the other agents; if $m = \perp$, then no message is sent by i);
- $\delta_i : L_i \times A_i \times \mathcal{P}(\cup_{j \in Ag} M_j) \rightarrow Mem_i$ is a function that updates the local memory as a function of the previous local state (comprised of the previous memory state and the previous sensor reading), an action, and a set of messages received.

An *action protocol* P for an information-exchange protocol \mathcal{E} , is a tuple $\{P_i\}_{i \in Ag}$ containing, for each agent i , $P_i : L_i \rightarrow A_i$ mapping the local states L_i for agent i in \mathcal{E}_i to actions in A_i .

4 Adversary model

Intersection protocols need to operate in an environment with several forms of nondeterminism: how messages are broadcast through the environment, failures of transmitters and receivers, and the arrival pattern of vehicles. We model these aspects of the environment in terms of an adversary.

The precise physics of the intersection may affect how broadcasts are transmitted through the environment. Rather than attempt to model Euclidean distances and obstacles, we abstract the effects of these factors on transmission. A *transmission environment* is a relation $T \subseteq (\mathcal{L}_{in} \times \mathbb{N})^2$. Intuitively, $((\ell, p), (\ell', p')) \in T$ represents that, provided the agents' transmitters and receivers do not fail, a message broadcast by an agent at position p in lane ℓ , will be received by an agent at position p' in lane ℓ' . Transmission environments encode our assumption that the communication range is ρ . We make one assumption about this relation: that for all $\ell, \ell' \in \mathcal{L}_{in}$, we have $((\ell, 0), (\ell', 0)) \in T$. That is, messages broadcast by an agent at the front of some lanes are received (barring failure) by all agents that are at the front of any lane.

An *adversary model* \mathcal{F} is a set of adversaries; formally, an *adversary* is a tuple $\alpha = (\tau, T, F_t, F_r)$, where $\tau : Ag \rightarrow \mathbb{N} \times \mathcal{L}_{in} \times \mathcal{L}_{out}$, T is a transmission environment, $F_t : \mathbb{N} \times Ag \rightarrow \{0, 1\}$, and $F_r : \mathbb{N} \times Ag \rightarrow \{0, 1\}$. Intuitively, τ is an *arrival schedule*, which describes when each agent arrives in the system (i.e., enters a queue), its lane of arrival, and its intended departure lane. The function F_t represents failures of agents' transmitters and the function F_r represents failures of agents' receivers. $F_t(k, i) = 1$ means that if i tries to broadcast in round $k + 1$ (i.e., between time k and time $k + 1$), then the broadcast will be sent to all agents within range (i.e., within ρ of i), and perhaps others; similarly, $F_r(k, j) = 1$ means that j receives all broadcasts sent in round $k + 1$ by agents within range (but again, it may receive other broadcasts as well). Thus, a broadcast by agent i in round $k + 1$ is received by a j within range of i in round $k + 1$ iff $F_t(k, i) = F_r(k, j) = 1$. The function τ describes when agents arrive in the system (which we assume is under the control of the adversary). In more detail, if $\tau(j) = (k, (l_1, l_2))$, then at time k , agent j arrives in the system on lane l_1 with the intention of departing on lane l_2 . We assume that τ is *conflict-free* in the sense that, for all agents $i \neq j$, if $\tau(i) = (k, (l_1, l_2))$ and $\tau(j) = (k, (l'_1, l'_2))$, then $l_1 \neq l'_1$. This ensures that we do not have a conflict of two agents wanting to enter the same queue for lane l_1 simultaneously. (Exactly how this mutual exclusion of queue entry is assured is outside the scope of the model. One way that it may come about is that vehicles approaching the intersection are already ordered along an approaching lane.)

We consider adversary models that involve the following types of failures:

- No failures (*NF*): the set of all adversaries (τ, T, F_t, F_r) where $F_r(k, i) = F_t(k, i) = 1$ for all $i \in Ag$ and $k \in \mathbb{N}$.
- Crash failures (*CR*): the set of all adversaries (τ, T, F_t, F_r) where for all $i \in Ag$ and $k \in \mathbb{N}$, (1) $F_t(k, i) = 0$ implies $F_t(k', i) = 0$ for all $k' > k$, and (2) $F_r(k, i) = 1$ for all k and i .
- Sending omissions (*SO*): the set of all adversaries (τ, T, F_t, F_r) where for all $i \in Ag$ and $k \in \mathbb{N}$, $F_r(k, i) = 1$.

An adversary model \mathcal{F} has a *fixed transmission environment* if all adversaries in \mathcal{F} include the same transmission environment T . We believe that our techniques can be applied without change to the general omissions case.

5 Intersection Contexts

A *context* is a triple $(\mathcal{E}, \mathcal{F}, \pi)$ consisting of an information-exchange protocol \mathcal{E} , an adversary model \mathcal{F} , and an interpretation π . To deal with intersections, we restrict information-exchange protocols and interpretations so that they satisfy certain conditions. $(\mathcal{E}, \mathcal{F}, \pi)$ is an *intersection context* if it satisfies the following conditions:

- The set of environment states L_e consists of states of the form $s_e = (\alpha, t, q_1, \dots, q_{|\mathcal{L}_{in}|}, done)$ where $\alpha \in \mathcal{F}$ is an adversary, $t \in \mathbb{N}$ is a time, for each approach lane $l \in \mathcal{L}_{in}$, q_l is a queue (list) of agents, intuitively the ones who have lane i and not yet departed, and a set $done \subseteq Ag$, representing the agents that have already passed through the intersection.
- The sensor model, in principle, could be defined to include information from a large variety of sensors and information sources, such as GPS, in-road or road-side beacons, lidar, radar, or vision systems. We start with a minimal location-based sensor model, and leave it open for other fields to be added. Our minimal sensor model $\mathcal{S} = \{\mathcal{S}_i\}_{i \in Ag}$ is defined so that the sensor function \mathcal{S}_i maps environment states to tuples of the form $\langle front_i, lane_i, intent_i \rangle$, where $front_i \in \{0, 1\}$, $lane_i \in \mathcal{L}_{in} \cup \{\perp, \top\}$, and $intent_i \in \mathcal{L}_{out}$. For $s_e = (\alpha, t, q_0, q_1, \dots, q_{|\mathcal{L}_{in}|}, done)$, we have $\mathcal{S}_i(s_e) = \langle front_i, lane_i, intent_i \rangle$, where if τ is the arrival schedule in the adversary α ,
 - pos_i maps from global states to $\mathbb{N} \cup \{\perp, \top\}$; $pos_i(s_e) = \top$ if $i \in done$, $pos_i(s_e) = k$ if there exists a queue l such that i is the k th position in queue q_l (with the front of the queue counted as position 0), and $pos_i(s_e) = \perp$ otherwise. (It follows from the state dynamics given below that i is in at most one queue, so pos_i is well-defined.)
 - $front_i = 1$ iff $pos_i(s_e) = 0$,
 - if i is in the queue q_l for lane l , then $lane_i = l$; if $i \in done$ then $lane_i = \top$; and if $i \notin done$ then $lane_i = \perp$.
 - if $\tau(i) = (k, (l, l'))$ then $intent_i = l'$.

We have modelled an agent's intended departure lane $intent_i$ as being received from the environment since, from the point of view of protocol design, this is part of the adversary.

- The set of possible actions of agent i in \mathcal{E}_i is $A_i = \{\text{go}, \text{noop}\}$. Intuitively, **go** represents that action of the agent making its planned move through the intersection. This action can be performed by agent i only if i is at the front of its queue. The action **noop** represents that the agent does not move, unless it is either scheduled for arrival in some queue, or in some position in a queue but not at the front, and the position before it is being vacated, in which case it advances in the queue.
- A global state is a tuple of the form $(s_e, \{s_i\}_{i \in Ag})$, where $s_e \in L_e$ and $s_i \in L_i$ for each agent $i \in Ag$. An *initial* global state has
 - $s_e = (\alpha, t, q_1, \dots, q_{|\mathcal{L}_{in}|}, done)$, where $t = 0$, each queue q_l is empty, and $done$ is the empty set, and
 - for each agent $i \in Ag$, the local state $s_i = (m_i, \mathcal{S}_i(s_e))$ where $m_i \in Mem_i^{init}$ is an initial memory state.
- π interprets the following atomic propositions based on the global state in the obvious way: $front_i, lane_i = l$ for $l \in \mathcal{L}_{in}$, $intent_i = l$ for $l \in \mathcal{L}_{out}$, $pos_i = k$ for $k \in \mathbb{N} \cup \{\perp, \top\}$.

Given an intersection context $\gamma = (\mathcal{E}, \mathcal{F}, \pi)$ and a protocol P , we construct an interpreted system $\mathcal{I}_{\gamma, P} = (\mathcal{R}_{\mathcal{E}, \mathcal{F}, P}, \pi)$ representing all the possible behaviours of the protocol P in context γ . The set $\mathcal{R}_{\mathcal{E}, \mathcal{F}, P}$ of runs consists of all runs r that satisfy the following properties:

- The initial state $r(0)$ of r is an initial global state.
- For each $k \in \mathbb{N}$, the global state $r(k+1) = (s'_e, \{s'_i\}_{i \in Ag})$ is determined from $r(k) = (s_e, \{s_i\}_{i \in Ag})$ by a procedure in which the order of events is as follows. First, the agents decide their actions (to go through the intersection or not). They then perform these actions, causing the queues to be updated; any newly arriving agents are also added to the queues in this step. The agents then take a sensor reading, from which they obtain new information about their position. This new information may be included in the message that an agent broadcasts. Finally, each agent updates its memory state, based

on their previous local state, the action performed, and the messages that were broadcast in the current round and received by the agent. We then proceed to the next round. Formally, state transitions are determined by the following procedure:

- First, each agent i determines its action $P_i(s_i)$ according to the protocol P .
- If $s_e = (\alpha, m, q_1, \dots, q_{|\mathcal{L}_m|}, done)$, then we take $s'_e = (\alpha, m + 1, q'_1, \dots, q'_{|\mathcal{L}_m|}, done')$, defined as follows. Note that the adversary α is the same in s'_e , and the time m is incremented. Each queue q'_ℓ is obtained from q_ℓ by the following operations:
 - * If $q_\ell(0) = i$ and $P_i(s_i) = \text{go}$, then let q''_ℓ be the result of dequeuing agent i from q_ℓ . Otherwise $q''_\ell = q_\ell$.
 - * If $\tau(i) = (m+1, (l_1, l_2))$ for any agent i , then we define $q'_\ell = \text{enqueue}(i, q''_\ell)$, otherwise $q'_\ell = q''_\ell$. (Recall that such an i is unique, by assumption on τ .)
- Finally, we take $done'$ to be the result of adding to the set $done$ all agents i who were at the front of any queue in s_e such that $P_i(s_i) = \text{go}$.
- Next, for each agent i , we obtain a new sensor reading $\mathcal{S}_i(s'_e)$ of the updated state s'_e of the environment. Using this sensor readings, each agent i constructs the message $m_i = \mu_i(s_i, P_i(s_i), \mathcal{S}_i(s'_e))$, which it broadcasts.
- For each agent i , we determine the set of messages B_i^m that the agent receives in round $m+1$. If agent i is not in any queue in state s'_e , or $F_r(m, i) = 0$ (agent i 's receiver fails in round $m+1$) then $B_i^m = \emptyset$. Otherwise, for each agent i that is in a lane queue, let ℓ_i be the lane it is in and p_i its position in the queue. We define B_i^m to be the set of messages m_j for which both $((p_j, \ell_j), (p_i, \ell_i)) \in T$ (j 's transmission can be heard by agent j , given their positions) and $F_i(m, j) = 1$ (j 's transmitter does not fail in this round.)
- Finally, if $s_i = (u_i, \mathcal{S}_i(s_e))$, then $s'_i = (u'_i, \mathcal{S}_i(s'_e))$, where $u'_i = \delta_i(s_i, P_i(s_i), B_i^m)$. (Note that we use the old sensor reading $\mathcal{S}_i(s_e)$ to determine the new memory state, but not the new sensor reading $\mathcal{S}_i(s'_e)$, since the latter will be visible to the agent in its new local state s'_i .)

P is an *intersection protocol* for context $\gamma = (\mathcal{E}, \mathcal{F}, \pi)$ if the following are valid in $\mathcal{I}_{\gamma, P}$ for all $i, j \in \text{Ag}$ where $i \neq j$, where $going_i$ is an abbreviation for $front_i \wedge \neg front_i$.

- **Validity:** $going_i \Rightarrow front_i$.
- **Safety:** $(going_i \wedge going_j) \Rightarrow ((lane_i, intent_i), (lane_j, intent_j)) \in \mathcal{O}$.
- **Liveness:** $front_i \Rightarrow \Diamond going_i$.

Intuitively, **Validity** states that an agent does not move through the intersection unless it is at the front of the queue in its lane. **Safety** states that if two agents go through the intersection at the same time, their moves are compatible and do not cause a collision. (Note that the semantics of the action **go** has been defined so as to ensure that an agent makes its planned move, and not any other.) **Liveness** states that an agent eventually gets to make its move through the intersection. (The model implicitly assumes that vehicles do not have mechanical failures and block other vehicles in their lane.)

6 Unnecessary waiting and optimality

One desirable property of an intersection protocol is that it never makes agents wait unnecessarily. Eliminating unnecessary waiting is also a criterion that has been considered in the distributed mutual-exclusion literature [14]. Intuitively, unnecessary waiting occurs if, given what happens in a certain run r , there is a point where if an agent had gone through the intersection instead of waiting, safety would not be violated. In this section, we define a notion of optimality that captures eliminating unnecessary waiting.

We first give some definitions to define unnecessary waiting and a domination-based notion of optimality. For an intersection context γ and protocol P ,

- $GO(r, m)$ is the set of agents that go through the intersection in round $m + 1$, that is, the agents i with $\mathcal{I}_{\gamma, P}(r, m) \models \text{going}_i$.
- $\mathcal{I}_{\gamma, P}(r, m) \models \text{safe-to-go}_i$ if $\mathcal{I}_{\gamma, P}(r, m) \models \text{pos}_i = 0$ and for all agents $j, k \in GO(r, m) \cup \{i\}$ where $j \neq k$, $(\text{lane}_j(r, m), \text{intent}_j(r, m))$ and $(\text{lane}_k(r, m), \text{intent}_k(r, m))$ are compatible moves according to \mathcal{O} .
- For a run r of a protocol P in context γ , define $\text{gotime}(r, i)$ to be the time $m \in \mathbb{N}$ such that $\mathcal{I}_{\gamma, P}(r, m) \models \text{going}_i$, and ∞ if there is no such time.
- $\text{front}(r, m)$ is the set of agents that are in front of each queue, that is, the agents i with $\text{front}_i(r, m) = 1$.

► **Definition 1** (unnecessary waiting). *An intersection protocol P has unnecessary waiting with respect to an intersection context γ if there exists $i \in \text{Ag}$ and point (r, m) such that $\mathcal{I}_{\gamma, P}(r, m) \models \text{safe-to-go}_i$ and $i \notin GO(r, m)$.*

► **Definition 2** (corresponding runs). *Given action protocols P, P' and context γ , two runs $r \in \mathcal{I}_{\gamma, P}$ and $r' \in \mathcal{I}_{\gamma, P'}$ correspond if $r(0) = r'(0)$.*

Intuitively, corresponding runs have the same adversary, so agents arrive at the intersection in the same sequence and at the same times in the two runs. We use this notion to define the following notion of one protocol being better than another if it always ensures a faster flow of traffic.

► **Definition 3** (domination). *An action protocol P dominates action protocol P' with respect to a context γ if for all pairs of corresponding runs $r \in \mathcal{I}_{\gamma, P}$ and $r' \in \mathcal{I}_{\gamma, P'}$, all $i \in \text{Ag}$, we have $\text{gotime}(r, i) \leq \text{gotime}(r', i)$. If P dominates P' but P' does not dominate P , then P strictly dominates P' .*

► **Definition 4** (optimality). *An intersection protocol P is optimal with respect to an intersection context γ if there is no intersection protocol P' that strictly dominates P with respect to γ .*

Our goal is to connect the notions of unnecessary waiting and optimality. The following result shows that the absence of unnecessary waiting is sufficient for optimality.

► **Proposition 4.** *If an intersection protocol P has no unnecessary waiting with respect to an intersection context γ then P is optimal with respect to γ .*

From here on, we consider contexts that require some conditions on broadcasting. This is because if not enough information is exchanged or adversaries are too powerful, we cannot have a protocol that avoids unnecessary waiting. To see why, consider a setting where the intersection has two incoming lanes and one outgoing lane, each agent has access to a global clock, and the information-exchange protocol does not send any messages. While a correct protocol exists that uses the global clock to determine when an agent at the front of a queue can proceed to the intersection (essentially, we use the global clock to simulate a traffic light, and have the agents proceed in turns), unnecessary waiting cannot be eliminated, simply because the agents do not exchange enough information to rule out safety violations.

However, even with full information exchange where each agent broadcasts its entire local state in each round and records every broadcast it receives, the converse of Proposition 4 still does not hold. A protocol may have unnecessary waiting and still be optimal even with full information exchange.

► **Proposition 4.** *There exists an intersection context γ with full information exchange and no failures and an intersection protocol P such that P has unnecessary waiting and is optimal with respect to γ .*

Proposition 4 suggests that the definition of optimality doesn't exactly capture the lack of unnecessary waiting. We thus consider another definition that we call *lexicographic optimality*.

► **Definition 5** (lexicographical domination). *An action protocol P lexicographically dominates action protocol P' with respect to a context γ if for all corresponding runs $r \in \mathcal{I}_{\gamma, P}$ and $r' \in \mathcal{I}_{\gamma, P'}$, either $GO(r, m) = GO(r', m)$ for all times m or, at the first time m when $GO(r, m) \neq GO(r', m)$, we have $GO(r', m) \subsetneq GO(r, m)$. If P lexicographically dominates P' but P' does not lexicographically dominate P , then P strictly lexicographically dominates P' .*

► **Definition 6** (lexicographic optimality). *An intersection protocol P is lexicographically optimal with respect to an intersection context γ if there is no intersection protocol P' that strictly lexicographically dominates P with respect to γ .*

► **Proposition 6.** *If an intersection protocol P has no unnecessary waiting with respect to an intersection context γ , then P is lexicographically optimal with respect to γ .*

The following result provides a partial converse to Proposition 6.

► **Proposition 6.** *If an intersection protocol P is lexicographically optimal with respect to an information context γ with full information exchange and no failures, then P has no unnecessary waiting with respect to γ .*

While considering a full-information context shows that lexicographic optimality captures the condition on unnecessary waiting better, it is also possible to get a similar result in a context with much less information exchange, even without a global clock.

We say that an intersection context $\gamma = (\mathcal{E}, \mathcal{F}, \pi)$ is *sufficiently rich* if \mathcal{E} satisfies the following conditions:

- In round m , if agent i is going to be at the front of some lane at time m , then i broadcasts a message encoding $lane_i, intent_i$. (Note that we are here using the fact that in agent's message in round m can incorporate the effect of its round m action. Thus, if an agent i moves to the front of the queue for some lane in round m , then i will sense that it is at the front of the queue, and i can send a message in round m saying that it is about to be at the head of the queue for its lane.)
- Each agent records the $(lane, intent)$ pair for each agent in the front of a queue, and either no agents in the queue other than those at the front broadcast, or agents at the front of a queue tag their messages to indicate that they are at the front of their queue.

Intuitively, if an intersection context is sufficiently rich, in the round m that an agent i reaches the front of the queue for some lane, it knows about all other agents that are in the front of their queues at time m , and knows their intentions (if there are no failures).

► **Lemma 6.** *If γ is a sufficiently rich intersection context with no failures, P is an intersection protocol, and $front_i(r, m) = 1$, then*

$$\mathcal{I}_{\gamma, P} \models \forall l \in \mathcal{L}_{in} (K_i(\exists j \in Ag \exists l' \in \mathcal{L} (front_j \wedge lane_j = l \wedge intent_j = l') \vee K_i(\forall j \in Ag (lane_j \neq l))))).$$

Given a sufficiently rich intersection context γ , all protocols that we care about will depend only on what the agents hear from agents at the front of each queue. We say that an intersection protocol P *depends only on agents in the front of their queues* in intersection

context $\gamma = (\mathcal{E}, NF, \pi)$ if, for all $i \in Ag$, the following condition holds: for all pairs s_i, s'_i of possible local states of agent i drawn from L_i in \mathcal{E} , if $front(r, m) = front(r, m')$, then $P_i(r_i(m)) = P_i(r'_i(m'))$. Note that this condition makes sense only in a sufficiently rich intersection context in the no-failures setting, since otherwise an agent may not know which agents are at the front of their queues, so its protocol cannot depend on this fact.

► **Proposition 6.** *Let γ be a sufficiently rich intersection context with no failures. If an intersection protocol P is lexicographically optimal with respect to γ and P depends only on agents in the front of their queues, then P has no unnecessary waiting with respect to γ .*

7 Intersection policies

Intuitively, an *intersection policy* describes which moves are permitted, as a function of a history describing what happened in the run until that point in time (in particular, the nondeterministic choices that have been made by the adversary up to that moment of time), but excluding details of the agent's local states and protocol.

We will use intersection policies as a tool to design standard protocols that solve the intersection problem. Roughly, the methodology is the following. Initially, we will design an intersection policy σ that guarantees safety and liveness for agents complying with σ . We will then find standard intersection protocols that implement a knowledge-based program using σ . Finally, we will show that every intersection protocol can be obtained in this way.

A history captures the nondeterministic choices made by the adversary up to some moment of time. Given an adversary $\alpha = (\tau, T, F_t, F_r)$ for a context γ and natural number $m \in \mathbb{N}$, define the *choices of α in round $m+1$* to be the tuple $\alpha_m = (\tau^m, T, F_r^m, F_t^m)$, where $\tau^m = \{(i, \ell, \ell') \in Ag \times \mathcal{L}_{in} \times \mathcal{L}_{out} \mid \tau(i) = (m+1, \ell, \ell')\}$, and for $a = r$ and $a = t$, the function $F_a^m : Ag \rightarrow \{0, 1\}$ is defined by $F_a^m(i) = F_a(m, i)$. (Recall that the transmission environment T is fixed for the run, so the same T applies in each round.) An *adversary history* is a finite sequence of such tuples; for an adversary α and time m , define $H(\alpha, m) = \langle \alpha_0, \dots, \alpha_{m-1} \rangle$. (If $m = 0$, $H(\alpha, m)$ is the empty sequence.) Given a context γ , \mathcal{H}_γ is the set of all adversary histories $H(\alpha, m)$ such that α is an adversary for γ and $m \geq 0$. If r is a run of context γ with adversary α , we also write $H(r, m)$ for $H(\alpha, m)$.

► **Definition 7 (intersection policy).** *An intersection policy for a context γ is a mapping $\sigma : \mathcal{H}_\gamma \rightarrow \mathcal{P}(\mathcal{L}_{in} \times \mathcal{L}_{out})$.*

Intuitively, an intersection policy says which moves are *permitted* in the given round. An agent at the front of a queue for lane ℓ *may* go if its intent is to make move to lane ℓ' and the move (ℓ, ℓ') is permitted. (However, in contexts with failures, the agent may fail to go because it does not know that its move is permitted.)

An infinite sequence h_0, h_1, \dots is *feasible* in a context γ if there exists an adversary α of γ such that $h_m = H(\alpha, m)$ for all $m \geq 0$. An intersection policy σ for a context γ is *correct* for a context γ if it satisfies the following specification:

- **Conflict-free:** For all histories $h \in \mathcal{H}_\gamma$, and agents $i \neq j$, if $(l_i, l'_i), (l_j, l'_j) \in \sigma(h)$ then $(l_i, l'_i, l_j, l'_j) \in \mathcal{O}$.
- **Fairness:** For all feasible infinite sequences of histories h_0, h_1, h_2, \dots , all moves $(\ell, \ell') \in \mathcal{L}_{in} \times \mathcal{L}_{out}$, and all $m \geq 0$, there exists $m' \geq m$ such that $(\ell, \ell') \in \sigma(h_{m'})$.

Intuitively, an intersection policy σ is conflict-free if σ never permits a conflicting set of moves to occur simultaneously. An intersection policy σ is fair if, in every feasible infinite sequence of histories, σ permits every possible move infinitely often. A context γ is *σ -aware* for an intersection policy σ if, for all protocols P for γ , agents i , lanes $\ell \in \mathcal{L}_{in}$ and $\ell' \in \mathcal{L}_{out}$, we have $\mathcal{I}_{\gamma, P} \models ((\ell, \ell') \in \sigma \wedge lane_i = \ell) \Rightarrow K_i((\ell, \ell') \in \sigma)$.

► **Example 8.** A simple correct intersection policy is a cyclic traffic light. Suppose that the set of all moves $\mathcal{L}_{in} \times \mathcal{L}_{out}$ is partitioned into a collection S_0, \dots, S_{K-1} , such that each set S_k is a compatible set of moves. Then the intersection policy defined on histories h by $\sigma(h) = S_{|h| \bmod K}$ is easily seen to be correct (whatever the context γ). Clearly, every synchronous context is σ -aware for this policy.

► **Example 9.** A more complicated intersection policy is one that prioritizes certain lanes if they contain specific agents (e.g., an ambulance). Suppose that $\mathcal{A} \subseteq Ag$ is a finite set of higher-priority agents. Consider the intersection policy that allows moves given by a cyclic traffic-light policy unless there is an agent in \mathcal{A} that has arrived and is yet to make a move. In that case, the policy runs the traffic-light policy restricted to lanes containing higher-priority agents. This requires considering past moves permitted by the policy and the adversary history to determine the state of the queues. In a context with no failures, synchrony, and a transmission environment such that the presence of a higher-priority agent is known by agents in the front, we get σ -awareness.

Given an intersection policy σ , consider the following knowledge-based program \mathbf{P}^σ :

■ **Program \mathbf{P}_i^σ .**

if $K_i(\text{front}_i \wedge (\text{lane}_i, \text{intent}_i) \in \sigma)$ **then go**
else noop

Here the formula $(\text{lane}_i, \text{intent}_i) \in \sigma$ is satisfied at a point (r, m) if we have $(\text{lane}_i(r, m), \text{intent}_i(r, m)) \in \sigma(H(r, m))$.

An action protocol P implements a knowledge-based program of the form “if $K_i\phi$ then go else noop” in a context γ if, for all points (r, m) of $\mathcal{I}_{\gamma, P}$, we have $P_i(r_i(m)) = \text{go}$ iff $\mathcal{I}_{\gamma, P}(r, m) \models K_i\phi$. (See [8] for the definition for more general program structures.)

We immediately get the following.

► **Proposition 9.** *For every synchronous context γ and intersection policy σ for γ , there exists a behaviorally unique² P implementing the knowledge-based program \mathbf{P}^σ with respect to γ . If σ is a correct intersection policy with respect to γ , then every implementation P of the knowledge-based program \mathbf{P}^σ with respect to γ satisfies safety and validity.*

Proposition 9 provides a way of deriving an intersection protocol from an intersection policy. We can also show that every intersection protocol can be derived from some intersection policy in this way.

► **Proposition 9.** *If P is a protocol satisfying validity and safety then there exists a conflict-free intersection policy σ for γ such that P implements \mathbf{P}^σ with respect to γ .*

► **Definition 10** (efficient intersection policies). *An intersection policy σ for a context γ is efficient if for all points $h \in \mathcal{H}_\gamma$, we have that $\sigma(h)$ is a maximal conflict-free set of moves.*

8 A Knowledge-Based Program with Lexicographically Optimal Implementations

We would like to have a way to derive lexicographically optimal protocols under a range of failure assumptions. Moreover, we want these protocols to be fair to all agents, even if there are agents present that are not. To satisfy these goals, we start with an intersection policy σ

² “Behavioral uniqueness” here means that any two implementations take the same actions at all reachable states, and can differ only on unreachable states.

that can be run by all vehicles, including those without V2V communications equipment. One example of such σ is the traffic light policy σ^{TL} . In all cases, moves permitted by this policy will have priority, but we allow vehicles to violate the policy provided that they know that they can do so safely. To avoid clashes, we establish a priority order on the violations. Let $next$ be a function from histories such that $next(h) \in \mathcal{L}_{in}$ for each history h . Intuitively, the agent at the front of the queue for lane $next(h)$ will get precedence in going through the intersection at the point (r, m) . The context γ is *next-aware* if, for all protocols P for γ and agents i and $\ell \in \mathcal{L}_{in}$, we have that $\mathcal{I}_{\gamma, P} \models next = \ell \Rightarrow K_i(next = \ell)$.

Consider the following knowledge-based program \mathbf{P} , where V_i is the proposition

$(lane_i, intent_i) \notin \sigma$ and the move $(lane_i, intent_i)$ is compatible with (a) all moves $(lane_j, intent_j) \in \sigma$ where j is an agent who is about to enter the intersection (i.e., $going_j$ holds) (b) all moves $(lane_j, intent_j) \notin \sigma$ where $j \neq i$ is an agent for which $going_j$ holds and $lane_j \in [next, lane_i]$. (Here $[next, lane_i]$ is the set of lanes from $next(r, m)$ to $lane_i \pmod{|\mathcal{L}_{in}|}$.)

■ **Program \mathbf{P}_i .**

if $K_i(front_i \wedge ((lane_i, intent_i) \in \sigma \vee V_i))$ **then go**
else noop

Intuitively, this knowledge-based program allows all agents permitted by σ to go to do so, as well as allowing agents not permitted by σ to go, provided they do so in a cyclic priority order, and each agent that goes knows that its move is compatible with the moves of all agents of higher priority (including agents permitted to go by σ).

► **Proposition 10.** *Let σ be a conflict-free intersection policy. If context γ is synchronous, next-aware, and σ -aware, then there exists a unique implementation P of \mathbf{P} that satisfies safety and validity, is lexicographically optimal with respect to γ , and lexicographically dominates the unique implementation of \mathbf{P}^σ . Moreover, if σ is fair then P satisfies liveness.*

We can also obtain liveness of the implementations of \mathbf{P} under some other conditions. Define the function $next$ to be *fair* if, for all feasible sequences of histories h_0, h_1, \dots , all $m \geq 0$ and all lanes $\ell \in \mathcal{L}_{in}$, there exists $m' \geq m$ such that $next(h_{m'}) = \ell$. Intuitively, fairness of $next$ will ensure that $next$ fairly selects the first agent that can violate the intersection policy according to \mathbf{P} when this can be done safely.

We also need to ensure that it is not the case that σ always gives priority to other lanes whenever the lane ℓ is selected by $next$. For this, define a pair $(\sigma, next)$, consisting of an intersection policy σ and a function $next$, to be *fair* if for all feasible sequences of histories h_0, h_1, \dots , all $m \geq 0$ and all moves $(\ell, \ell') \in \mathcal{L}_{in}$, there exists $m' \geq m$ such that either $(\ell, \ell') \in \sigma(h_{m'})$, or $next(h_{m'}) = \ell$ and (ℓ, ℓ') is compatible with all the moves in $\sigma(h_{m'})$.

► **Proposition 10.** *Let P be an implementation of \mathbf{P} with respect to a synchronous, next-aware and σ -aware context. If the pair $(\sigma, next)$ is fair, then P satisfies liveness.*

Note that if $next$ is fair, and the σ_\emptyset is the (unfair) intersection policy defined by $\sigma_\emptyset(h) = \emptyset$ for all histories h , then the pair $(\sigma_\emptyset, next)$ is fair. For examples in which σ is not trivial, consider the following properties of σ . Say that σ is *cyclic* (with cycle length k) if for all histories h and h' with $|h| \equiv |h'| \pmod{k}$, we have $\sigma(h) = \sigma(h')$. Say that σ is *non-excluding* if for all moves (ℓ, ℓ') , there exists a history h such that (ℓ, ℓ') is compatible with all moves in $\sigma(h)$. Given a non-excluding σ with cycle length k , let $next$ be defined by $next(h) = \lfloor |h|/k \rfloor \pmod{k}$. Then $(\sigma, next)$ is fair. This is because the value of $next$ cycles through all values in

\mathcal{L}_{in} , but is held constant through each cycle of σ . Thus, for each move (ℓ, ℓ') , eventually a point in these combined cycles will be reached for which the value of $next$ is ℓ and (ℓ, ℓ') is compatible with all moves permitted by σ .

8.1 Implementing \mathbf{P} when there is no communication

We now consider standard implementations of \mathbf{P} in two particular contexts of interest. Since we would like the implementations to be correct and lexicographically optimal, we use $next$ and σ defined as $next(h) = m \bmod |\mathcal{L}_{in}|$ and $\sigma(h) = \emptyset$ for all histories h of length m . Using this choice of $next$ and σ in the construction of \mathbf{P} ensures that in any synchronous intersection context, both $next$ -awareness and σ -awareness hold; moreover, the pair $(next, \sigma)$ is fair. Therefore, implementations P of \mathbf{P} in such contexts are correct and lexicographically optimal, by Propositions 10 and 10.

We have taken σ to be empty for ease of exposition. For practical implementations, the construction given by the proof of Proposition 10 can be used to get other implementations that prioritize moves permitted by σ . (For example, in an intersection where certain lanes are often busier, moves originating from those lanes can be prioritized.) Note that for empty σ , the condition $K_i(front_i \wedge ((lane_i, intent_i) \in \sigma \vee V_i))$ reduces to $K_i(front_i \wedge V_i')$, where V_i' is the proposition

“the move $(lane_i, intent_i)$ is compatible with all moves $(lane_j, intent_j)$ of agents $j \neq i$ with $lane_j \in [next, lane_i)$ such that $going_j$ ”,

since σ is empty. Consider the following context with no communication. Let γ_\emptyset be a synchronous intersection context where agents do not broadcast messages. Formally, for a failure model \mathcal{F} , we define $\gamma_\emptyset(\mathcal{F}) = (\mathcal{E}_\emptyset, \mathcal{F}, \pi_\emptyset)$, where

- $(\mathcal{E}_\emptyset)_i$ is an information-exchange protocol where the following hold:
 - The set of memory states is a singleton so, effectively, local states consist only of the sensor reading $L_i = \Sigma_i$.
 - No messages are sent, so $M_i = \emptyset$, μ_i is the constant function with value \perp , and δ_i is omitted.
 - The sensor model is defined as in the definition of intersection contexts. The only modification is that the sensor model now maps environment states to tuples of the form $\langle front_i, lane_i, intent_i, time_i \rangle$, where $time_i$ is determined by the time encoded in the environment state.
- π_\emptyset interprets the propositions defined for intersection contexts in the obvious way.

We now define a procedure to compute a set Pos_i of moves that agent i believes may be performed as a function of $next$ and the structure of the intersection represented by \mathcal{O} . We capture stages of the construction of this set as sets of moves Pos_i^l for $l \in [next - 1, lane_i)$. (By $next$ -awareness, $next$ is computable from the agent’s local state. For brevity, we interpret $next - 1$ as $next - 1 \pmod{|\mathcal{L}_{in}|}$.)

1. Start with $Pos_i = Pos_i^{next-1} = \emptyset$
2. For $l \in [next, lane_i)$ do
 - a. Let L be the set of moves (l, l') where $l' \in \mathcal{L}_{out}$ such that (l, l') is compatible with Pos_i , and let $Pos_i^l := Pos_i \cup L$ and $Pos_i := Pos_i^l$.
3. Output Pos_i .

Let P^\emptyset be the standard protocol given by the following program, where move (l, l') is compatible with a set of moves S if it is compatible with all moves in S according to \mathcal{O} .

■ **Program** P_i^0 .

if $front_i \wedge (lane_i, intent_i)$ *is compatible with* Pos_i **then go**
else noop

► **Proposition 10.** P^0 implements \mathbf{P} with respect to $\gamma_\emptyset(\mathcal{F})$ for $\mathcal{F} \in \{NF, CR, SO\}$.

Proposition 10 shows that, without communication, a protocol that essentially implements traffic lights is lexicographically optimal.

8.2 Implementing \mathbf{P} in a context with limited communication

If we allow messages regarding the current lane and agents' intentions by agents that reach the front, this changes how implementations of \mathbf{P} behave. Roughly speaking, in runs where the intersection gets crowded, a much larger set of agents can proceed through the intersection. Let γ_{intent} be a synchronous context with communication failures such that if an agent is in the front of some lane, it broadcasts $(lane, intent)$. (This information exchange broadcasts a lot less information than a full-information exchange.) More formally,³ for a failure model \mathcal{F} , we define $\gamma_{intent}(\mathcal{F}) = (\mathcal{E}_{intent}, \mathcal{F}, \pi_{intent})$, where

- $(\mathcal{E}_{intent})_i$ is defined as an information-exchange protocol where the following hold:
 - The local states maintain a set of moves M in the memory component in addition to the sensor readings. Intuitively, this set represents the set of moves from broadcasts that were received by i in the current round. Note that M_i may not contain i 's move since i 's broadcast may fail.
 - The set of messages is $M_i = \mathcal{L}_{in} \times \mathcal{L}_{out}$, and μ_i broadcasts the message $(lane_i, intent_i)$ by reading $lane_i$ and $intent_i$ from the sensor reading, if $front_i$, and broadcasts no message otherwise. Note that these variable references are from $\mathcal{S}(s'_e)$ where s'_e is the new environment state that the system moves to in the course of the round.
 - The sensor model is defined as in the definition of intersection contexts (while including *time* as a sensor reading as in \mathcal{E}_\emptyset).
 - δ_i maps the set of received messages directly into the memory with i 's own move; that is, $\delta_i(s_i, a, Mes) = Mes$. Note that an agent can determine from this set whether its own broadcast was successful.
- π_{intent} interprets the propositions defined for interpretation contexts in the obvious way.

We now proceed as in Subsection 8.1 and define a procedure to compute from an agent i 's local state $s_i = (M_i, (lane_i, intent_i, time_i))$ a set Pos_i of moves that agent i believes may be performed by higher-priority agents in the next round. We again capture stages of the construction of this set as sets of moves Pos_i^l for $l \in [next - 1, lane_i)$.

1. Start with $Pos_i = Pos_i^{next-1} = \emptyset$
2. For $l \in [next, lane_i)$ do
 - a. If for some $l' \in \mathcal{L}_{out}$, the move (l, l') is in M_i then
 - if (l, l') is compatible with Pos_i
 - then $Pos_i^l := Pos_i \cup \{(l, l')\}$ and $Pos_i := Pos_i^l$
 - else $Pos_i^l := Pos_i$.

³ This context satisfies the *sufficiently rich* condition of Section 6.

- b. Otherwise, let L be the set of moves (l, l') where $l' \in \mathcal{L}_{out}$ such that (l, l') is compatible with Pos_i , and let $Pos_i^l := Pos_i \cup L$ and $Pos_i := Pos_i^l$.⁴
3. Output Pos_i .

Let the output of running this procedure on a local state with memory state M_i be denoted by Pos_i , and let P^{intent} be the standard protocol defined using the following program:

■ **Program** P_i^{intent} .

if $front_i \wedge (lane_i, intent_i)$ *is compatible with* Pos_i **then go**
else noop

► **Proposition 10.** P^{intent} implements \mathbf{P} with respect to $\gamma_{intent}(\mathcal{F})$ for $\mathcal{F} \in \{CR, SO\}$.

Again, by Proposition 10, it follows that the intersection protocol P^{intent} is lexicographically optimal with respect to the contexts $\gamma_{intent}(\mathcal{F})$ for $\mathcal{F} \in \{CR, SO\}$.

9 Discussion

We introduced the *intersection problem*, identified the appropriate notion of optimality called *lexicographical optimality*, and designed protocols that are optimal in a variety of contexts. A knowledge-based analysis and the use of *intersection policies* were crucial in this process.

Previous work has considered many models ranging from computing individual trajectories of vehicles to relying on centralized schedulers [6]. In [16, 15], a four-way intersection is considered in a context with failures. [10, 17] consider *virtual traffic lights*; the approach is evaluated using a large-scale simulation. [9] solves the same problem probabilistically, in contexts with failures. Work in the control theory literature has focused on vehicle dynamics when going through an intersection [11] to avoid collision. Efforts have also been made to build distributed intersection management systems through V2V communication [5].

While there has been considerable effort in designing protocols for specific intersections or designing architectures for intersection management systems, we aim to develop a context- and architecture-independent approach. Our goal in this paper is to lay the theoretical foundations of optimal intersection protocol design in a variety of contexts, including contexts with failures. We do so abstractly by defining the model to capture any intersection topology with minimal requirements on V2V communication range. While the protocols we design do not require sensors such as lidar and radar, the use of a knowledge-based program \mathbf{P} provides a direct method to develop optimal implementations in contexts with extra sensors.

The problem we study in this paper can be viewed as a generalization of the classical problem of *mutual exclusion*, which requires that two distinct agents are not simultaneously in a *critical section* of their code. Indeed, a variant of mutual exclusion called *group mutual exclusion* [12] is strictly weaker than the intersection problem. In group mutual exclusion, each process is assigned a session when entering the critical section and processes are allowed to enter the critical section simultaneously provided that they share the same session. If agents form an equivalence relation based on their move compatibility according to \mathcal{O} , we can identify each equivalence class to be in the same session and think of the intersection as the critical section. However, our setting differs in some critical ways:

⁴ Intuitively, since M is the set of moves that i hears about from agents in the front of some lane, in this case i did not hear from anyone in lane l . However, in settings with sending omissions, there may nevertheless be an agent at the front of lane l' . Such an agent will move only if it can do so safely.

- Intersections often have an \mathcal{O} relation that is not an equivalence relation. For instance, the fact that agents' moves conflict in lanes A-B and in lanes B-C does not imply that their moves in lanes A and C conflict (e.g., if agents want to move straight in a four-way intersection with two lanes in each direction).
- We take the set Ag of agents to be unbounded, while group mutual exclusion (and equivalent problems such as *room synchronization* [3]) consider a bounded number of agents.
- Our agents arrive according to a (possibly infinite) schedule determined by the adversary.
- To the best of our knowledge, fault-tolerance has not been considered in the group mutual-exclusion setting.

The mutual-exclusion problem is generally studied with respect to an interleaving model of asynchronous computation, but as Lamport [13] noted, this model is not physically realistic, and already builds in a notion of mutual exclusion between the actions of distinct agents. The *Bakery* mutual-exclusion protocol [13] is correct with respect to models allowing simultaneous read and write operations. Moses and Patkin [14] develop an improvement of Lamport's Bakery algorithm for the mutual-exclusion problem using a knowledge-based analysis, noting that there are situations in which Lamport's protocol could enter the critical section, but fails to do so. A weaker knowledge-based condition for mutual exclusion is used by Bonollo et al. [4]; it states that an agent i may enter its critical section when it knows that no other agent will enter its critical section until agent i has exited from its critical section. Clearly these knowledge-based approaches are similar in spirit to ours. We hope to study the exact relationship between these problems in the near future.

There are several directions that we hope to explore in the future. One involves extending the current results to contexts with stronger adversaries and evaluating implementations of \mathbf{P} in other contexts. Another is considering strategic agents, who may deviate from a protocol to cross the intersection earlier.

References

- 1 K. Alpturer, J. Y. Halpern, and R. van der Meyden. Optimal eventual Byzantine agreement protocols with omission failures. In *Proc. 42nd ACM Symposium on Principles of Distributed Computing*, pages 244–252, 2023.
- 2 K. Alpturer, J. Y. Halpern, and R. van der Meyden. A knowledge-based analysis of intersection protocols, 2024. [arXiv:2408.09499](https://arxiv.org/abs/2408.09499).
- 3 Guy E. Blelloch, Perry Cheng, and Phillip B. Gibbons. Room synchronizations. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, SPAA '01, pages 122–133, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/378580.378605.
- 4 U. Bonollo, R. van der Meyden, and E.A. Sonenberg. Knowledge-based specification: Investigating distributed mutual exclusion. In *Bar Ilan Symposium on Foundations of AI*, 2001. URL: <https://www.cse.unsw.edu.au/~meyden/research/bisfai.pdf>.
- 5 António Casimiro, Jörg Kaiser, Elad M. Schiller, Pedro Costa, José Parizi, Rolf Johansson, and Renato Librino. The karyon project: Predictable and safe coordination in cooperative vehicular systems. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12, 2013. doi:10.1109/DSNW.2013.6615530.
- 6 Lei Chen and Cristofer Englund. Cooperative intersection management: A survey. *Trans. Intell. Transport. Syst.*, 17(2):570–586, January 2016. doi:10.1109/TITS.2015.2471812.
- 7 K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of A.I. Research*, 31:591–656, 2008. doi:10.1613/JAIR.2502.

- 8 R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, 1995. A slightly revised paperback version was published in 2003.
- 9 N. Fathollahnejad, E. Villani, R. Pathan, R. Barbosa, and J. Karlsson. On reliability analysis of leader election protocols for virtual traffic lights. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12, 2013.
- 10 M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz. Self-organized traffic control. In *Proceedings of the Seventh ACM International Workshop on Vehicular InterNetworking*, pages 85–90, 2010.
- 11 Michael R. Hafner, Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. Cooperative collision avoidance at intersections: Algorithms and experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1162–1175, 2013. doi:10.1109/TITS.2013.2252901.
- 12 Yuh-Jzer Joung. Asynchronous group mutual exclusion. *Distributed Computing*, 13(4):189–206, November 2000. doi:10.1007/PL00008918.
- 13 Leslie Lamport. A new solution of Dijkstra’s concurrent programming problem. *Commun. ACM*, 17(8):453–455, 1974. doi:10.1145/361082.361093.
- 14 Yoram Moses and Katia Patkin. Mutual exclusion as a matter of priority. *Theor. Comput. Sci.*, 751:46–60, 2018. doi:10.1016/j.tcs.2016.12.015.
- 15 E. Regnath, M. Birkner, and S. Steinhorst. CISCAY: consensus-based intersection scheduling for connected autonomous vehicles. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pages 1–7, 2021.
- 16 V. Savic, E. M. Schiller, and M. Papatriantafidou. Distributed algorithm for collision avoidance at road intersections in the presence of communication failures. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1005–1012, 2017.
- 17 Rusheng Zhang, Frank Schmutz, Kyle Gerard, Aurélicn Pomini, Louis Basseto, Sami Ben Hassen, Akihiro Ishikawa, Inci Ozgunes, and Ozan Tonguz. Virtual traffic lights: System design and implementation. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018. doi:10.1109/VTCFall.2018.8690709.