Compositional Safety Rules for Inter-Triggering Hybrid Automata

Kwesi J. Rutledge∗
Univ. of Michigan, Ann Arbor, USA
krutledg@umich.edu

Glen Chou∗
Univ. of Michigan, Ann Arbor, USA
gchou@umich.edu

Necmiye Ozay
Univ. of Michigan, Ann Arbor, USA
necmiye@umich.edu

ABSTRACT
In this paper, we present a compositional condition for ensuring safety of a collection of interacting systems modeled by inter-triggering hybrid automata (ITHA). ITHA is a modeling formalism for representing multi-agent systems in which each agent is governed by individual dynamics but can also interact with other agents through triggering actions. These triggering actions result in a jump/reset in the state of other agents according to a global resolution function. A sufficient condition for safety of the collection, inspired by responsibility-sensitive safety, is developed in two parts: self-safety relating to the individual dynamics, and responsibility relating to the triggering actions. The condition relies on having an over-approximation method for the resolution function. We further show how such over-approximations can be obtained and improved via communication. We use two examples, a job scheduling task on parallel processors and a highway driving example, throughout the paper to illustrate the concepts. Finally, we provide a comprehensive evaluation on how the proposed condition can be leveraged for several multi-agent control and supervision examples.

ACM Reference Format:
https://doi.org/10.1145/3447928.3456659

1 INTRODUCTION
Proving safety or designing controllers guaranteeing safety in a multi-agent setting is a challenging task for multiple reasons. On one extreme, one can try to come up with a monolithic safety rule, which may be hard to verify due to scalability issues, and hard to follow at run-time without a central coordinator. On the other extreme, one can analyze agents individually, assuming all of the remaining agents act adversarially, in which case safety is hard to attain, if at all possible. Several frameworks have been developed between these two extremes to capture various notions of coordination, collaboration, or contracts [5, 6, 9, 14, 17, 23, 25, 28],

∗Both authors contributed equally to the paper. This work is supported in part by ONR grant # N00014-18-1-2501, NSF grants #1553873 and #1918123. KJR is also supported by an NSF graduate research fellowship and GC is supported by an NDSEG fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

HSCC ’21, May 19–21, 2021, Nashville, TN, USA.
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8339-4/21/05...
https://doi.org/10.1145/3447928.3456659

An alternative viewpoint is presented by responsibility-sensitive safety [7, 11, 24, 27], in the context of autonomous driving, where some hard safety constraints are replaced by a notion of not causing a crash and avoiding one whenever possible. This is particularly well-suited for scenarios where some of the agents are human-controlled, which can lead to unpredictable behaviors. From this viewpoint, we expect autonomous agents to act in a well-behaved fashion as long as others reciprocate, and we should not punish the autonomous agents for failures that are out of their control.

In this paper, we consider multi-agent systems modeled by inter-triggering hybrid automata (ITHA), a modeling framework for interacting parallel processes and multi-agent systems [20]. Inter-triggering hybrid automata consist of a collection of agents, where each agent is modeled with a discrete-time dynamical system locally. In addition to their local dynamics, agents are also equipped with triggering actions as a means to interact with other agents in the collection. In particular, these triggering actions can collectively induce a reset (i.e., jump) in the dynamics of other agents. For this class of systems, we propose a simple two part condition for each agent to follow that is shown to be sufficient to guarantee safety of the overall collection. The two parts pertain to self-safety (in the local dynamics) and responsibility (in the interactions), jointly called responsibility-sensitive safety. Controlled invariant sets for the individual dynamics [3] are used for both parts of the condition. Intuitively, each agent aims to remain in their corresponding invariant set, and when they use a triggering action that affects another agent, they do so in a “responsible way”, by trying to ensure that the other agent’s state does not leave its invariant set due to this action. The responsibility-sensitive safety conditions, being based on local invariant sets, enable us to use the same conditions either for proving the safety of other policies, for supervising existing policies, or for designing new control policies for guaranteed safety.

A central component of the inter-triggering hybrid automaton is what we call a global resolution function, which determines the reset induced on a given agent based on all agent’s triggering actions. This function’s value cannot be known, in general, to any of the agents at run-time. This is because the value of the global resolution function depends on the triggering actions chosen by all agents. To overcome this issue, we introduce over-approximations of the resolution functions that can be used within the responsibility rule. We also show how individual agents may compute such over-approximations, and how the conservativeness in this computation can be reduced if (local) communication between agents is allowed.

We apply our proposed framework on several multi-agent control problems, including task coordination for parallel processing on a server farm and navigation for autonomous highway driving, empirically verifying the safety guarantees of our method and also demonstrating how different over-approximations can lead to
The vertex disturbances, and \( f \) of a vertex connections mean that the nature of directed graphs, \( v \) in invariant set notation which will be used throughout the paper.

We summarize our contributions as:

- We refine the inter-triggering hybrid automaton model: a flexible modeling formalism for representing multi-agent interactions, vastly expanding on the initial ideas in [20].
- We design compositional conditions which are sufficient for global safety, provided that each agent ensures it remains self-safe and responsible with respect to the agents it triggers.
- For tractability, we provide practical over-approximations of the collective triggering behavior and show how they can be made less conservative with local communication.
- We perform a comprehensive evaluation of our method on a variety of multi-agent control and supervision tasks.

2 PRELIMINARIES

In this section, we introduce the graph, dynamical system, and invariant set notation which will be used throughout the paper.

2.1 Graph notation

A directed graph \( G = (V,E) \) is a tuple containing a set of vertices \( V \) and a set of directed edges \( E \). Note that each edge is an ordered pair \( (v_1, v_2) \in E \) of vertices from \( V \) (i.e. \( v_1, v_2 \in V \)). We say that a vertex \( v_1 \) is connected to a vertex \( v_2 \) if and only if \( (v_1, v_2) \in E \). By the nature of directed graphs, \( v_1 \) being connected to \( v_2 \) does not mean that \( v_2 \) is connected to \( v_1 \). Within directed graphs the inward connections of a vertex \( v \) can be defined as follows:

\[
\text{in}_{G}(v) = \{ v' \in V \mid (v',v) \in E \}.
\]

The outward connections of a vertex \( v \) can be defined similarly:

\[
\text{out}_{G}(v) = \{ v' \in V \mid (v,v') \in E \}.
\]

2.2 Invariant sets

Consider a discrete-time dynamical system \( \Sigma : (X, U, D, f) \) where \( X \) is the state space, \( U \) is the set of control inputs, \( D \) is the set of disturbances, and \( f : X \times U \times D \rightarrow X \) is the state update function. The state of the system \( \Sigma \) evolves according to

\[
x(t+1) = f(x(t), u(t), d(t)).
\]

Controlled invariant sets play an important role in ensuring safety of systems with dynamics of the form (1). Formally, a robust controlled invariant set \( C_{\text{inv}} \) inside a given safe set \( X_{\text{safe}} \subseteq X \) (i.e., \( C_{\text{inv}} \subseteq X_{\text{safe}} \)) is a set of states that satisfies:

\[
x \in C_{\text{inv}} \Rightarrow \exists u \in U \ \forall d \in D \ f(x, u, d) \in C_{\text{inv}}.
\]

In words, this means that if the state \( x(t) \) is in \( C_{\text{inv}} \), there is an input \( u(t) \) to ensure that \( x(t+1) \) will be in \( C_{\text{inv}} \), for any disturbance within given bounds, thus allowing the states to stay in \( C_{\text{inv}} \) indefinitely.

3 A MODELING FORMALISM FOR INTERACTING SYSTEMS

We introduce inter-triggering hybrid automata (ITHA), a hybrid modeling formalism for collections of discrete-time hybrid systems with a special form of interaction between them. In particular, these interactions are such that they induce jumps or resets on the state evolution of individual agents.

DEFINITION 1. An inter-triggering hybrid automaton is a collection \( \{ \mathcal{H}_i \}_{i \in I} \) of systems together with a function \( \rho = (\rho_1, \ldots, \rho_{|I|}) \), which we refer to as a resolution function, with each \( \mathcal{H}_i \), i.e., agent \( i \), being a hybrid automaton of the form \( \mathcal{H}_i = (X_i, T_i, R_i) \), where:

- \( \Sigma_i = (X_i; U_i, D_i, f_i) \) are the individual dynamics for agent \( i \);
- \( T_i \) is the set of triggering actions of agent \( i \), including a null triggering action \( e \in T_i \) that indicates that agent \( i \) is not triggering a reset on any other agent;
- \( R_i : X_i \times T_i \times [2, \infty) \rightarrow 2^{X_i} \) is the (potentially time-varying) reset map for agent \( i \);
- and where each \( \rho_i : X_i \times T_i \times \cdots \times T_{|I|} \rightarrow 2^{X_i} \) is the resolution function of agent \( i \) that takes the triggering inputs of the entire collection and determines the set of agents that trigger a reset on agent \( i \).

For notational simplicity when referring to an ITHA, we will omit the resolution function and simply say "an ITHA \( \{ \mathcal{H}_i \}_{i \in I} \)". We assume the resolution function \( \rho \) satisfies the following property, which essentially says if agent \( i \) is using a null triggering action at a given time, it will not appear in the output of the resolution function of any agents at that time.

ASSUMPTION 1. For all \( i \), for all \( t \) such that \( \tau_i(t) = e \), we have \( i \notin \rho_j(t, \ldots, \tau_i(t), \ldots) \) for all \( j \), which implies, as a special case, \( \rho_i(t, e, e, \ldots) = \emptyset \).

DEFINITION 2 (Execution of an ITHA). Given sequences of control inputs \( u_i = u_i(0), u_i(1), \ldots \) and triggering inputs \( \tau_i = \tau_i(0), \tau_i(1) \ldots \) for each agent \( i \), an execution of \( \{ \mathcal{H}_i \}_{i \in I} \) is a collection of sequences \( \{ \epsilon_i \}_{i \in I} \), each a sequence of alternating states and actions \( \epsilon_i = x_i(0), u_i(0), \tau_i(0), x_i(1), u_i(1), \tau_i(1), \ldots \) such that:

\[
x_i(0) \in X_i \quad \forall i \in I \quad (3a)
\]

\[
(u_i(t), \tau_i(t)) \in U_i \times T_i \quad \forall i \in I, \forall t \geq 0 \quad (3b)
\]

\[
x_i(t+1) =
\begin{cases}
  f_i(x_i(t), u_i(t), D_i) & \text{if } \rho_i(t) = 0, \\
  R_i(t, x_i(t), u_i(t), \rho_i(t)) & \text{otherwise.}
\end{cases}
\]

where \( \rho_i(t) \triangleq \rho_i(t, \tau_i(t), \tau_i(t), \ldots, \tau_{|I|}(t)) \).

1With a slight abuse of notation, the last argument of the reset map is shown as an index set, but the actual reset value depends on the state, input, and triggering action of the agents in that set.
For an element $e_i$ of an execution, we denote the corresponding state trajectory by $x_i = x_i(0), x_i(1), \ldots$. We consider problems related to safety of an execution of an ITA, where we require the state trajectory of each agent $\mathcal{H}_i$ to remain in a safe set $X_i_{\text{safe}} \subseteq \mathcal{X}_i$ for all times. We use an execution $\{e_i\}_{i \in I}$ remaining in a collection of sets $\{X_i_{\text{safe}}\}_{i \in I}$ inter-changeably with the corresponding state trajectories $\{x_i\}_{i \in I}$ remaining in the same collection.

To make the definition of inter-triggering hybrid automaton concrete, we present two examples used throughout the paper.

**Example 1 (Parallel Processors on a Server Farm).** A collection of processors in a server farm can be treated as a collection of agents where each agent’s state is the number of jobs it has to compute. In other words, agent $i$’s state $x_i \in \mathbb{N}$, where there is a limit of jobs, $n_{\text{overflow}}$, over which the processor will create a stack overflow and fail. External jobs $d_i$ for processor $i$ are passed into the server according to a protocol that blocks new jobs from coming in if $x_i \geq n_{\text{throttle}}$ where $n_{\text{throttle}} < n_{\text{overflow}}$ and the processor always can take the action to address a job in its queue or do nothing. Thus, the individual dynamics can be visualized as shown in Fig. 2. In addition, the processors can be recruited by other processors according to a directed graph $\mathcal{G}$ that indicates which processors can send jobs to which other processors, i.e., processor $i$ can recruit processor $j$ if $(i, j) \in E$. An example of such a graph is shown in Fig. 1. This scenario can be modeled with the representation $\{\mathcal{H}_i\}_{i \in I}$, where each agent $\mathcal{H}_i = (\Sigma_i, T_i, R_i)$ is composed of the following parts:

- **Dynamics** $\Sigma_i$ where $X_i = \mathbb{N}$ is the queue of jobs to be done by agent $i$; $U_i = \{0, -1\}$ represents processor $i$’s choice to do nothing (i.e., $u_i(t) = 0$) or to address one of the jobs in its queue (i.e., $u_i(t) = -1$); $D_i = \{0, 1, 2\}$ represents the number of external jobs passed into processor $i$, and $f_i$ given by
  \[
  f_i(x_i, u_i, d_i) = \begin{cases} 
  x_i + u_i & \text{if } x_i \geq n_{\text{throttle}} \\
  x_i + u_i + d_i & \text{otherwise},
  \end{cases}
  \]
  describes how the queue of jobs is changing for agent $i$ in the absence of it being recruited;

- **Initial Condition** $T_i = 2^{x_i(u_i)}(i)$ represents the possible sets of agents that agent $i$ recruits to help it with its queue, with null element $e = \emptyset$, according to its outgoing edges in $\mathcal{G}$;

- **Reset Map** $R_i$ describes how the queue for agent $i$ changes if it was recruited by or recruited another agent. To each agent in the recruit set (i.e., $\forall j \in T_i(t) \in T_i$), agent $i$ sends 1 job from its queue to that processor:
  \[
  R_i(t, x_i(t), u_i(t), S) = \begin{cases} 
  0 & \text{if } s_i(t) \leq 0 \\
  s_i(t) & \text{otherwise},
  \end{cases}
  \]
  where $s_i(t) = x_i(t) - |r_i(t)| + u_i(t) + |S| + d_i(t)$, where $S$ is the set of agents recruiting agent $i$.

Furthermore, we can write the $i^{th}$ component of the resolution function $\rho_i(t) = \{ j \in I \mid i \in T_j(t) \}$; that is, the set of agents triggering agent $i$ at time $t$ is the set of agents which contain $i$ in its triggering action at time $t$. Concretely, for agent $i$ in Fig. 1, $\rho_i(t) \subseteq \{2, 4\}$, for all $t$. If $r_2(t) = e$ and $r_4(t) = \{1, 6\}$, then $\rho_1(t) = \{4\}$, regardless of the triggering actions of the remaining agents.

Note that each processor can avoid the overflow states indefinitely if it is within a robust control invariant set that is completely contained in the safe set $\{x \in \mathbb{N} \mid x < n_{\text{overflow}}\}$. Under the individual dynamics, the maximal control invariant set in the safe set (i.e., $\mathcal{C} \subseteq \{x \in \mathbb{N} \mid x < n_{\text{overflow}}\}$) can be quickly shown to be $\mathcal{C} = \{x \in \mathbb{N} \mid x \leq n_{\text{overflow}} - 1\}$. Depending on the objectives of the processors, i.e., maximizing throughput, each processor may need to trigger other agents, and without adequate precaution the triggering can reset the states of some processors above $n_{\text{overflow}}$, leading to unsafe behavior.

**Example 2 (Highway Driving).** Consider a collection of vehicles travelling in the same direction on a highway (see Fig. 3). This collection can be represented by an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i \in I}$ where each agent $\mathcal{H}_i = (\Sigma_i, T_i, R_i)$ is composed of:

- **Dynamics** $\Sigma_i$ with $X_i = [0, v_{\text{max}}] \times [0, \infty] \times [0, v_{\text{max}}]$, where state $x_i = [v_i, h_i, d_i]^T$ contains $v_i$ (velocity of current agent $i$, henceforth referred to as the ego vehicle), $h_i$ (headway between this agent and the nearest car in front of it on the same lane, henceforth referred to as the lead vehicle), and $d_i$ (the velocity of the lead vehicle), $U_i$ is the set of allowed inputs, with input $u_i = a_i$ being the acceleration of the ego car, $D_i$ is the set of allowable disturbances, with disturbance $d_i = a_i^+$ being the acceleration of the lead vehicle, and the system dynamics $f_i : X_i \times U_i \times D_i \rightarrow X_i$ is such that
  \[
  f_i(x_i, u_i, d_i) = \begin{bmatrix} 0 & 0 & 0 \\
  -\Delta t & 1 & 0 \\
  0 & 0 & 1
  \end{bmatrix} x_i + \begin{bmatrix} 0 \\
  \Delta t \\
  0
  \end{bmatrix} u_i + \begin{bmatrix} 0 \\
  0 \\
  0
  \end{bmatrix} d_i
  \]
  where $\Delta t$ is the sampling time;

- **Initial Condition** $T_i = \{\text{stay, left, right}\}$ is the set of possible lane change decisions, with null element $e = \text{stay}$.

- A reset $R_i(t, x_i(t), u_i(t), \rho_i(t))$ is triggered on agent $i$ if one of the following happens (1) $r_i(t) \neq e$, (2) if their lead car is $j$ and $\tau_j(t) \neq e$ and/or (3) a car becomes the current agent’s lead car (i.e., lead car was $j$ at time $t - 1$ but the lead car becomes $k \neq j$ at time $t$). The value of the reset depends on the triggering actions of all agents $\rho_i(t)$, which can possibly affect agent $i$ at time $t$. This determines the new lead car and hence the new values of $h_i$ and $d_i$, while $v_i$ evolves according to individual dynamics in (5).

Controlled invariant sets for the system defined in (5) can be computed using polyhedral set computation methods such as those discussed in [18, 25]. An example of such a set is shown in Fig. 4.

To further illustrate how the reset map is defined, if a lead change happens, the value of $R_2$ will be explained for some vehicles in Fig. 3. First, suppose that the current time is $t$ and vehicles $E$ and $F_2$ have states $x_E = [v_E, h_E, d_E]^T$ and $x_{F_2} = [v_{F_2}, h_{F_2}, d_{F_2}]^T$, respectively, and apply control inputs $u_E$ and $u_{F_2}$. Suppose that the ego vehicle makes a left lane change at time $t$; this action triggers a reset of its own continuous state as well as that of $F_2$ (leading to changes in headway and lead car velocity for both vehicles). Formally, if $r_2(t) = \{\text{left}\}$, then

\[
R_E(t, x_E, u_E, \{E\}) = \begin{bmatrix} v_E + \Delta t u_E \\
\infty \\
v_{\text{max}}
\end{bmatrix}
\]

and

\[
R_{F_2}(t, x_{F_2}, u_{F_2}, \{E\}) = \begin{bmatrix} v_{F_2} + \Delta t u_{F_2} \\
\Delta t h_{F_2} \left( (v_E - v_{F_2}) \Delta t \right) \\
v_{\text{max}} + \Delta t u_{F_2}
\end{bmatrix}
\]
This section shows that safety can be guaranteed if robust controlled invariant sets are found for the inter-triggering structure, we will deduce the sets \( \mathcal{E}_i \) and \( \mathcal{L}_i \) in time. Moreover, instead of working with a fixed observation access to in order to be able to comply with the conditions. Any architecture defining the information flow in a distributed synthesis problem can be captured by choosing some conditions on local controllers under which global safety is guaranteed is a distributed synthesis problem \([21]\). Verifying existence of such controllers is known to be undecidable in general even when the sets \( \mathcal{E}_i, \mathcal{U}_i, \mathcal{D}_i \) are finite \([4, 8]\). Any architecture defining the information flow in a distributed synthesis problem can be captured by choosing some possible observations of agent \( i \), a memoryless local controller (or, control policy) is a function \( \gamma_i : Y_i \rightarrow U_i \times T_i \). Similarly, a local controller with memory is a function \( \gamma_i : Y_i^+ \rightarrow U_i \times T_i \), where the superscript \( + \) denotes finite non-zero repetition. If agent \( i \)'s decisions only depend on its own state or the state of all agents, we have \( Y_i = \{X_i\} \) or \( Y_i = X_1 \times \cdots \times X_{|\mathcal{I}|} \), respectively. Also, if an agent can access a (potentially time-varying) subset of other agents’ states, we have \( Y_i = \bigcup_{j \in \mathcal{I}} \{X_j\} \). In addition to states, \( Y_i \) can incorporate observations of actions of the other agents, which would be relevant when introducing the communication scheme in section 4.3.2.

**Definition 3 (Controlled Execution of an ITIA)**. Given a collection of controllers \( \{\gamma_i\}_{i \in \mathcal{I}}, \{\gamma_i\}_{i \in \mathcal{I}}' \)-controlled executions of \( \{\mathcal{H}_i\}_{i \in \mathcal{I}} \) are the set of executions where control inputs \( u_i \) and triggering inputs \( \tau_i \) are produced according to the function \( \gamma_i \) for all \( i \).

Given a collection of local safe sets \( \{X_{i, safe}\}_{i \in \mathcal{I}} \) and information \( Y_i \) available to each agent, synthesizing local controllers for each agent to guarantee global safety is a distributed synthesis problem \([21]\). Verifying existence of such controllers is known to be undecidable in general even when the sets \( \mathcal{X}_i, \mathcal{U}_i, \mathcal{D}_i \) are finite \([4, 8]\). Any architecture defining the information flow in a distributed synthesis problem can be captured by choosing some \( I_i' \) and setting \( Y_i = \{X_j\}_{j \in I_i'} \), therefore synthesis in the ITIA setting cannot be easier. Given this hardness result, we instead search for sufficient conditions on local controllers under which global safety is guaranteed. These conditions can be checked locally and instantaneously in time. Moreover, instead of working with a fixed observation structure, we will deduce the sets \( Y_i \) each local agent should have access to in order to be able to comply with the conditions.

**4.2 Responsibility-Sensitive Safety**

Responsibility-sensitive safety consists of two rules. The first rule handles safety of the individual dynamics and the second rule handles safety during triggering interactions. Consider the first rule:

**Definition 4 (Self-Safety)**. A control policy \( \gamma_i \) renders agent \( \mathcal{H}_i \) self-safe on a set \( X_{i,c} \subseteq X_i \) if for all states in \( X_{i,c} \), the control input guarantees \( \mathcal{H}_i \)’s own safety assuming a reset will not happen.
in the next step. In math, for all $t$, if $x_i(t) \in X_{ie}$, then $u_i(t)$ produced by $y_j$ is such that

$$f_i(x_i(t), u_i(t), D_i) \subseteq X_{ie}.$$  

(6)

It is clear from (2) that for the existence of a self-safe controller on $X_{ie}$, $X_{ie}$ should be a robust controlled invariant set. Moreover, the controller would only need information on the agent’s own state to be in $\Upsilon_i$. Though even when $X_{ie}$’s are robust controlled invariant sets for $\Sigma_i$’s, adopting a controller that renders $X_{ie}$ invariant, an agent $\mathcal{H}_i$ cannot be guaranteed to remain in $X_{ie}$ because its state trajectories depend on both $f_i$ and $R_i$. To incorporate the potential resets of agent $\mathcal{H}_i$ into an invariance condition, any agent $j$ contributing to a reset on agent $i$, i.e., $j \in \rho_i(t)$, should somehow make guarantees about $R_i$ on the same set. While it may be problematic to expect agents to know $\rho$ in a distributed setting, an over-approximation of the value of $\rho$ at each time step, as defined next, can be obtained locally in many practical scenarios.

**Definition 5 (Resolution Over-approximation).** A function $\hat{\rho}_i : T_0 \times T_i \times T_2 \times \cdots \times T_{I_i} \to 2^{\mathbb{D}_i}$ is an over-approximation of the $i$th component of the resolution function if and only if:

$$\rho_i(t, t_1', t_2', \cdots, t_{I_i}') \in \hat{\rho}_i(t, t_1', t_2', \cdots, t_{I_i}').$$

for all values of $t$ and all $t_i' \in T_i$ for all $i$. Similarly, we say $\hat{\rho}$ is an over-approximation of the resolution function $\rho$, denoted as $\hat{\rho} \supseteq \rho$, if and only if each $\hat{\rho}_i$ is an over-approximation of corresponding $\rho_i$.

For notation convenience, when the triggering action arguments of $\hat{\rho}$ are clear from the context or are irrelevant, we simply write $\hat{\rho}_i(t)$. We discuss how resolution over-approximations can be obtained locally by each agent in Section 4.3. In general, different agents $j$ might have different resolution over-approximations $\hat{\rho}^{(j)} \supseteq \rho$ depending on their local information. With this in mind, to enable safety through resets, we define a responsibility rule that uses such over-approximations.

**Definition 6 ($\hat{\rho}$-Responsibility).** Given an over-approximation $\hat{\rho}$ of the resolution function and a collection $\{X_{ie}\}_{i \in I}$ of sets, a controller $y_j$ renders an agent $\mathcal{H}_j$ $\hat{\rho}$-responsible with respect to the sets $\{X_{ie}\}_{i \in I}$ if, when agent $j$ triggers a reset on other agents, agent $j$’s triggering action does not lead to safety violations for any other agent that it could induce a reset on according to $\hat{\rho}$, possibly including itself. In math, the controller $y_j$ renders $\mathcal{H}_j$ $\hat{\rho}$-responsible, if for all $t$, $\tau_j(t)$ and $u_j(t)$ produced by the controller are such that if $\tau_j(t) \neq \epsilon$ and $x_i(t) \in X_{ie}$ for $i \in I$, then for all $i \in I$ and $S \in \hat{\rho}_i(t)$ with $j \in S$ we have:

$$R_i(t, x_i(t), U_i, S) \subseteq X_{ie} \quad \text{if } i \neq j,$$

$$R_i(t, x_i(t), u_j(t), S) \subseteq X_{ie} \quad \text{if } i = j.$$ 

(7)

We use controller being $\hat{\rho}$-responsible (or, self-safe), agent being $\hat{\rho}$-responsible (or, self-safe) and controller rendering an agent $\hat{\rho}$-responsible (or, self-safe), interchangeably. With all of the above the following theorem can be stated, which provides a recursive safety guarantee.

**Theorem 1.** Consider an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i \in I}$, an accompanying collection of sets $\{X_{ie}\}_{i \in I}$ that are robustly controlled invariant for respective $\Sigma_i$’s in their respective safe sets $\{X_{isafe}\}_{i \in I}$ and a collection $\{\hat{\rho}^{(i)}\}_{i \in I}$ of resolution over-approximations. Then,

1. there exists local controllers $y_i$ for each agent $\mathcal{H}_i$ that render them self-safe and $\hat{\rho}^{(i)}$-responsible with respect to the sets $\{X_{ie}\}_{i \in I}$ and
2. if each agent uses a controller $y_i$ that renders itself self-safe and $\hat{\rho}^{(i)}$-responsible with respect to the sets $\{X_{ie}\}_{i \in I}$, the state trajectories corresponding to any $\{y_i\}_{i \in I}$-controlled execution of $\{\mathcal{H}_i\}_{i \in I}$ beginning in $\{X_{ie}\}_{i \in I}$ always remain within these sets.

Proof. To prove statement (1), consider, for each $\mathcal{H}_i$, a controller that produces an input $u_i(t) \in \{u \mid f(x_i(t), u, D) \subseteq X_{ie}\}$ and the triggering action $\tau_i(t) = \epsilon$ for all time $t$ for which $x_i(t) \in X_{ie}$. With the triggering action $\tau_i(t) = \epsilon$, the controller $y_i$ trivially satisfies the definition of $\hat{\rho}$-responsibility. Also, $X_{ie}$ being a robust controlled invariant set guarantees $u_i(t)$ exists whenever $x_i(t) \in X_{ie}$ and with this $u_i(t)$ the controller satisfies (6).

To show statement (2), we use induction on $t$. In the base case ($t = 0$), by assumption, all agents satisfy $x_i(0) \in X_{ie}$. Assume at time $t = k$, each agent’s state $x_i(k)$ is in its corresponding set $X_{ie}$. The controller $y_i$ either produces (i) $\tau_i(k) = \epsilon$ or (ii) $\tau_i(k) \neq \epsilon$.

First, consider case (i). Since controller $y_i$ renders $\mathcal{H}_i$ self-safe, $u_i(k) \in U_i$ produced by it satisfies (6). With this choice of $u_i(k)$, there are two possibilities for state evolution. If $\tau_i(k) = \emptyset$, the state $x_i$ evolves with the first line of Eq. (3c) and we have $x_i(k+1) \in X_{ie}$ by (6). If $\tau_i(k) \neq \emptyset$, state $x_i$ evolves with the second line of Eq. (3c), that is, $x_i(k+1) \in R_i(k, x_i(k), u_i(k), \rho_i(k))$. Let $j \neq i \in \rho_i(k) \in \hat{\rho}^{(j)}(k)$. By Assumption 1, $\tau_j(k) \neq \epsilon$. By agent $j$ being $\hat{\rho}^{(j)}$-responsible with $\tau_j(k) \neq \epsilon$, for any $S \in \hat{\rho}^{(j)}$ with $j \in S$, and, in particular for $S = \rho_i(k)$, the first line of (7) is satisfied. Since $R_i(k, x_i(k), u_i(k), \rho_i(k)) \subseteq R_i(k, x_i(k), U_i, \rho_i(k))$, and $j \in \rho_i(k)$ was arbitrary, $x_i(k+1) \in X_{ie}$ follows.

Now, consider case (ii). By assumption, the controller $y_i$ produces $\tau_i(k) \neq \epsilon$ and $u_i(k)$ such that both conditions in (7) and condition (6) are satisfied. Then, if $\tau_i(k) = \emptyset$, the state $x_i$ evolves with the first line of Eq. (3c) and we have $x_i(k+1) \in X_{ie}$ by (6). If $\tau_i(k) \neq \emptyset$, state $x_i$ evolves with the second line of Eq. (3c), that is, $x_i(k+1) \in R_i(k, x_i(k), u_i(k), \rho_i(k))$. Let $j = i \in \rho_i(k) \in \hat{\rho}^{(j)}(k)$. If $j \neq i$, the reasoning in case (i) above holds. If $j = i \in \rho_i(k)$, by assumption, $u_i(k)$ also satisfies the second line of (7) for any $S \in \hat{\rho}^{(j)}(k)$ with $j \in S$, and in particular for $S = \rho_i(k)$. Therefore, $x_i(k+1) \in R_i(k, x_i(k), u_i(k), \rho_i(k)) \subseteq X_{ie}$. □

This theorem essentially says for IThA, existence of controlled invariant sets for individual dynamics is a sufficient condition for ensuring global safety. However, this is not a necessary condition and our results do not apply to the cases where the only way to ensure safety is via triggering. The next result relates the safety and responsibility conditions to “not being at fault” as in [24] in the sense that if an agent’s control policy is self-safe and $\hat{\rho}$-responsible, there exists controllers for the remaining agents such that the overall system stays safe.

**Corollary 1.** Consider an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i \in I}$, an accompanying collection of sets $\{X_{ie}\}_{i \in I}$ that are robustly controlled invariant for respective $\Sigma_i$’s in their respective safe sets $\{X_{isafe}\}_{i \in I}$ and a collection $\{\hat{\rho}^{(i)}\}_{i \in I}$ of resolution over-approximations. If some subset $\{\mathcal{H}_i\}_{j \in I'}$ with $I' \subset I$ of agents
have controllers \( y_i \) which are both self-safe and \( \hat{\rho}^{(j)} \)-responsible on the sets \( \{X_{ij}\}_{j \in J} \), then there exists controllers \( y_i \), for all of the other agents \( \{\mathcal{H}_j\}_{j \in J} \), such that the state trajectories corresponding to any \( \{y_i\}_{i \in I} \)-controlled execution of \( \{\mathcal{H}_j\}_{j \in J} \) beginning in \( \{X_{ij}\}_{i \in I} \) always remain within these sets.

Proof. For each \( i \in I \setminus I' \), consider the controller \( y_i \) that produces an input \( u_i(t) \in \{u | f(X_i(t), u, D) \subseteq X_{ic} \} \), which exists by \( X_{ic} \) being robust controlled invariant, and triggering action \( \tau_i(t) = \epsilon \) for all time \( t \) for which \( x_i(t) \in X_{ic} \). As shown in the proof of Theorem 1 statement (1), such \( y_i \) is self-safe and \( \hat{\rho}^{(i)} \)-responsible on \( X_{ic} \). Since \( y_j \) for \( j \in I' \) are given to be self-safe and \( \hat{\rho}^{(j)} \)-responsible on \( X_{jc} \), with the above choice of controllers for agents in \( I \setminus I' \), all the controllers are self-safe and \( \hat{\rho} \)-responsible, which by statement (2) of Theorem 1 ensures safety of the executions. \( \square \)

One can try to verify self-safety and responsibility for given sets \( \{X_{ic}\}_{i \in I} \), controllers \( y_i \) and resolution over approximations \( \hat{\rho} \). Conditions (6) and (7) can also be used to synthesize controllers that render an ITHA self-safe and \( \hat{\rho} \)-responsible or to supervise existing controllers at run-time. The latter two are the use cases we demonstrate in Section 5 using robust controlled invariant sets for \( \{X_{ic}\}_{i \in I} \). Given \( \hat{\rho} \), the basic idea is to construct the set of all triggering actions and control inputs that together satisfy conditions (6) and (7). This set is always non-empty when \( \{X_{ic}\}_{i \in I} \) are robust controlled invariant sets and it can be constructed at run-time. Then, for synthesis, a pair \( (u_j, \tau_j) \) is picked from this set and for supervision, we check if the controller’s \( u_j, \tau_j \) is in this set or not. A few comments are in order as to what information, in general, is needed to construct this set, which also prescribes what observations should be included in \( Y_j \) to implement a controller \( y_j \) constructed this way. In general, the states of all agents \( i \), for which \( j \) appears in the sets in \( \hat{\rho}(t) \) should be included in \( Y_j \). However, we note that the reset maps together with the collection \( \{X_{ic}\}_{i \in I} \) of sets in practice have more structure that can simplify checking for \( \hat{\rho} \)-responsibility or the amount of observations needed. For instance, for the processor example, for all \( S \subseteq T \), \( R_t(\cdot, \cdot, S) \subseteq X_{ic} \) implies for all \( S' \) with \(|S'| \leq |S|\), \( R_t(\cdot, \cdot, S') \subseteq X_{ic} \). In words, if the processor \( i \) is safe when recruited by a number of other processors, it will be safe when recruited by a smaller number of processors. This implies that it is enough to check the condition (7) only for the largest cardinality \( S \) containing \( j \) instead of all such sets. Similarly, for the highway example, for all \( S \subseteq I \), there is \( S' \subseteq S \) such that \( R_t(t, \cdot, S') \subseteq X_{ic} \) implies \( R_t(t, \cdot, S) \subseteq X_{ic} \) for all non-empty \( S \subseteq S \). This is because there is a “worst-case” lane switching leading to a “worst-case” reset. In a sense, it does not matter what switching actions an arbitrary agent takes; only agents close to agent \( j \) matter. Thus, an ego vehicle can reason over the sets of agents switching lanes nearest to itself while still being able to guarantee safety. It is also worth remarking that when either \( R_t \) or \( X_{ic} \) is not known exactly, an over-approximation of \( R_t \) and an under-approximation of \( X_{ic} \) can be used in (7) while still guaranteeing overall safety of the ITHA per Theorem 1. Moreover, as discussed in the next section, \( \hat{\rho}^{(i)} \) can be constructed on the fly, meaning \( \hat{\rho}^{(i)} \) is only known up to \( \hat{\rho}^{(i)}(t) \) at a given time \( t \) but this is enough to construct a controller that is self-safe and \( \hat{\rho}^{(i)} \)-responsible at time \( t \).

4.3 Finding Resolution Over-Approximations

Both complexity and conservativeness can be exacerbated if the over-approximation \( \hat{\rho}^{(i)} \) is “far” from the true \( \rho \). The task of identifying proper over-approximations of \( \rho \) is thus a vitally important one. Insights into the structure of the problem can be used to generate good over-approximations.

We start this section with a relatively easy to compute, yet possibly conservative, over-approximation. Then, we define an order between agents through which they can communicate and substantially reduce conservatism. Along the way, we also discuss how these over-approximations look for our running examples.

4.3.1 Trivial Over-Approximations: If we assume each agent knows the resolution function \( \rho \), a trivial over-approximation of the resolution function can be locally computed at each time step by considering every possible choice of triggering inputs for all other agents. In math, agent \( i \) computes the \( j^{th} \) component of a trivial over-approximation \( \hat{\rho}^{(j)} \) as:

\[
\hat{\rho}^{(j)}(t) = \begin{cases} 
\rho^j \in 2^T \quad \exists \eta_j \in T_{ij} \quad \forall k \in I \setminus \{j\} : 
\rho^k = \rho^j(t, \tau_1, \ldots, \tau_i, \ldots, \tau_{I\setminus I}) 
\end{cases}
\] (8)

It can be easily shown that \( \hat{\rho}^{(ij)} \), components of which are constructed as above is an over-approximation of \( \rho \).

Example 3 (Cont’d Example 1). For the server farm in Fig. 1, we revisit the case where \( \tau_1(t) = \epsilon \) and \( \tau_2(t) = \{1, 6\} \), which leads to \( \rho_1(t) = \{4\} \), regardless of the triggering actions of the remaining agents. Using the trivial over-approximation, processor 1 has \( \hat{\rho}_1^{(1)}(t) = \{\emptyset, \{2\}, \{4\}, \{2, 4\}\} \), processor 2 has \( \hat{\rho}_2^{(2)}(t) = \{\emptyset, \{4\}\} \) and processor 4 has \( \hat{\rho}_4^{(4)}(t) = \{\{4\}, \{2, 4\}\} \). Note that no estimates depend on \( \tau_1(t) \), as agent 1 cannot recruit itself.

Example 4 (Cont’d Example 2). Consider the trivial over-approximation from the perspective of the ego agent, in the case where \( \tau_2(t) = \epsilon \); the only possible resets depend on if \( L_1 \) does or does not trigger a reset on \( E \); that is, \( \hat{\rho}_E^{(1)}(t) = \{\emptyset, \{L_1\}\} \). If instead \( \tau_2(t) = \text{left} \), it is more complicated: \( \hat{\rho}_E^{(1)}(t) = \{\{E\}, \{E, L_3\}, \{E, L_3, L_1\}\} \). The first case occurs if neither \( L_1 \) nor \( L_3 \) changes to the center lane simultaneously, the second case occurs if \( L_3 \) changes to the center lane, regardless of the triggering action of \( L_1 \), and the last case occurs if \( L_1 \) makes a lane change and \( L_3 \) does not.

4.3.2 Ordered Actions. In some situations, agents in an inter-triggering hybrid automaton \( \{\mathcal{H}_j\}_{j \in J} \) can communicate their planned actions with one another. If such communication is done in an “ordered” manner, it can allow agents to obtain much refined over-approximations as they no longer need to consider all possible actions of the other agents.

We again assume each agent knows the resolution function \( \rho \). Moreover, we assume at each time that there is a total order \( \succeq_t \) among the agents that all agents know and use to communicate their planned triggering inputs.

Footnote 2: The assumption of \( \succeq_t \) being a total order can be relaxed. In particular, agents can still get an over-approximation for a class of partial orders \( \succeq_t \) for which the Hasse diagram of the partially ordered set \( \{\mathcal{H}_j\}_{j \in J}, \succeq_t \) is a rooted forest at each time, i.e., an agent does not receive information from two incomparable agents at a given time.
Given such an order, we propose Algorithm 1 for each agent to construct their resolution over-approximation at each time $t$. With abuse of notation and without loss of generality, we assume that the automaton $\{\mathcal{H}_i\}_{i \in I}$ is (re)ordered/reindexed (by keeping track of their associated overestimates computed so far) at each time $t$ so that $\{\mathcal{H}_i\}_{i \in I} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_|I|\}$ and the index of each agent represents its ranking according to $\succeq$. Then, Algorithm 1 is called starting with $\mathcal{H}_i$ to compute resolution over-approximations $\hat{\rho}^{(i)}$ and to choose a triggering action $\tau_i(t)$ for which a self-safe and $\hat{\rho}^{(i)}$-responsible control input $u_i(t)$ exists. Then, $\tau_i(t)$ is shared with the next agent and agent $\mathcal{H}_2$ calls the algorithm with $\{\tau_i(t)\}$, and so on, until all agents compute their triggering actions for time $t$. Then these actions are executed and time progresses.

**Algorithm 1: Resolution over-approximation construction with ordered actions**

**Result:** $\hat{\rho}^{(j)}(t, \tau_j, \tau_j(t))$

**Input:** $\mathcal{H}_j, \{\tau_j(t)\}_{t \in \mathbb{I}}, \{x_j(t)\}_{t \in \mathbb{I}}$

1. $\mathcal{T}_j \leftarrow \emptyset$
2. for $t \in \tau_j$ do
3.   $\hat{\rho}^{(j)}(t, \tau_j) \leftarrow \emptyset$
4.   for $(\tau_j, \cdots, \tau_j(t)) \in \mathcal{T}_j \times \cdots \times \mathcal{T}_j(t)$ do
5.     $S \leftarrow \rho(t, \tau_j(t), \tau_j(t), \cdots, \tau_j(t), \tau_j(t), \cdots, \tau_j_{|I|})$
6.     if $\forall u_j \in \mathcal{U}_j, R_j(t, x_j(t), u_j, S) \nsubseteq X_{j, s}$ then
7.       continue;
8.     if $\exists k \neq j$ s.t. $R_k(t, x_k(t), \mathcal{U}_k, S) \nsubseteq X_{k, s}$ then
9.       continue;
10.    $\hat{\rho}^{(j)}(t, \tau_j) \leftarrow \hat{\rho}^{(j)}(t, \tau_j) \cup \{S\}$
11.    $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{t\}$
12. $\tau_j(t) \in \mathcal{T}_j$

This scheme, and particularly the method for constructing $\hat{\rho} = \{\hat{\rho}_i\}_{i \in I}$ can be shown to produce an over-approximation.

**Lemma 1.** Calling Algorithm 1 at each time step according to order $\succeq$ produces functions $\hat{\rho}^{(j)}$, each of which is over-approximation of $\rho$ at every time step.

**Proof.** Note that $\rho(t) = \{\rho_1(t, \tau_1(t), \tau_2(t), \cdots, \tau_{{|I|}}(t))\}_{t \in \mathbb{I}}$. Thus, when $\{\tau_i(t)\}_{i \in I}$ is completely known, we explicitly know $\rho(t)$. At the $|I|$th call of Algorithm 1 at time $t$, after $\tau_{{|I|}}(t)$ is chosen then $\{\tau_i(t)\}_{i \in I}$ is completely known. This indicates that $\rho(t) \in \hat{\rho}^{(|I|)}(t)$ for all $t$.

Now, consider an arbitrary call $k$ of the Algorithm 1 at time $t$. By definition, $\hat{\rho}_i^{(k)}(t)$ contains all resolutions $\rho_i(t, \tau_i(t), \tau_i(t), \cdots, \tau_i_{k-1}(t), \tau_i_{k+1}(t), \cdots, \tau_i_{|I|}(t))$ where $\tau_i_{k-1} \cdots \tau_i_{|I|}$ are arbitrarily chosen. Similarly, $\hat{\rho}_i^{(k-1)}$ contains all resolutions $\rho_i(t, \tau_i(t), \tau_i(t), \cdots, \tau_i_{k-2}(t), \tau_i_{k-1}(t), \tau_i_{k+1}(t), \cdots, \tau_i_{|I|}(t))$ where $\tau_i_{k-1} \cdots \tau_i_{|I|}$ are arbitrarily chosen. By observation one can see that $\hat{\rho}_i^{(k)} \supseteq \hat{\rho}_i^{(k-1)}$. Therefore, by induction $\hat{\rho}_i = \hat{\rho}_i^{(k)}$ for any $k$ and any $i$. Therefore, the functions $\hat{\rho}^{(j)}$ produced by Algorithm 1 are an over-approximation of $\rho$ at every time step. □

**Example 5 (Order in Highway Example).** In the highway example, one can assume that any vehicle (e.g., vehicle $E$ in Fig. 3) on the highway sees the actions of the vehicles in front of it (or on a slower lane when two agents are aligned) and can use those to inform its own lane change decisions. In this way, a time-varying ordering is implemented where $\mathcal{H}_2 \succeq \mathcal{H}_1$ if and only if $\mathcal{H}_2$ is in front of $\mathcal{H}_1$ or they are aligned and $\mathcal{H}_1$ is on a slower lane compared to $\mathcal{H}_2$. This gives a total order across agents at each time $t$.

For example, in Fig. 3 the ordering is $\mathcal{H}_1 \succeq \mathcal{H}_2 \succeq \mathcal{H}_E \succeq \mathcal{H}_F$. This choice is motivated by the intuition that the ego vehicle $\mathcal{H}_E$ is behind vehicles $\mathcal{H}_1$, and $\mathcal{H}_2$, so it can see their actions. As concrete examples of what Algorithm 1 outputs in this case, if $\tau_E(t) = e$, then $\hat{\rho}^{(E)}(t) = \{L_1\}$ when $\tau_2(t) \neq e$ and $\hat{\rho}^{(E)}(t) = \emptyset$ otherwise; that is, $\hat{\rho}^{(E)}(t)$ is not conservative, as it will observe the triggering action of $L_1$. Similarly, if $\tau_E(t) = L_1$, then $\hat{\rho}^{(E)}(t) = \{f_E(t)\}$, since $E$ sees the triggering actions of both $L_3$ and $L_4$ and thus there is no conservativeness.

**5 EXPERIMENTS**

We evaluate the flexibility and applicability of ITHA by using it to perform single-agent control in the highway driving scenario (Section 5.1) and multi-agent control in the parallel processing scenario (Section 5.2). Finally, we evaluate the conservativeness of the ITHA-based responsibility-sensitive safety rules on a real highway driving data-set (Section 5.3). Our software implementation is published at [1].

**5.1 Single-agent control: highway driving**

We demonstrate that ITHA can be used on the highway driving scenario, as described in Example 2, where only the ego vehicle is controlled and seeks to remain safe and responsible with respect to the uncontrolled vehicles. The ego vehicle $E$ seeks to track a nominal velocity $v_{nom} = 15m/s$, formally solving the following receding horizon control problem at each time-step:

$$\min_{x_E(t)}, t, \min_{x_{E}(t)} \sum_{t_0 \leq t < t_0 + H} \|u(t) - v_{nom}\|^2_2$$

s.t. $x_E(t + 1) = f_E(x_E(t), u_E(t), \tilde{d}(t)), \quad t = t_0 + 2, \ldots, t_0 + H - 1$

$x_E(t_0 + 1) \in X_{E, i}$

$x_E(t_0 + 1) = f_E(x_E(t_0), u_E(t_0), \tilde{d}(t_0)), \quad \text{if } t_0(t_0) = e$  \hspace{1cm} (9)

$x_E(t_0 + 1) = R_E^F(x_E(t_0), u_E(t_0), \|\hat{\rho}_j(t_0)\|) \in X_{E, c}, \quad \forall u_j(t_0) \in \mathcal{U}_j, \forall j \in \mathcal{J}_E, \quad \text{if } t_0(t_0) = l$

$x_E(t_0 + 1) = R_E^F(x_E(t_0), u_E(t_0), \|\hat{\rho}_j(t_0)\|) \in X_{E, c}, \quad \forall u_j(t_0) \in \mathcal{U}_j, \forall j \in \mathcal{J}_E, \quad \text{if } t_0(t_0) = r$

and executing $u_E(t_0) \in \mathcal{U}_E$, where the prediction horizon $H = 25$, the predicted disturbance $\tilde{d}(t) = 0$ if $t > t_0$ and $\tilde{d}(t) = -10$ if $t = t_0$, and the continuous dynamics $f_E(\cdot, \cdot, \cdot) \rightarrow \mathbb{R}$ are as in (5), where $\Delta t = 0.1$. Furthermore, $\mathcal{J}_E = [-10, 10]$ for all agents $i$, and we define $\hat{\rho}_j(t) = \{i \in \mathcal{I} \mid j \in \hat{\rho}_i(t), j \notin \mathcal{E} \neq \emptyset\}$ as an over-approximation of the set of all agents that $E$ can trigger at time $t$. We will shortly describe the specific $\hat{\rho}$ that we use in our experiments. Finally, we abuse notation to define $R_E^F(\cdot, \cdot, \cdot)$ and $R_E^F(\cdot, \cdot, \cdot)$ as functions which output the reset state upon making a left and right lane change, respectively.

To interpret (9), we note that the first constraint enforces the continuous dynamics from the second timestep onwards, and the second constraint enforces self-safety. The third constraint enforces...
the continuous dynamics at the first timestep if no triggering action is taken, while the fourth and sixth constraints enforce an appropriate state reset if the ego agent performs a left or right lane change, respectively. Finally, the fifth and seventh constraints enforce that all agents that are triggered by the ego vehicle’s lane change action can remain safe by applying any control input.

Note that (9) can be represented as a mixed integer quadratic program, where \( \tau (t_0) \in \{ e, \text{left}, \text{right} \} \) can be modeled with an integer decision variable \( z \in \{ 0, 1, 2 \} \) used within a big-M formulation [2] to determine the lane change choice. To improve performance, (9) only seeks to enforce self-safety and responsibility at the first time-step (the system remains safe as only the input from first time-step is executed; hence, only safe actions are applied).

The uncontrolled vehicles are simulated using the Intelligent Driver Model (IDM) [26]:

\[
\begin{align*}
    x_i(t + 1) &= x_i(t) + \Delta t v_i(t) \\
    v_i(t + 1) &= v_i(t) + \Delta t a \left( 1 - \frac{v_i(t)}{v_i^*} \right) - \left( \frac{c_i(t)}{2} \right)^2,
\end{align*}
\]

with parameters \( \delta = 4, s_0 = 5, T_H = 1.5, a = b = 10 \) and randomly sampled nominal velocities \( \hat{v}_i \). Here, \( c_i(t) \) refers to the velocity of agent \( i \)'s lead car. We execute for 500 time-steps, solving (9) at each time-step; see Fig. 5 for a visualization of an example execution.

To illustrate the impact of different over-approximations of \( \rho \) on conservativeness, we compare control performance under these \( \hat{\rho} \):

(A) The trivial over-approximation (8)

(B) The ordered over-approximation described in Algorithm 1, with the time-varying ordering as described in Example 5

A video showing the behavior of the \( \hat{\rho} \)-responsible ego vehicle when using the two different over-approximations is available here: https://youtu.be/a5IULWQYVzM. Under over-approximation (A), the ego vehicle travels 351.9 meters, while it travels 415.3 meters under over-approximation (B), averaged over 25 random initializations of the uncontrolled vehicles. In all simulations for both over-approximations, we did not experience any unsafe behavior, as guaranteed by the theory. We note that the performance improvement of the second over-approximation is a result of it being less conservative than the first. For instance, consider the example in Fig. 5. The simulation remains the same for both over-approximations up until time \( t = 86 \). At time \( t = 87 \), the ego agent is unable to make an advantageous lane change to lane 3 under over-approximation (A), because if the car in lane 4 also changes to lane 3 simultaneously, it would lead to a safety violation. However, under over-approximation (B), the ego agent can safely make that lane change because the ordering implies that the car in lane 4 observes and should yield to the triggering action of the ego vehicle. A similar event occurs at time \( t = 201 \): under (A), the ego vehicle cannot make an advantageous switch to lane 4, because if the car in lane 5 is to simultaneously switch to lane 4, it would result in safety violations. By the end, the ego vehicle travels 161 meters further under (B) than under (A) (Fig. 5, bottom).

Finally, we note that while (B) outperforms (A) on average, (A) can still possibly outperform (B) for specific assignments of the

<table>
<thead>
<tr>
<th>( \rho ) estimate (A)</th>
<th>Avg. no. accepted jobs</th>
<th>Avg. safety violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rho ) estimate (B)</td>
<td>268.6</td>
<td>147.64</td>
</tr>
<tr>
<td>( \rho ) estimate (C)</td>
<td>302.72</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Parallel processor statistics, averaged over 25 runs.

uncontrolled vehicles. This occurs because (9) is restricted to a one-step plan for triggering actions, so the ego vehicle can make extra lane changes under (B) that can cause it to get trapped behind a slow car without realizing that it can free itself using a long sequence of lane changes; planning triggering actions over a longer horizon would aid in escaping from these “local optima”.

Overall, this experiment suggests we can use ITHA-based controllers to control an agent in a multi-agent environment with safety guarantees under limited communication, and that conservativeness of \( \hat{\rho} \) can affect control performance.

5.2 Multi-agent control: parallel processors

We demonstrate ITHA on the parallel processor scenario, as described in Example 1, where we control all agents (processors). Our task is to maximize the number of accepted jobs over a finite horizon. At each time-step, we indirectly achieve this in a decentralized, receding-horizon fashion by computing control inputs and triggering inputs individually for each agent, which greedily minimize the number of remaining jobs for that agent. Formally, for each agent \( i \), we solve the following integer program at each time-step \( t \):

\[
\begin{align*}
    \min_{u_i(t), \tau (t)} & \quad x_i(t + 1) \\
    \text{s.t.} & \quad x_i(t + 1) = x_i(t) - u_i(t) + \delta_i(t) - \sum_{j=1}^{|I|} t_j^i(t) \\
    & \quad x_i(t + 1) \in X_i, \\
    & \quad x_i(t + 1) \geq 0 \\
    & \quad t_j^i(t) = 0, \forall j \notin \text{out}_{G} (i) \\
    & \quad \mathcal{R}_j(t, \tau_j(t), u_j(t), \bigcup_{i \in I} \{ \hat{\rho}_j(t) (i) \}) \in X_j, \\
    & \quad \forall v_j(t) \in \mathcal{U}_j \forall j : t_j^i(t) = 1
\end{align*}
\]

where \( u_i(t) \in [0, 1] \) and \( \tau (t) \in [0, 1]^{\mathcal{I}} \). Here, \( |I| = 10, n_{\text{hhrottle}} = 3, \) and \( n_{\text{overflow}} = 5 \). We define an over-approximation of the set of agents that agent \( i \) can trigger at time \( t \), \( \hat{\rho}_i(t) \), in the same way as in the highway example. Similar to the highway example, we will compare performance between three \( \rho \) estimates:

(A) The trivial over-approximation (8). Here, \( \hat{\rho}_i(t) = 2^{\text{in}(i)} \).

(B) An under-approximation \( \rho_{\text{bad}}(t) \subseteq \{ i \} \), that is, when planning \( \tau_i(t) \), agent \( i \) assumes no other agent will recruit \( j \).

(C) The ordered over-approximation described in Algorithm 1, with a time-invariant priority order sorted by processor index, i.e. \( \mathcal{H}_1 > \ldots > \mathcal{H}_f \).

We simulate 25 runs, each over a horizon of 50 time-steps, and report the performance statistics in Table 1. In each run, we generate a random undirected connectivity graph \( G \), where an edge between agents \( i \) and \( j \) exists if a sample uniformly drawn from \( [0, 1] \) is greater than or equal to 0.1. Disturbances \( d_i(t) \) are also generated randomly. Note that using the \( \rho \) estimate (A) leads to conservative performance, since there is no communication; thus, for agent \( i \) to recruit agent \( j \), it must guarantee that agent \( j \) can remain safe if the rest of agent \( j \)'s neighbors also trigger it. This overall leads to few recruitment actions, and thus many jobs are
With the ordered contract, we can avoid this mismanagement, more velocity. Violations can occur, such as when many other processors recruit simultaneous lane change from the lane 4 agent to lane 3.

Figure 5: Highway driving example. Red: ego vehicle. Blue: uncontrolled vehicles. Arrow magnitudes are proportional to agent velocity. Top row: the ego vehicle can make an advantageous change to lane 3 under (B), but not under (A) due to a hypothetical simultaneous lane change from the lane 4 agent to lane 3. Middle row: the ego vehicle cannot make an advantageous lane change to lane 4 under (A) due to a hypothetical simultaneous lane change from the lane 5 agent to lane 4. Bottom row: At the end of the simulation, there is a large performance gap between using over-estimates (A) and (B).

rejected. On the other hand, (B) is less conservative, but as it is an unsafe estimate of $\rho$ (since it is an under-approximation), safety violations can occur, such as when many other processors recruit one processor during the same time-step, leading to the recruited processor exceeding $t_{\text{overflow}}$. As a side effect, the average number of accepted jobs is also lower under (B) since many processors are often over the throttle limit, limiting the number of incoming jobs. With the ordered contract, we can avoid this mismanagement, more efficiently allocating jobs among the processors and preventing jobs from being unnecessarily rejected while remaining safe.

Overall, this experiment suggests that we can also use ITHA to control multiple agents in a decentralized fashion with safety guarantees, and that it is vital to select an appropriate $\rho$ estimate to ensure safety and good performance.

5.3 Supervision: Evaluation of ITHA on data

Several frameworks for autonomous driving have sought to supervise a performance controller with a safety supervisor, which overrides when the performance controller may lead the system to an unsafe state. These supervisors often use invariant sets or control barrier functions to detect these safety violations. However, the usefulness of a safety supervisor is often dependent on its conservativeness, i.e. it should not unnecessarily override, as the jerkiness of changing controllers may annoy or frighten the user. To empirically demonstrate that the ITHA framework can serve as a high-quality safety supervisor that provides rigorous safety guarantees while remaining sufficiently permissive, we demonstrate that an ITHA-based supervisor achieves low override rates when supervising on a real world highway driving data-set [13].

The HighD data-set consists of trajectories of each driver’s position with annotations, such as the vehicle lane and vehicle class (motorcycle, truck, or car), with data recorded at six different locations on the German Autobahn at various times of day. We use 110516 trajectories from the HighD data-set, containing a total of around $4 \times 10^7$ (x, u, d) data-point tuples. The state, input, and disturbance trajectories for each car in the data-set under the dynamics (5) (the $v_i$, $h_i^d$, and $d_i^d$ trajectories) are generated as follows. $v_i$, $h_i$, and $s_i^d$ are directly provided in the HighD data-set; we compute $d_i$ via finite-differencing using the lead car velocities and $\Delta t = 0.04$ seconds, which is the provided time discretization of the data-set.

We hold out 20% of the data and use the remaining 80% as a “training set” to compute disturbance bounds. These disturbance bounds are used to compute invariant sets which are well-calibrated to the driving behavior observed in the data-set. Let the set of all disturbance trajectories in the training data-set for dynamics (5) be denoted $\Xi_d = \{\xi_d,i \}_{i=1}^N$, where N is the number of trajectories in the data-set. While the data-set can be noisy, we do not perform any denoising in this step, and instead process outliers when computing the disturbance bounds. Specifically, we process $\Xi_d$ for outliers by only keeping the data between the 0.025- and 0.975-sample quantiles $\hat{d}_{0.025}, \hat{d}_{0.975}$; that is, we concatenate $\Xi_d$, sort the result in increasing order (i.e. obtain the order statistics of $\Xi_d$), denoted $\Xi_d(1), \Xi_d(2), \ldots$, and remove all disturbances belonging in the first 2.5 and last 2.5 percent of $\Xi_d$ (this is possible, since $d$ is scalar in (5)). Formally, we define the y-sample quantile, $y \in (0, 1)$, as $\hat{d}_y = \Xi_d(\lceil yN \rceil)$, where N is the number of elements in $\Xi_d$. Let this modified data-set be denoted $\hat{\Xi}_d = \{d \in \Xi_d \mid d \notin [\hat{d}_{0.025}, \hat{d}_{0.975}] \}$. We compute an invariant set assuming disturbances $d$ satisfy $d \in D = D_t = \{d_{0.025}, \hat{d}_{0.975}\}$, and use these invariant sets $X_i,c = C_{\text{inv}}$ within an ITHA-based supervisor.

To quantify the conservativeness of using ITHA-based responsibility-sensitive safety rules to supervise highway driving,
we calculate the number of times our supervisor overrides the human control input on the trajectories observed in the data-set. Specifically, we calculate the fraction of datapoints in which self-safety (as defined in Definition 4) and $\bar{\rho}$-responsibility (as defined in Definition 6) are violated; we denote this the override rate, reported in Table 2. A low override rate indicates that our supervisor will not frequently engage and is not excessively conservative, which is desirable since there are no crashes in the data-set. As mentioned, we compare between two different over-approximations to evaluate responsibility: the first uses the trivial over-approximation (8), while the second uses the ordering contract where any agent $j$ behind agent $i$ in the direction of travel must yield to the triggering actions of agent $i$ (see Example 5 for more details).

Analyzing the override percentages in Table 2, we observe that supervising self-safety with ITHA results in relatively low override percentages, while the ordered contract outperforms the trivial contract substantially. This is to be expected, since using the trivial over-approximation leads to overrides being counted if the ego car is changing lanes and there exists another car adjacent to the new lane with similar longitudinal position as the ego car. This is common behavior (i.e., many cars may simultaneously be at similar longitudinal positions on the highway in different lanes). Note that these override rates can be further improved by employing context-dependent invariant sets generated with disturbance bounds computed on different clusters of data (contexts), i.e. only on trajectories recorded in the fast lane, or only on trajectories recorded at rush hour. Further investigation of the impact of context-dependence on the conservativeness of ITHA-based supervisor rules is an interesting direction for future work.

Overall, this experiment suggests that an ITHA-based supervisor can obtain low override rates on a real driving dataset, indicating that driving data-sets can be used to calibrate an ITHA-based safety supervisor and that such supervisors are permissive enough to avoid excessive overrides (10% for self-safety and 2% for an appropriate responsibility contract) and act as a useful safety supervisor.

### 6 DISCUSSION

In this section, we provide a few remarks on limitations and simple extensions of our framework:

- ITHA is appropriate in modeling systems whose individual dynamics are decoupled but have additional triggering actions for interaction. Our self-safety and responsibility rules utilize this structure to provide sufficient conditions for global safety. In comparison, existing compositional frameworks, such as [5, 9, 14, 17, 19, 22, 23, 25], that give sufficient conditions for global safety allow agents’ dynamics to be coupled but do not allow for triggering actions.

- We note that our approach can be extended to guarantee safety for settings in which individual systems may have communication delays or sensor noise by leveraging recent advances in invariant set computation [10, 12, 15, 29] for systems with these imperfections.

- The increased complexity of our method over other responsibility-sensitive safety frameworks for driving (i.e. [7]) can be attributed in part to analyzing "second-order" triggers, i.e. reasoning about the set of agents which can have their feasible triggering set modified by the triggering action of another agent. In the two-lane highway driving setting, such behavior does not exist (which is what is considered in most existing responsibility-sensitive highway driving frameworks) since there are no "second-order" neighbors; however, to guarantee safety when there are more than two lanes of traffic, it is vital to consider second-order behavior.

- While we show communication and introducing an order for triggering action selection can reduce conservativeness, it can be further reduced by communicating control inputs. We assume that agents evaluate responsibility using all inputs $u$ that the triggered agents can apply (see Eq. (7)); however, if agents can communicate their state and input to the triggered agent, we can relax this "for all inputs" condition to "there exists an input" according to a similar priority.

- Finally, we note our framework allows for priorities between agents which are not fixed a priori and dynamic orders can be chosen to improve performance. For instance, in the processor example, agents within the same connected component of the communication graph can mutually communicate their state, and agents with the most remaining jobs can be reassigned to have higher priority in recruiting other agents, as they are closer to the throttle threshold. So, there is a potential to employ distributed algorithms to select such dynamic orders.

### 7 CONCLUSIONS

In this paper, we introduce a novel modeling paradigm for multi-agent systems, the inter-triggering hybrid automaton, and apply it to two very different systems: parallel processing and autonomous driving. We derive an approach for proving safety of these systems using the notions of self-safety and $\bar{\rho}$-responsibility, which we show both theoretically and empirically result in guaranteed safe execution of the entire collection of agents. We also describe methods for generating practical approximations of the resolution function, and how local communication can be leveraged to improve these approximations. Finally, we demonstrate our approach on single- and multi-agent control in the parallel processing and highway driving scenarios, and furthermore evaluate the conservativeness of our approach on a safety supervisor task using real highway driving data. In future work, we wish to use assume-guarantee contracts between different agents in an ITHA to allow more coordination during resets. Also, we would like to extend our analysis to handle collections of agents whose triggering actions take multiple time steps to complete (i.e. non-instantaneous lane changes in the highway example) or have a delayed effect on the rest of the collection. Finally, we conjecture that ITHA is a special (less expressive) type of discrete-time hybrid I/O automaton [16] where each individual agent and the resolution function are hybrid I/O automata. This connection will be further investigated.
REFERENCES


A DEFINING THE RESETS AND RESOLUTION FUNCTION FOR THE HIGHWAY EXAMPLE

In order to define the reset map and the resolution function for the highway example, we need to introduce global states in a global co-ordinate system for the overall system. Both the reset map and the resolution function can be expressed as time-invariant functions of these global states. On the other hand, we model the states of individual vehicles in ITHA in some local coordinates in (5), Global-state dependent reset maps and resolution functions can be converted to time-varying variants on the individual states. Since the self-safety and responsibility conditions are defined for time-varying reset maps and resolution functions that depend on individual states, our safety results are still applicable with this conversion. This section clarifies what information each vehicle would need at a given time to compute $\rho$ and to comply with condition (7).

Let agent $i$'s global state be defined as $\vec{x}_i = [v_i \ p_i \ \ell_i]^{T} \in \mathcal{X}_i = [0, \bar{v}_{\text{max}}] \times [0, \infty) \times \{1, \ldots, n_i\}$ where $\ell_i \in \{1, \ldots, n_i\}$ is the current lane's number, $v_i$ is the current longitudinal velocity in the direction of the current lane, and $u_i$ is the current longitudinal acceleration in the direction of the current lane (i.e. the same value in (5)). Here we take the state spaces $\mathcal{X}_i$ of the vehicles to be identical for simplicity. Also, by convention, we take the rightmost lane to have value 1. Let $\{\vec{x}_i\}_{i=1}^{I}$ be the collection of all agents' global states, with $\mathcal{X}_i \triangleq \mathcal{X}_i^{I}$ denoting this state space, and $\mathcal{X}_G$ denoting the restriction of this space to agents $S \subseteq I$.

The dynamics of the global state can be written as:

$$\dot{\vec{x}}_i(t + 1) = \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{x}_i(t) + \begin{bmatrix} \Delta t \\ 0 \\ 0 \end{bmatrix} u_i(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_{\ell_i}(t),$$

where $u_i$ is the acceleration, i.e., the same control input in the ITHA representation, and

$$u_{\ell_i}(t) = \begin{cases} 1 & \ell_i(t) = \text{left}, \\ 0 & \ell_i(t) = \text{stay}, \\ -1 & \ell_i(t) = \text{right}. \end{cases}$$

Note that it is more intuitive to define control invariant sets over the local state space defined in (5) rather than in the global state space of the global coordinates since safety depends only on inter-vehicle distances. Thus, local reasoning is sufficient for the proofs of this paper. However, to define the resets, local states are not sufficient.

Let us define the following nearest leader of agent $i$ operator $L_i : \mathcal{X}_G \times \bigcup_{\ell \in I} \mathcal{X}_i \rightarrow I \cup \emptyset$. It is defined by the following simple optimization:

$$L_i(\vec{x}_i, \{\vec{x}_j\}_{j \in S}) = \arg \min_{j \in S \backslash \{i\}} |p_i - p_j|$$

subject to

$$p_i \leq p_j$$

$$\ell_i = \ell_j.$$  

In words, it finds the closest lead car within the set $S$ to agent $i$ on the same lane with it. Assuming that two cars cannot occupy the same point on the highway (in this case, the same longitudinal position and lane), the value of argmin will always be either a singleton or the empty set. However, even if it is not a singleton, picking any minimizer works for the purposes of the next lemma, which relates the global states to individual (local) ones.

**Lemma 2.** There exists a unique mapping from the global state $\{\vec{x}_i\}_{i=1}^{I}$ to the states $\{\vec{x}_i\}_{i=1}^{I}$ of the individual vehicles in the ITHA representation.

**Proof.** To map the global state $\{\vec{x}_i\}_{i=1}^{I}$ to the ITHA states $\{\vec{x}_i\}_{i=1}^{I}$, consider the mapping for one agent $H_i$:

$$\vec{x}_i = \begin{cases} v_i & L_i = \emptyset \\ \infty & \ell_i = \ell_j \\ v_i \leftarrow p_L - p_L \\ v_i \leftarrow v_{\text{max}} \end{cases}$$

where $L_i$ is the abbreviation of $L_i(\vec{x}_i, \{\vec{x}_j\}_{j \in S})$. $\square$

We define a localized version of the mapping (14), denoted $\bar{G}_S : \bar{\mathcal{X}}_S \rightarrow \prod_{i \in S} \mathcal{X}_i$, to be the mapping when $L_i$ is taken to be $L_i(\vec{x}_i, \{\vec{x}_j\}_{j \in S})$. We denote $\bar{G}_S : \bar{\mathcal{X}}_S \rightarrow \mathcal{X}_i$ the component of $G_S$ corresponding to agent $H_i$.

Now we will define the resolution functions $\rho_i : \mathcal{X}_G \times \mathcal{X}_G \rightarrow \mathcal{X}_G$ and reset maps $R_i : \mathcal{X}_G \times \mathcal{X}_G \times \mathcal{L}_G \times \mathcal{U}_G \rightarrow \mathcal{X}_G$ that depend on the global quantities.$^3$ We have

$$\rho_i : \{\{\vec{x}_i\}_{i=1}^{I}, \ell_1, \ldots, \ell_I\} \Rightarrow$$

$$\{j \in I \mid \tau_j \neq \text{stay}, \ell_j = \ell_i, p_j \geq p_i\} \cup$$

$$\{j \in I \mid \tau_j \neq \text{stay}, \ell_j + u_{\tau_j} = \ell_i + u_{\tau_i}, p_j + v_j \Delta t \geq p_i + v_i \Delta t\}.$$  

In words, the resolution function $\rho_i$ maps the global state and the triggering actions to the set of all agents that use a non-null triggering action that are on the same lane with agent $i$ and ahead of it and those whose triggering action will put them on the same lane with agent $i$ ahead of it in the next step. Then, the trivial resolution over-approximation $\rho_i(t)$ used in our experiments considers, in the worst-case, all agents ahead of agent $i$ on the same lane at time $t$ and all agents that can be on the same lane with agent $i$ ahead of it at time $t + 1$. However, many of these combinations $S$ of reset-trigerring agents appear as an output of $\rho_i(t)$ lead to the same reset value, as defined by:

$$R_i(\{\vec{x}_i\}_{i=1}^{I}, \vec{x}_i, u_i, S) = \bar{G}_S^i(\{\bar{f}_j(\vec{x}_i, u_j, u_{\tau_j})\}_{j \in S'})$$

where $S' = \{j \in I \mid \ell_j + u_{\tau_j} = \ell_i + u_{\tau_i}, p_j + v_j \Delta t \geq p_i + v_i \Delta t\}$. By construction, $\tau_j = \text{stay}$ for $j \in S' \setminus S$. Therefore, the triggering actions of the agents in $S' \setminus S$ is sufficient for estimating $R_i$ at time $t$, in addition to the estimates of other arguments. Note that the first set in the $p_i$ definition does not directly seem to contribute to the reset map but it captures the agents leaving in front of agent $i$, which in turn affect who the lead car will be in the next step. At run-time, the local controller $y_i$ does not need the knowledge of the entire global states but needs to know the lanes and relative positions ($p_i(t) - p_j(t)$) and relative velocities ($v_i(t) - v_j(t)$) of agents $j$ for which $i \in \hat{\rho}_j(t)$.

$^3$To be precise, the actual value the state is reset to, depends on the triggering actions, states, and inputs of agents in $\rho_i$ rather than the agents’ indices as mentioned in footnote 1.