Synthesis of Reactive Protocols for Vehicle-to-Vehicle Communication – Technical Report*

Clemens Wiltsche
University of Oxford, UK
cw395@cantab.net

Ufuk Topcu
University of Pennsylvania, USA
utopcu@seas.upenn.edu

Richard M. Murray
California Institute of Technology, USA
murray@cds.caltech.edu

ABSTRACT

We present a synthesis method for communication protocols for active safety applications that satisfy certain formal specifications on quality of service requirements. The protocols are developed to provide reliable communication services for automobile active safety applications. The synthesis method transforms a specification into a distributed implementation of senders and receivers that together satisfy the quality of service requirements by transmitting messages over an unreliable medium. We develop a specification language and an execution model for the implementations, and demonstrate the viability of our method by developing a protocol for a traffic scenario in which a car runs a red light at a busy intersection.

Categories and Subject Descriptors

C.2.2 [**Protocol Verification**]; B.1.2 [**Automatic Synthesis**]

Keywords

Vehicle-to-vehicle communication; Discrete controller synthesis; Active safety

1. INTRODUCTION

Active safety systems have the potential to transform automobile traffic by complementing a human operator's capabilities to prevent accidents and increase efficiency [4]. Communication between cars enables cooperative safety applications by further augmenting information gained from local sensors. Examples for active safety applications are traffic signal violation warning, cooperative collision warning and electronic emergency brake light [4, 11].

By transmitting information between each other, cars can gain a view of the traffic situation more refined than it would

---

be possible merely with sensors [8], because global information about traffic is being made available locally to the cars. Such information can be used in active safety systems to avoid accidents and even improve traffic efficiency by enabling both communication with the infrastructure and between cars [3, 4, 6, 21, 22].

Depending on the active safety application, different types of vehicle-to-vehicle (V2V) communication paradigms have been investigated. Safety applications often require to maintain continuous tracking of other cars in the vicinity, which is typically done by having cars broadcast information about their position, velocity and other parameters of their state in regular intervals [11]. Vehicle Ad-Hoc Networks (VANETs) using routing protocols such as Geocasting or Ad-Hoc Distance Vector (AODV) handle applications involving several cars in a peer-to-peer (P2P) connection [9, 13, 15]. The main challenge is to maintain reliable communication in the presence of possible channel congestion if several cars use the transmission medium simultaneously [9, 14].

Traditionally, in the development of communication protocols, the programs are implemented manually, and verification of the protocol is only done after prototyping, either through testing or model checking [7, 2, 20]. A slight improvement over this bottom-up approach is to develop a framework for distributed protocol specification and automatically generate inputs to model checkers and theorem provers [19].

In contrast, synthesis finds the programs to be executed on each car directly from a global protocol specification. In this approach, the synthesis method is guaranteed to generate distributed implementations that satisfy their specification by construction. However, so far only small problems have been considered in synthesis without particular applications in mind [17, 16, 10]. Also, protocol implementations are only valuable in practice if it is clear under which assumptions they are correct and if the communication requirements are clearly specified [9]. Only if a precise model of the environment is provided, i.e. the worst-case behavior of the transmission medium, is an argument of correctness convincing.

Synthesis is made difficult on the one hand by distributing a single global specification in a way that the distributed implementation operates correctly in an adverse environment, and on the other hand by having to ensure correctness of the results, which has to be ensured for *any* valid protocol specification given as input to the synthesis.

Our main contribution is the development of a method that automatically translates global specifications of the pro-
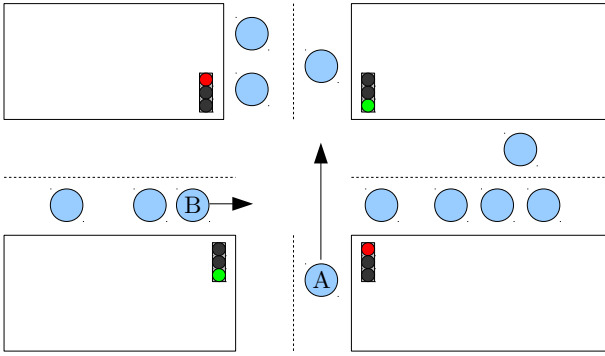
Figure 1: Intersection scenario: Car A runs the red traffic light. Hence car B (and potentially other cars) must stop. Other cars at the intersection may be transmitting over the wireless medium at the same time, which has a deteriorating effect on the communication between A and B.

tocol into implementations that formally guarantee the desired quality of service requirements under the environment assumptions. Specifically, we develop a synthesis method for reliable asynchronous communication protocols with clearly defined interfaces that can be used in a layered design. We focus on providing communication services to enable active safety applications for cars and therefore lump any active safety activities into an abstract higher level that interacts via strictly defined interfaces with the lower level communication services that we develop.

Our work addresses several shortfalls in previous work on protocol synthesis [5, 12, 17, 23]. We introduce a formal specification language to allow a textual representation of the protocol specifications, which are typically given in graphical form. Moreover, we make precise the semantics of protocol specifications and their implementations that are usually only informally described.

2. COMMUNICATION

Consider the scenario of two cars wanting to communicate with each other at an intersection, c.f. Fig. 1. One major complicating factor for reliable V2V communication is that the cars are constantly moving and have to communicate over wireless links. Cars that intend to communicate share restricted bandwidth availability with all other cars within reach. When constantly tracking cars using broadcasts, scalability is limited by the susceptibility for flooding and frequent message collisions [14], and by having to keep track of the state of every car.

It would be desirable to be able to implement communication protocols that guarantee the correct transmission of data even in the presence of a large number of other cars. We therefore consider initiating communication on-demand when required by an active safety application in an emergency. In this approach we do not track every car but only exchange information when required. This approach has the advantage that traffic on the network is lower, more predictable, and reliability guarantees can be provided, as demonstrated in this paper. We adopt an approach in which the sender is responsible for correct delivery by retransmitting data when a package drop is detected [18].

When considering a V2V communication between two cars, we do not explicitly consider the behavior of all other cars.

Since from the point of view of the transceivers it is only relevant whether a message is correctly received, we lump together the behavior of all other cars that are not directly involved in the communication and consider them as a single environment. Our method allows us to explicitly state the assumptions on this environment under which the protocol has to perform correctly.

Communication between cars is governed by a set of rules summarized as a protocol. After a data transfer is initiated, messages are transmitted and received in order to guarantee a reliable delivery. A protocol is implemented by equipping each car with a communication service automaton (CSA), which can be seen as a building block or "controller" handling all communication activities. Hence, the protocol can be seen as a building block with clearly defined behavior and interfaces to its environment consisting of higher level active safety components (ASCs) and to the lower level that handles the transmission of the messages over the physical medium.

Each CSA operates locally, i.e. it can only interact with the sensors and actuators of the car it is located on. However, since a communication protocol defines events potentially involving several cars, CSAs need to interact with each other. This interaction is done by transmitting messages between the cars e.g. using wireless transceivers.

Defining a clear hierarchy of layers is inspired from the ISO OSI architecture prevalent in most modern communication networks [24]: A *network layer* is dedicated to establishing host-to-host connections with basic quality of service (QoS) guarantees. A *data-link layer* is layered below the network layer and provides error-corrected single hop connections. Above the network layer is the *transport layer*, that among other services provides the destination address of a message and QoS requirements. We consider an abstraction in which a car's ASC contains the transport layer and all above layers. The ASC specifies parameters such as the data to be sent, the destination address, and limits on transmission delay.

3. SETUP

Developing a synthesis method requires a formal specification language for protocols and a modelling framework to formally describe executable CSAs. Moreover, the CSAs should include interfaces to their corresponding ASCs at the higher level, and hence our synthesis method is designed to introduce this inter-level interaction. To illustrate our method we will use the following example motivated by Caveney [4], and Farkas et al. [8]:

Example 1 Consider the scenario of cars at a road intersection shown in Fig. 1. Car A runs a red traffic light, and car B approaches the intersection on a trajectory that would lead to a collision. The two cars have to communicate in order to avoid an accident. At the intersection there might be other cars that share the same broadcast medium and hence might interfere with the communication between A and B. □

3.1 Operation of a Protocol

The primary objective of a communication protocol is to transfer information between cars. Information transfer between two cars can be interpreted as synchronizing two local events between the cars. Events are indexed by elements ϵ from a set $\mathcal{E}$. An event ϵ may be associated with data d from a set $\mathcal{D}$, written $\epsilon(d)$. $\mathcal{D}$ contains an auxiliary element
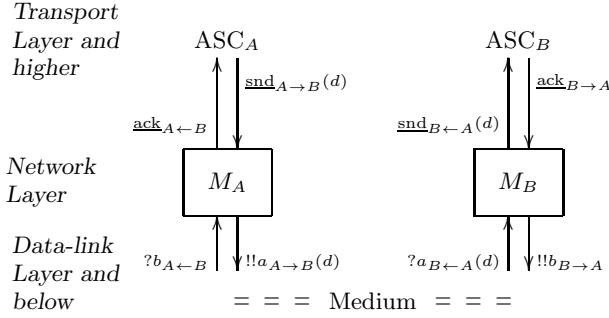
Figure 2: Two cars A and B communicating with each other: A sends a message (by calling $\underline{\text{snd}}_{A \to B}(d)$) and B responds with an acknowledgement on reception (by calling $\underline{\text{ack}}_{B \to A}$). The transmission medium and the ASCs are the environment of the CSAs.

$\perp$, indicating the absence of data. We simply write ϵ for notational convenience if $\delta = \perp$ in $\epsilon(d)$.

A *local event* is an event that is triggered either by the ASC of a car (an *environment-triggered event* from the point of view of the CSA), or by the CSA of a car itself (a *system-triggered event*). An environment-triggered event that is initiated by the ASC at car A and is to be synchronized with car B is written as $\underline{\epsilon}_{A \to B}(d)$. It is synchronized with the corresponding system-triggered event $\underline{\epsilon}_{B \leftarrow A}(d)$ by the CSA of car B. The sets of environment-triggered and system-triggered events are written as $\mathcal{E}_E$ and $\mathcal{E}_S$ respectively.

If $\underline{\epsilon}_{A \to B}(d)$ on car A is synchronized with $\underline{\epsilon}_{B \leftarrow A}(d)$ on car B, then the data d is transferred from A to B. This is summarized as a single *global event* $\epsilon_{A \to B}(d)$ (note the absence of the line under ϵ). The set of global events is denoted by $\mathcal{E}_G$. A protocol specification defines a desired temporal order on such global events. Since global events involve several cars, a protocol specification is *centralized*, i.e., it is assumed that the actions of all cars can be influenced independently by a single controller.

Synchronization is achieved by sending messages across a shared transmission medium. A CSA interacts with the medium by transmitting messages and waiting for reception of messages. A message *transmission* is indicated by "!!", while a *reception* is indicated by "?" prefixed to a message.

The interaction with the higher-level ASC is managed by *calls* and *upcalls*. A call is initiated by the ASC and causes an environment-triggered event in the CSA. An upcall is initiated by a system-triggered event in the CSA.

Example 1 (Continued) Consider again the intersection problem in Fig. 1. As car A is approaching the intersection, it needs to establish whether it is safe to enter the intersection. It therefore wants to establish a communication with any car that might pose a safety hazard.

Car A needs to communicate with car B to find out if B is willing and able to stop or whether A should attempt an emergency brake. Each car is assigned a unique address for labelling messages, so that when a car receives a message, it knows whether it is the intended destination. We assume that the ASC at A provides its CSA with the address of B, so that a P2P communication with B can be established.

This communication scenario is shown in Fig. 2, where the CSA associated with each car is shown as a box. Data d is transferred from A to B, and B should send an acknowledgement back to A. Sending d from A to B is done by synchro-

(a) Centralized.

(b) Distributed.

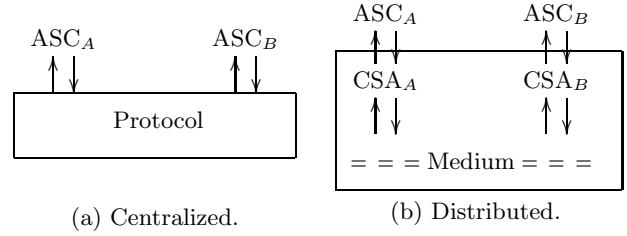**Figure 3: The protocol on two levels of detail.**

nizing the local events $\underline{\text{snd}}_{A \to B}(d)$ and $\underline{\text{snd}}_{B \leftarrow A}(d)$, while the acknowledging synchronizes $\underline{\text{ack}}_{A \to B}$ with $\underline{\text{ack}}_{B \leftarrow A}$. A call by an ASC triggers the corresponding environment-triggered event in the CSA on the same car, while an upcall is initiated by the CSA when some system-triggered event requires the attention of the higher level. □

3.2 Quality of Service

Depending on the application, it is be necessary to guarantee that a transmission is completed with certain requirements on particular aspects such as end-to-end delay, message drop probability or bandwidth. These aspects are called Quality of Service (QoS). We are concerned with automatically implementing protocols that guarantee that certain requirements on QoS are met.

Whether QoS requirements can be satisfied depends on the properties of the medium used to transmit messages over. In our work we assume minimal capabilities for a transceiver, so the only way to satisfy QoS requirements is to select the appropriate frequency and number of retransmissions for messages.

Also, when finding the CSAs that satisfy the protocol, we have to take into account that the performance of the transmission medium typically degrades as consequence of messages being transmitted. Moreover, a car cannot predict the behavior of the transmission medium merely on the basis of its own actions, since there might be other cars sharing the same medium that exhibit unpredictable behavior from the point of view of the car. In Example 1, while cars A and B are communicating, other cars might be trying to transmit messages itself, leading to a degradation in performance for A and B that neither car can predict.

We restrict the package drop probability Δ of the transmission medium by assuming that it is below a given threshold probability δ at all times. We write this as $\Box(\Delta \leq \delta)$, where "$\Box$" is the always operator "$\Box$" of linear temporal logic (LTL) Hence, a full specification in the framework can be stated as an assumption/guarantee specification [1] $\Box(\Delta \leq \delta) \to \varphi$, where a protocol specification φ only has to hold as long as the assumption that at all times $\Delta \leq \delta$ is satisfied.

A straightforward extension to take time into account would be to consider each (re)transmission to take up some amount of time $\mathcal{T}$. We can then include another assumptions of the form $\Box(\mathcal{T} \leq \tau_{max})$, where τ_{max} is an upper bound on the transmission time.

4. TECHNICAL APPROACH

In this section the concepts described above are formalized.

4.1 Protocol Specification Language

In a protocol specification, the protocol is viewed as a

single component interacting with the ASCs, cf. Fig. 3(a). Only the interaction across the interface between ASCs and CSAs is specified. The CSAs, representing an implementation of a protocol, then interface with the lower level transmission medium in order to provide the required services to the higher level. In this way, the ASCs never come in direct contact with the transmission medium. As introduced above, a specification of the protocol is given as a temporal (partial) order of global events. A global event is any event involving the interaction of several cars, such as a message transmission (involving both the sender and the receiver).

Hence, a protocol specification can be seen as an allowed set of sequences of global events. Moreover, each sequence is tagged with a QoS requirement, which, in our case simply is the required probability of the sequence to be synchronized correctly. In order to avoid having to write a list of sequences with potentially many global events repeating, we use the following temporal logic-like language to define *protocol specifications*:

$$\varphi ::= e^p \,|\, e \to \bigcirc \varphi \,|\, \varphi \vee \varphi,$$

where e^p is a global event $e \in \mathcal{E}_G$ together with a probability p indicating the required QoS. Extensions of this specification language can also include time, bandwidth or other QoS requirements in the same way in the specification. A *specification* is a protocol specification together with an environment assumption, in our case an upper bound δ on the drop probability Δ.

Example 1 (Continued) The protocol described in the intersection example of Fig. 1 can be specified as

$$\varphi = \mathrm{snd}_{A \to B}(d) \to \bigcirc(\mathrm{ack}_{B \to A}^{p_1} \vee \mathrm{nack}_{B \to A}^{p_2}), \qquad (1)$$

which can be illustrated as a tree as in Fig. 4. In the numerical results presented later for this example we will use different values for p_1 and p_2.

The results of the synthesis also depend on the drop probability bound δ. A complete specification that includes the assumptions on the transmission medium dynamics would be

$$\Box(\Delta \le \delta) \to \varphi. \qquad (2)$$

□

Since we are interested in QoS requirements over the drop probability of the transmission medium, a probability p on e^p labels each leaf of the tree representing a protocol specification, specifying the desired probability of the (unique) sequence of global events σ occurring that leads to the leaf. We call a sequence σ with a probability p attached to it a *p-sequence* and write $(\sigma)^p$. The semantics of the protocol specification language is defined by a satisfaction relation: If a p-sequence σ of global events satisfies the protocol specification φ, this is written as $(\sigma)^p \models \varphi$.

We first develop an intuitive understanding of a sequence $\sigma = e_1 e_2 \dots$ satisfying a specification φ. Recall that a protocol specification only takes the interfaces between CSAs and ASCs into account, and hence views the protocol implementation as a monolithic entity as in Fig. 3(a). Each global event $e_i = \epsilon_{x \to y}(d)$ in the sequence σ is interpreted as the synchronization of an environment-triggered event $\underline{\epsilon}_{x \to y}(d)$ and a system-triggered event $\underline{\epsilon}_{y \leftarrow x}(d)$. The ASC on car x triggers $\underline{\epsilon}_{x \to y}(d)$ by a call to its CSA. The intention is that the corresponding system-triggered event $\underline{\epsilon}_{y \leftarrow x}(d)$ is synchronized with that event in the CSA on car y (and an upcall
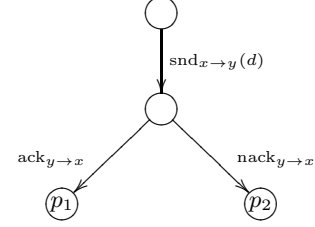

Figure 4: Visualization of protocol specification φ in (1)**, which establishes a partial order between the global events** $\mathrm{snd}_{x \to y}(d)$**,** $\mathrm{ack}_{y \to x}$ **and** $\mathrm{nack}_{y \to x}$**, represented by the order of the edge labels.**

is made to its ASC). The synchronization is correct if after an environment-triggered event $\underline{\epsilon}_{x \to y}(d)$, the first system-triggered event is $\underline{\epsilon}_{y \leftarrow x}(d)$, i.e. no other system-triggered event is interleaved between them. Note that environment-triggered events that do not correspond to global events in the specification may be interleaved, as the protocol has no control over the higher level. Then, the statement $(\sigma)^p \models \varphi$ expresses that σ satisfies the partial order defined in φ and has a high enough probability p attached to it.

We now formally define $\models$ recursively on the structure of a protocol specification φ (cf. (1)):

$$
\begin{array}{llll}
(e)^p & \models e^q & \Leftrightarrow & p \ge q \\
(e, \sigma)^p & \models e \to \bigcirc \varphi & \Leftrightarrow & (\sigma)^p \models \varphi \\
(\sigma)^p & \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & (\sigma)^p \models \varphi_1 \text{ or } (\sigma)^p \models \varphi_2,
\end{array} \qquad (3)
$$

where adding a global event e to the head of a sequence σ is written as e, σ. Under these semantics a protocol specification is satisfied exactly by those sequences of global events that both obey the partial order induced by φ and that have a sufficiently high probability attached to them.

4.2 Communication Service Automata

We are interested in finding a way to implement a protocol in a distributed manner by finding CSAs for the cars so that their joint execution satisfies the protocol specification. That is, the implementation of the protocol must use the transmission medium in order to guarantee the required services to the higher level, see Fig. 3(b). A set $\mathcal{M}$ of CSAs satisfies a protocol specification if it produces only the allowed sequences of global events, and these with a sufficiently high probability. In this section we make precise the concept of a CSA and define its semantics in the next section.

A CSA is a finite state machine with labelled transitions, which is similar to a protocol entity specification used by Ishida et al. [12]. Transition labels either indicate which actions should be executed when the transition is taken, or impose conditions on a transition. A transition for which all conditions are satisfied is called *enabled*. The labels that are available to the synthesis method are explained below, and transitions are typically labelled with combinations of labels.

Firstly, interaction with the higher-level ASCs is encoded by edges labelled with environment-triggered and system-triggered events. We also introduce two special system-triggered events "fail" and "success" to $\mathcal{E}_S$ that have no corresponding environment-triggered events to be synchronized with. The purpose of these events is merely to inform the ASC of the outcome of a transmission. A fail event indicates that allowable retransmission count is exceeded, while
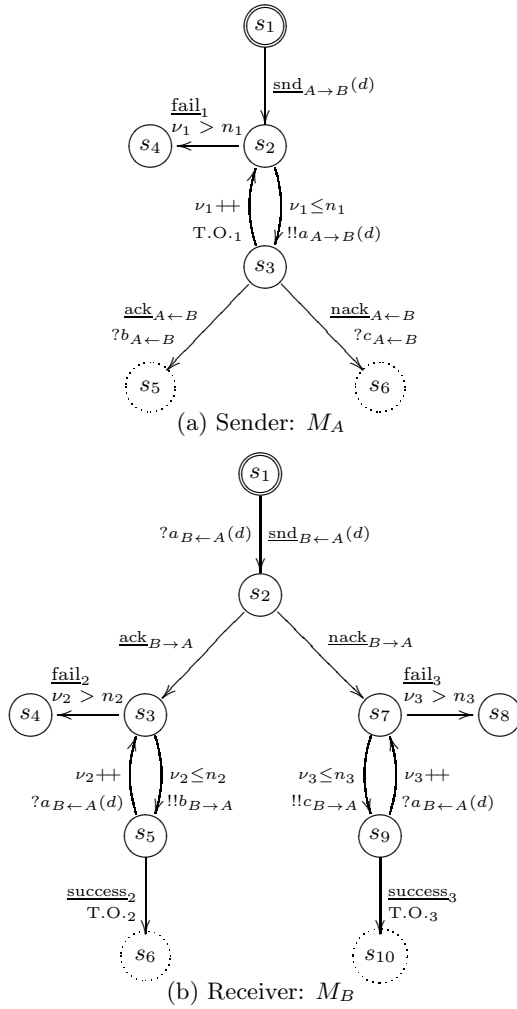
(a) Sender: M_A

s_1

$\underline{\text{snd}}_{A\to B}(d)$

$\underline{\text{fail}}_1$
$\nu_1 > n_1$

$s_4 \leftarrow s_2$

$\nu_1{+}{+}$ $\nu_1 \leq n_1$
T.O._1 $!!a_{A\to B}(d)$

s_3

$\underline{\text{ack}}_{A\leftarrow B}$ $\underline{\text{nack}}_{A\leftarrow B}$
$?b_{A\leftarrow B}$ $?c_{A\leftarrow B}$

s_5 s_6

(b) Receiver: M_B

s_1

$?a_{B\leftarrow A}(d)$ $\underline{\text{snd}}_{B\leftarrow A}(d)$

s_2

$\underline{\text{ack}}_{B\to A}$ $\underline{\text{nack}}_{B\to A}$

$\underline{\text{fail}}_2$ $\underline{\text{fail}}_3$
$\nu_2 > n_2$ $\nu_3 > n_3$

$s_4 \leftarrow s_3$ $s_7 \to s_8$

$\nu_2{+}{+}$ $\nu_2 \leq n_2$ $\nu_3 \leq n_3$ $\nu_3{+}{+}$
$?a_{B\leftarrow A}(d)$ $!!b_{B\to A}$ $!!c_{B\to A}$ $?a_{B\leftarrow A}(d)$

s_5 s_9

$\underline{\text{success}}_2$ $\underline{\text{success}}_3$
T.O._2 T.O._3

s_6 s_{10}

Figure 5: Two CSAs that realize the protocol specification in (1). **The transitions are labelled with broadcast messages (e.g.** $!!a_{A\to B}(d)$**), receptions (e.g.** $?a_{B\leftarrow A}(d)$**), and local events (e.g.** $\underline{\text{snd}}_{A\leftarrow B}(d)$ **and** $\underline{\text{snd}}_{B\to A}(d)$**). Initial states and final states are shown as doubled and dotted circles respectively. In each retransmission loop, a retransmission counter** ν_i **is increased by one on each timeout or reception. The transmissions are conditioned on the retransmission counters** ν_i**. If the counter is exceeded, indicated by** $\nu_i > n_i$**, a** $\underline{\text{fail}}_i$ **occurs.**

the $\underline{\text{success}}$ event indicates a successful transmission to the ASC triggering the last global event.[1]

Secondly, to interact with the transmission medium, transitions can be labelled with message transmissions and receptions. Each message has a unique identifier m. A broadcast message is written as $!!m_{x\to y}(d)$, where x and y are the source and destination respectively, and $d \in \mathcal{D}$ is the data transmitted. It is read as "send m with data d to y from source x." Similarly, a reception is written as $?m_{y\leftarrow x}(d)$, where x, y and d have the same interpretation as for a broadcast message. It is read as "receive m with data d from x destined for y." Again, if $d = \perp$, the parameter is not written.

---

[1]This is necessary since no response from another ASC can indicate completion of the transmission.

Define $\mathcal{B}$ and $\mathcal{R}$ to be the set of broadcasts and receptions respectively.

Lastly, we introduce labels for internal actions of a CSA. In order to satisfy the QoS requirements of the protocol specification, it may be necessary to allow the retransmission of messages. To this end, we define a set of *variables* $\mathcal{V}$ over $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ that act as *retransmission counters*. We will construct the CSAs in such a way that for each message m that might be retransmitted, after a transmission $!!m_{x\to y}(d)$, either a reception of some other message is expected or a *timeout* "T.O." may occur. On the timeout, the retransmission counter ν of the message is increased by one. If ν exceeds its *retransmission bound* $n \in \mathbb{N}_0$, the transmission fails, causing a $\underline{\text{fail}}$ event and a corresponding upcall informing the ASC. We write an *update* of a variable $\nu \in \mathcal{V}$ as $\nu{+}{+}$, and denote the set of updates by $\mathcal{U} = \{\nu{+}{+} | \nu \in \mathcal{V}\}$. Further, a transition may be labelled by a *condition* on a retransmission counter, which can be either of the form $\nu \leq n$ or $\nu > n$. The set of conditions is defined as $\mathcal{C} = \{\nu \bowtie n | \nu \in \mathcal{V}, \bowtie \in \{\leq, >\}, n \in \mathbb{N}_0\}$

When synthesizing a CSA from a protocol specification, each transition can be of one of seven kinds, depending on the labels: A environment-triggered event, a conditional system-triggered event, a timeout with a system-triggered event, a timeout with update, a conditional broadcast, a reception with a system-triggered event or a reception with update. Hence, the set of labels is $\Sigma = \mathcal{E}_E \cup (\mathcal{E}_S \times \mathcal{C}) \cup (\{\text{T.O.}\} \times \mathcal{E}_S) \cup (\{\text{T.O.}\} \times \mathcal{U}) \cup (\mathcal{B} \times \mathcal{C}) \cup (\mathcal{R} \times \mathcal{E}_S) \cup (\mathcal{R} \times \mathcal{U})$ for the respective cases. The set of such transition labels is denoted by Σ. A CSA M is a quintuple

$$M \triangleq \langle S, \mathcal{V}, s^{init}, S^f, T \rangle,$$

where S is a set of *states* labelled by valuations of variables $\mathcal{V}$, $s^{init} \in S$ is the *initial state*, $S^f \subseteq S$ is the set of *final states*, and $T : S \times \Sigma \to S$ is the (partial) *transition function*.

Example 1 (Continued) The pair of CSAs shown in Fig. 5 represents one potential implementation of the protocol specification in (1). The transmissions and receptions are introduced in order to ensure that the QoS requirements as defined in the specification is preserved by the CSAs that can only communicate over the transmission medium. For example, the sender M_A may retransmit the message $!!a_{A\to B}(d)$ up to n_1 times in case of repeated timeouts to increase the likelihood of a successful transmission, in order to meet the specification. $\square$

4.3 Semantics of CSAs

In the semantics of a CSA, we want to reflect that a car should be able to execute it as a controller for its wireless transceiver.

Decisions when to make transitions should be based only on information available locally. For example, a transition labelled by a reception $?m_{x\leftarrow y}$ is taken only when a message m arrives that has x as its destination and y as its source. Since a CSA is executed locally on a car, we first define the *local semantics* of a single CSA. This describes how a CSA operates in isolation when receiving calls from the ASC on the same car, and messages from the transmission medium, cf. Fig. 3(b). We then define the *global semantics* of several CSAs that operate together, which requires to take the transmission medium dynamics into account, cf. Sec. 3.2. Hence, the global semantics can be interpreted as defining

[env]
$$\frac{T(s, \underline{\epsilon}_{y\to z}(d)) \overset{\mathcal{V}}{\approx} s'}{\langle \rho, s\rangle \xrightarrow{\;\text{e}\;}_{M_y} \langle \rho + \underline{\epsilon}_{y\to z}(d), s'\rangle}$$

[sys-c]
$$\frac{T(s, (\underline{\epsilon}_{y\leftarrow z}(d), \gamma)) \overset{\mathcal{V}}{\approx} s' \quad \gamma(s)}{\langle \rho, s\rangle \xrightarrow{\;\text{e}\;}_{M_y} \langle \rho + \underline{\epsilon}_{y\leftarrow z}(d), s'\rangle}$$

[to-sys]
$$\frac{T(s, (\text{T.O.}, \underline{\epsilon}_{y\leftarrow z}(d))) \overset{\mathcal{V}}{\approx} s'}{\langle \rho, s\rangle \xrightarrow{\;\text{t}\;}_{M_y} \langle \rho + \text{T.O.} + \underline{\epsilon}_{y\leftarrow z}(d), s'\rangle}$$

[to-upd]
$$\frac{T(s, (\text{T.O.}, \nu{+}{+})) \overset{\mathcal{V}\setminus\{\nu\}}{\approx} s' \quad s'(\nu) = s(\nu) + 1}{\langle \rho, s\rangle \xrightarrow{\;\text{t}\;}_{M_y} \langle \rho + \text{T.O.}, s'\rangle}$$

[b-c]
$$\frac{T(s, (!!m_{y\to z}(d), \gamma)) \overset{\mathcal{V}}{\approx} s' \quad \gamma(s)}{\langle \rho, s\rangle \xrightarrow{\;\text{e}\;}_{M_y} \langle \rho + !!m_{y\to z}(d), s'\rangle}$$

[r-sys]
$$\frac{T(s, (\varsigma, \underline{\epsilon}_{y\leftarrow z}(d))) \overset{\mathcal{V}}{\approx} s' \quad \varsigma = ?m_{y\leftarrow z}(d)}{\langle \rho + \varsigma, s\rangle \xrightarrow{\;\text{r}\;}_{M_y} \langle \rho + \varsigma + \underline{\epsilon}_{y\leftarrow z}(d), s'\rangle}$$

[r-upd]
$$\frac{T(s, (?m_{y\leftarrow z}(d), \nu{+}{+})) \overset{\mathcal{V}\setminus\{\nu\}}{\approx} s' \quad s'(\nu) = s(\nu) + 1}{\langle \rho + ?m_{y\leftarrow z}(d), s\rangle \xrightarrow{\;\text{r}\;}_{M_y} \langle \rho + ?m_{y\leftarrow z}(d), s'\rangle}$$

Table 1: Local semantic rules for deducing behavior of a single CSA M_y.

the behavior of the protocol in Fig. 3(a). An example of how the semantics are used is presented in Sec. 5.2.3.

4.3.1 Deduction Rules

For ease of presentation, the semantics are defined as a set of *deduction rules*. A deduction rule is of the form

$$\frac{H_1 \quad H_2 \quad \dots \quad H_n}{C},$$

which is the same as $\bigwedge_{i=1}^{n} H_i \Rightarrow C$, i.e. the *conclusion C* follows from the *hypotheses* $H_1, H2, \dots, H_n$. A deduction rule can be applied if all its hypotheses hold. Rules can either be applied forward, starting from one or several axioms, or backwards, starting from a conclusion. Forward application corresponds to simulation, while backwards application corresponds to verification.

4.3.2 Notation

We first introduce some notation to make the statement of the rules more compact. Retransmission uses conditional transitions and updating of variables. The value v of a variable $\nu \in \mathcal{V}$ in a state s is written $s(\nu) = v$. In the initial state s^{init} all variables valuate to zero. A condition $\gamma = \nu \bowtie n$ is satisfied in state s, written $\gamma(s)$, if and only if $s(\nu) \bowtie n$. Two states s and s' are equivalent on their values of the variables in $V \subseteq \mathcal{V}$, written $s \overset{V}{\approx} s'$, if and only if $\forall \nu \in V. s(\nu) = s'(\nu)$. We write $T(s, \varsigma) \overset{V}{\approx} s'$ if and only if $T(s, \varsigma) = s'$ and $s \overset{V}{\approx} s'$, where $\varsigma \in \Sigma$ may stand for any transition label. Furthermore, we use the $+$ operator to append an element to the end of a sequence.

4.3.3 Local Semantic Rules

The local semantics is defined by a relation $\longrightarrow_M \subseteq (\Sigma^* \times S) \times (\Sigma^* \times S)$ between sequences of transition labels and CSA states. The statement $\langle \rho, s\rangle \longrightarrow_M \langle \rho', s'\rangle$ means

that M at state s transforms ρ into ρ' by making a single transition to state s'. It holds if and only if it is deducible via the rules given in Table 1. To make the statement of the global semantics simpler, we may label the relation by a superscript to distinguish which rules are applied. For example, $\xrightarrow{\;\text{e}\;}_M$ indicates that either the rule [env], [sys-c] or [b-c] are applied. If the superscript is omitted, any rule may be applied.

We explain the [env] rule for CSA M_y in detail, the other rules are similar. The hypothesis $T(s, \underline{\epsilon}_{y\to z}(d)) \overset{\mathcal{V}}{\approx} s'$ expresses that M_y must allow a transition from s that is labelled with the environment-triggered event $\underline{\epsilon}_{y\to z}(d)$ and leads to a state s' in which the values of all variables in $\mathcal{V}$ are the same as in s (i.e. there is no update). If this hypothesis is satisfied, M_y at state s transforms ρ into ρ' by making a transition to state s'.

The [env] rule can be applied at any point if a transition labelled by an environment-triggered event is enabled. It is not dependent on an input from the higher level ASC. Stating the rule this way is sufficient for our presentation, but it can be substituted by

[env']
$$\frac{T(s, \underline{\epsilon}_{y\to z}(d)) \overset{\mathcal{V}}{\approx} s'}{\langle \rho + \underline{\epsilon}_{y\to z}(d), s\rangle \xrightarrow{\;\text{e}\;}_{M_y} \langle \rho + \underline{\epsilon}_{y\to z}(d), s'\rangle}$$

to explicitly require an input to be able to apply the rule. The input is the last element in the sequence, which is an environment-triggered event $\underline{\epsilon}_{y\to z}(d)$, indicating that the ASC must have made the corresponding call. This input may be placed in the sequence (i.e. added as last element) by the global semantics, similar to the inputs for the [r-sys] and [r-upd] rules.

The [sys-c] rule places no restriction on the input and contains as an additional hypothesis that the condition γ must be satisfied in state s. The system-triggered event $\underline{\epsilon}_{y\leftarrow z}(d)$ gives rise to an upcall. Such outputs can be read off the last element of the deduced sequence and hence are not modelled explicitly in these rules. The [to-sys] rule can be applied for a transition labelled with a timeout T.O. and a system-triggered event $\underline{\epsilon}_{y\leftarrow z}(d)$. In the [to-upd] rule the value of the variable $\nu \in \mathcal{V}$ is incremented by one as the transition is taken. Hence we use the operator $\overset{\mathcal{V}\setminus\{\nu\}}{\approx}$, since $\overset{\mathcal{V}}{\approx}$ would indicate that all variables in $\mathcal{V}$ retain their values as the transition is taken. The [b-c] rule can be applied for a conditional broadcast message. The outgoing message again can be obtained from the last element of the deduced sequence. Rules [r-sys] and [r-upd] require the reception $?m_{y\leftarrow z}(d)$ to occur, hence the rules require the corresponding input.

Each CSA may deduce a set of sequences of events by transitioning between its states. Decisions between environment-triggered events, receptions and timeouts are made by inputs (or the absence thereof) received either from the higher level ASC or the lower level transmission medium. These inputs can only be generated by the global semantics.

4.3.4 Transmission Medium Modelling

We define the global semantics by modelling how the transmission medium operates. That is, we define the behavior of the protocol in Fig. 3(a) by composing the behavior of the CSAs in Fig. 3(b) and abstracting away all lower level detail. The global semantics defines when inputs to a CSA are received from the transmission medium, and restricts the valid interleavings of locally generated sequences. The

$$[\text{trans}] \quad \dfrac{\langle \rho + ?m_{z \leftarrow y}(d), s_z \rangle \xrightarrow{\ \ r\ \ }_{M_z} \langle \rho', s'_z \rangle}{\varsigma\ =\!!m_{y \to z}(d)} \\ \overline{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho')^{(1-\delta)p}, s[z \leftarrow s'_z], z \rangle}$$

$$[\text{drop}] \quad \dfrac{\langle \rho + ?m_{z \leftarrow y}(d), s_z \rangle \xrightarrow{\ \ r\ \ }_{M_z} \langle \rho', s'_z \rangle}{\varsigma\ =\!!m_{y \to z}(d)}{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho)^{\delta p}, s, z \rangle}$$

$$[\text{nacc}] \quad \dfrac{\neg(\langle \rho + ?m_{z \leftarrow y}(d), s_z \rangle \xrightarrow{\ \ r\ \ }_{M_z} \langle \rho', s'_z \rangle)}{\varsigma\ =\!!m_{y \to z}(d) \qquad \langle \rho, s_x \rangle \xrightarrow{\ \ }_{M_x} \langle \rho', s'_x \rangle}{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho)^p, s, z \rangle}$$

$$[\text{pr-e}] \quad \dfrac{\varsigma \neq\!!m_{y \to z}(d) \quad \langle \rho + \varsigma, s_y \rangle \xrightarrow{\ \ e\ \ }_{M_y} \langle \rho', s'_y \rangle}{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho')^p, s[y \leftarrow s'_y], y \rangle}$$

$$[\text{pr-t}] \quad \dfrac{\varsigma \neq\!!m_{y \to z}(d) \quad \neg(\langle \rho + \varsigma, s_y \rangle \xrightarrow{\ \ e\ \ }_{M_y} \langle \rho'', s''_y \rangle)}{\langle \rho + \varsigma, s_y \rangle \xrightarrow{\ \ t\ \ }_{M_y} \langle \rho', s'_y \rangle}{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho')^p, s[y \leftarrow s'_y], y \rangle}$$

$$[\text{npr}] \quad \dfrac{\varsigma \neq\!!m_{y \to z}(d) \quad \neg(\langle \rho + \varsigma, s_y \rangle \xrightarrow{\ \ e,t\ \ }_{M_y} \langle \rho'', s'_y \rangle)}{\langle \rho + \varsigma, s_x \rangle \xrightarrow{\ \ }_{M_x} \langle \rho', s'_x \rangle}{\langle (\rho + \varsigma)^p, s, y \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho')^p, s[x \leftarrow s'_x], x \rangle}$$

Table 2: Global semantics for deducing behavior of several CSAs $\mathcal{M}$. In particular, the rules [trans], [drop] and [nacc] model the transmission medium.

medium therefore also acts as an arbiter or scheduler of transitions.

In the global semantics, we are interested in ensuring that several CSAs together satisfy the global protocol specification by interacting with each other. We therefore define the semantics of a list of CSAs $\mathcal{M} = \langle M_A, M_B \ldots \rangle$ that is executed together on the respective set of cars $\mathfrak{C} = \{A, B, \ldots\}$. Each execution starts with all CSAs in $\mathcal{M}$ being in their initial state $s^{init} = \langle s_A^{init}, s_B^{init}, \ldots \rangle$ and making only transitions allowed by the semantics. Only a single sequence $\rho \in \Sigma^*$ is deduced, which is an interleaving of the sequences deduced locally. The deduction rules also express that the medium transmits messages only with a given probability. Hence, the deduced sequence ρ is tagged with a probability p, indicating how likely it occurs.

Not only do the global semantics define how messages are transmitted, also the valid interleavings of locally deduced sequences are restricted. To motivate this, consider in Fig. 5 the execution of the environment-triggered event $\underline{\text{ack}}_{B \to A}$. Since time is abstracted away, the transition may be delayed by an arbitrary amount of time. However, then the retransmission loop in the sender A cannot reliably increase the likelihood of a successful execution, since the timeout transition can also be taken at any time. In order to prevent this from happening, the global semantics ensure that transitions that are not timeouts or receptions are taken immediately if enabled. Hence, only one CSA is allowed to make transitions until a timeout or reception is encountered. Then any CSA may make a transition. This is incorporated in the global semantics by always prioritizing one CSA is to make a transition. If this CSA has no transition enabled, any other CSA may make a transition.

4.3.5 Global Semantic Rules

In the global semantics, we are interested in ensuring that several CSAs together satisfy the global protocol specification by interacting with each other. The transmission medium therefore acts as an arbiter or scheduler of transitions. Hence, we can think of the global behavior of several CSAs $M_1, M_2, \ldots$ as an interleaving ρ of the locally generated sequences $\rho_1, \rho_2, \ldots$ of the respective CSAs. Messages are only transmitted with a certain probability. Hence, the sequence ρ is tagged with a probability p, indicating how likely it occurs.

The relation $\langle (\rho)^p, s, x \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho')^{p'}, s', x' \rangle$ defines the global semantics according to the rules in Table 2. It means that $\mathcal{M}$ with drop probability δ at state s transforms ρ into ρ' by making a transition to state s' while the priority changes from M_x to $M_{x'}$. In the statement of the rules, updating the zth element s_z in state $s = \langle s_A, s_B, \ldots, s_z, \ldots \rangle$ with s'_z is written as $s[z \leftarrow s'_z]$.

The [trans], [drop] and [nacc] rules define the transmission medium dynamics. If the last deduced element in the sequence is a broadcast message, i.e. $\varsigma =\!!m_{y \to z}(d)$, the medium tries to transmit. An application of the [trans] rule models a successful message transmission. This only occurs if the CSA for which the message was destined, M_z makes a transition labelled with the corresponding reception. That is, $\langle \rho + ?m_{z \leftarrow y}(d), s_z \rangle \xrightarrow{\ \ r\ \ }_{M_z} \langle \rho', s' \rangle$ is only satisfied if M_z can execute [r-sys] or [r-upd]. Since a message transmission occurs with probability $1 - \delta$, the probability with which the sequence ρ' is tagged in the conclusion of [trans] is $(1 - \delta)p$.

An application of the [drop] rule models a dropped message. It has exactly the same hypotheses as [trans], but its conclusion reflects that no progress has been made. The sequence ρ is tagged with δp due to the message drop probability δ. Note that the priority is at the source CSA M_y, which may now execute a timeout transition (if enabled).

The [nacc] rule is applied when a message should be transmitted, but the destination CSA has no transition enabled that is labelled by the corresponding reception. Similar to the [drop] rule, no progress is made. Also, the probability of the deduced sequence is not affected.

The [pr-e], [pr-t] and [npr] rules may be applied if the last element of the sequence is not a message transmission. Then the transmission medium is inactive, and and the CSA that is currently prioritized may execute: If a transition that is not a timeout or reception is enabled, then [pr-e] is applied. If a timeout transition is enabled, then [pr-t] is applied. The [npr] rule may only applied if the currently prioritized CSA has no such transitions enabled. In this case, any CSA M_x may execute.

The transitive closure $\langle (\rho)^p, s, x \rangle \Longrightarrow^*_{\mathcal{M}(\delta)} \langle (\rho')^{p'}, s', x' \rangle$ denotes that $\langle \rho, s \rangle$ is transformed into $\langle \rho', s' \rangle$ in an arbitrary number of deduction steps. The CSAs $\mathcal{M}$ execute by starting in state s^{init} with an empty 1-sequence $(\bullet)^1$ and any CSA M_x prioritized. Valid deductions are the tuples $\langle (\rho)^p, s, y \rangle$ for which $\langle (\bullet)^1, s^{init}, x \rangle \Longrightarrow^*_{\mathcal{M}(\delta)} \langle (\rho)^p, s, y \rangle$.

Note that an example of how the local and global semantic rules are used is included in Sec. 5.2.3.

4.3.6 Global Event Sequences

By applying the deduction rules, sequences over both environment-triggered and system-triggered events, broadcasts, receptions and timeouts can be obtained from a set of CSAs. Since protocol specifications are over global events, we need to extract the synchronizations of local events in the sequences generated by a set of CSAs. We therefore define

the projection function $[\![\cdot]\!] : \Sigma^* \to \mathcal{E}_G^*$ to find a sequence over global events from ρ. It is defined by

$$[\![\bullet]\!] \triangleq \bullet$$

$$[\![\rho + r]\!] \triangleq \begin{cases} [\![\rho]\!] + \underline{\epsilon}_{y \to z}(d) & \text{if } r = \underline{\epsilon}_{y \to z}(d) \\ [\![\rho']\!] + \epsilon_{y \to z}(d) & \text{if } [\![\rho]\!] = [\![\rho']\!] + \underline{\epsilon}_{y \to z}(d) \\ & \quad \text{and } r = \underline{\epsilon}_{z \leftarrow y}(d) \\ [\![\rho]\!] & \text{otherwise.} \end{cases}$$

We use the projection function $[\![\cdot]\!]$ to express whether a set of CSAs satisfies a protocol specification φ under the environment assumptions $\square(\Delta \leq \delta)$.

4.4 Correctness

In this section we define correctness of a protocol's implementation in form of a set of CSAs $\mathcal{M}$ with respect to a specification $\square(\Delta \leq \delta) \to \varphi$. If $\mathcal{M}$ satisfies this specification this is written as $\mathcal{M} \vdash \square(\Delta \leq \delta) \to \varphi$. Correctness depends on the probability of sequences σ being synchronized correctly by the CSAs $\mathcal{M}$ if the transmission medium's drop probability Δ is bounded from above by δ, i.e. it satisfies $\square(\Delta \leq \delta)$. If this assumption on the transmission medium is not satisfied, the specification $\square(\Delta \leq \delta) \to \varphi$ is trivially satisfied by any set of CSAs. However, this case is useless in practice, as the protocol will not deliver the data with the required QoS.

4.4.1 Definitions

Given a protocol specification φ, correctness of an implementation depends on whether all CSAs involved in synchronizing a sequence of global events are in a final state. We therefore define the set of *globally final* states $S_{\mathcal{M}}^f$ to include all tuples of states $\langle s_A, s_B, \ldots \rangle \in \prod_{x \in \mathfrak{C}} S_x$ so that if there is some sequence involving CSAs $x, y, \ldots$, the states $s_x, s_y, \ldots$ are actually final states from $S_x^f, S_y^f, \ldots$.

We say that a p-sequence $(\rho)^p$ is *generated* by a set of CSAs $\mathcal{M}$ and drop probability δ, and write $(\rho)^p \models \mathcal{M}(\delta)$, if it can be deduced by the rules in Table 1 and Table 2 and the deduction ends in a globally final state $s^f \in S_{\mathcal{M}}^f$. Formally,

$$(\rho)^p \models \mathcal{M}(\delta) \Leftrightarrow$$
$$\exists s^f \in S_{\mathcal{M}}^f. \exists x, y \in \mathfrak{C}. \langle (\bullet)^1, s^{init}, x \rangle \Longrightarrow_{\mathcal{M}(\delta)}^* \langle (\rho)^p, s^f, y \rangle.$$

As noted above in Sec. 3.1, a p-sequence $(\sigma)^p$ satisfies a specification φ exactly if the probability p that all (global) events in σ are correctly synchronized is high enough given that the corresponding environment-triggered events are all triggered through calls by the higher level ASCs. For a set of CSAs therefore to satisfy a specification, it is required that the synchronization of events in each sequence is performed with high enough probability.

4.4.2 Correctness Condition

The important criterion for correctness is not whether a sequence ρ is generated, but whether the QoS requirements are satisfied. This is because the decisions between environment-triggered events (which essentially generate the sequence) are made by the higher level ASCs, over which a CSA has no control. For example, in Fig. 5, the receiver CSA has no control over whether $\underline{\text{ack}}_{B \to A}$ or $\underline{\text{nack}}_{B \to A}$ is triggered by its ASC in state s_2. In our case the only QoS requirement is the probability of all global events being correctly synchronized, so the question for correctness becomes:

Given that the ASCs trigger the events necessary to generate σ, how likely is it that all synchronizations performed?

$\mathcal{M}$ might generate a given sequence σ in many different ways, since several sequences ρ deducible by the rules in Table 1 might satisfy $[\![\rho]\!] = \sigma$. For a sequence σ, we evaluate the sum of all probabilities p for distinct sequences ρ that satisfy

$$\text{cond}(\sigma, p, \delta, \mathcal{M}) \triangleq ([\![\rho]\!] = \sigma \wedge (\rho)^p \models \mathcal{M}(\delta)),$$

and get the probability

$$r(\sigma, \delta, \mathcal{M}) \triangleq \sum_{\text{cond}(\sigma, p, \delta, \mathcal{M})} p,$$

expressing the likelihood of the events in the sequence σ being correctly synchronized when executing all CSAs in $\mathcal{M}$ in parallel (i.e. using the global semantics). *Correctness* then is expressed by

$$\mathcal{M} \vdash \square(\Delta \leq \delta) \to \varphi \Leftrightarrow \forall \sigma. (\exists q. (\sigma)^q \models \varphi) \Rightarrow (\sigma)^{r(\sigma, \delta, \mathcal{M})} \models \varphi,$$

i.e. if σ is a sequence allowed by the specification φ, $\mathcal{M}$ synchronizes the events σ at least as likely as it is required. The algorithmically challenging part in establishing correctness is to evaluate $r(\sigma, \delta, \mathcal{M})$. However, we only need to compute this for the CSAs that we are synthesizing.

5. SYNTHESIS

The protocol synthesis method $\mathfrak{S}$ translates a specification into a set of CSAs that is guaranteed to satisfy the specification. The inputs to the synthesis are a protocol specification φ, a set of cars $\mathfrak{C}$ and the specification on the transmission medium dynamics $\square(\Delta \leq \delta)$. $\mathfrak{S}(\varphi, \mathfrak{C}, \delta)$ produces a CSA for each car $x \in \mathfrak{C}$ that interacts with the higher level ASCs as outlined in Sec. 3.

5.1 Realizability and Well-Posedness

Synthesis is preceded by a realizability check, i.e. checking whether a specification *can* be implemented. That is, checking realizability amounts to deciding whether there exists a set of CSAs that satisfies the specification $\square(\Delta \leq \delta) \to \varphi$. If a protocol specification φ is realizable for a set of cars $\mathfrak{C}$ under a drop probability δ, this is written as $\mathfrak{R}(\varphi, \mathfrak{C}, \delta)$.

Checking realizability consists of two parts: Firstly, the specification itself must be well-posed, i.e. φ must admit a "reasonable" implementation in the form of CSAs. Secondly, it must be possible to find retransmission bounds so that the QoS requirements are satisfied under the given drop probability δ.

Well-posedness is a purely syntactic requirement on the specification. We introduce this concept because it is easy to check and simplifies the presentation of the synthesis algorithm. A protocol specification φ is *well posed* if on every p-sequence satisfying φ, two ASCs take turns in triggering the events, and there are at least two events on each path through the tree induced by the specification.

These rather strict requirements on the specifications for well-posedness can be relaxed by generalizing the synthesis method presented in the next section appropriately. For example, a straightforward relaxation would be to allow protocol specifications in which for any disjunction $\varphi_1 \vee \varphi_2$, the system-triggered events corresponding to the immediately following global events are all triggered by the same ASC.

We do not develop a separate test for realizability but rather show how our method fails for well-posed but nonrealizable specifications.

5.2 Synthesis Algorithm

The synthesis method is implemented in two parts: First, the retransmission bounds are calculated. Then the CSAs are constructed using the retransmission bounds. The retransmission bounds are calculated with the structure of the resulting CSAs in mind, so we present the CSA construction first.

For any specification $\Box(\Delta \leq \delta) \to \varphi$ and any set of cars $\mathfrak{C}$, if the specification is realizable, the resulting set $\mathcal{M}$ of CSAs from the synthesis, $\mathfrak{S}(\varphi, \mathfrak{C}, \delta)$ must satisfy $\Box(\Delta \leq \delta) \to \varphi$. Formally,

$$\forall \varphi. \forall \mathfrak{C}. \forall \delta \in [0,1]. \mathfrak{R}(\varphi, \mathfrak{C}, \delta) \Rightarrow (\mathfrak{S}(\varphi, \mathfrak{C}, \delta) \vdash \Box(\Delta \leq \delta) \to \varphi).$$

The synthesis method $\mathfrak{S}$ is implemented in two parts: First, the retransmission bounds are calculated. Then the CSAs are constructed using the retransmission bounds. The retransmission bounds are calculated with the structure of the resulting CSAs in mind, so we present the CSA construction first.
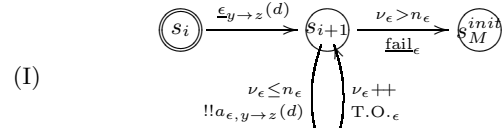
5.2.1 CSA Construction

Table 3 shows the algorithm $\textsc{Synthesize}(\varphi, x, i, E, n_\varphi)$. This algorithm constructs the CSA M_x for car x from the specification φ. The parameter i is used to uniquely index states in the CSA, and E is a set of global events that is used to construct appropriate criteria for retransmission (explained below). n_φ is the list of retransmission bounds calculated in the first step (cf. Sec. 5.2.2).

Each global event $e = \epsilon_{x \to y}(d)$ that occurs in the protocol specification φ is assigned an environment-triggered event $\underline{\epsilon}_{x \to y}(d)$, a system-triggered event $\underline{\epsilon}_{y \leftarrow x}(d)$, a message $m_\epsilon \in \mathrm{MSG}$, a variable (as retransmission counter) $\nu_\epsilon \in \mathcal{V}$, a retransmission bound n_ϵ from n_φ, and system-triggered events $\mathrm{T.O.}_\epsilon$ and $\underline{\mathrm{fail}}_\epsilon$.
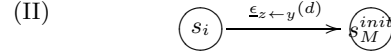
The algorithm is invoked by $\textsc{Synthesize}(\varphi, x, 0, \emptyset, n_\varphi)$, for each car $x \in \mathfrak{C}$:[2] It synthesises a CSA for the well-posed protocol specification φ for car x, where states are indexed starting from 0, no previous events are stored ($E = \emptyset$) and the retransmission bounds n_φ are used.

$\textsc{Synthesize}$ recursively decomposes φ into its subparts. If $\varphi = \varphi_1 \vee \varphi_2$, then two CSAs M_1 and M_2 are constructed from φ_1 and φ_2 first and joined together by forming the union of their state spaces, final states and transitions and substituting the initial state $s_{M_2}^{init}$ by the initial state $s_{M_1}^{init}$. For this purpose we define $M[s_1/s_2]$ to be the CSA M with all occurrences of s_2 substituted by s_1.

If $\varphi = \epsilon_{y \to z}(d) \to \bigcirc \varphi'$, the set E of global events that has last been received on the path through the CSA is updated first. Then again the CSA M for φ' is constructed. Depending on which car x the CSA is constructed for, different transitions are now introduced. If $x = y$ then the ASC on car x is responsible for triggering the event $\underline{\epsilon}_{y \to z}(d)$, and a retransmission loop is introduced:

---

[2]Note that x does not need to occur in the protocol specification φ.

(I)

If $x = z$, then car x synchronizes ϵ by the system-triggered event $\underline{\epsilon}_{z \leftarrow y}(d)$:

(II)

In any other case, simply the CSA for φ' is returned as then the car x is not directly involved in the transmission.

Finally, if $\varphi = \epsilon_{y \to z}^p(d)$ then no recursive call to $\textsc{Synthesize}$ is necessary, but a CSA is directly constructed. If $x = y$ then a retransmission loop is constructed:

(III)
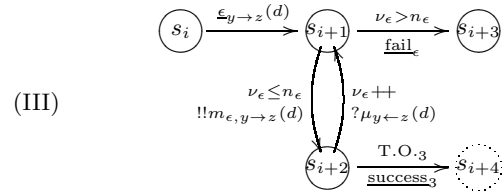
In this case, a retransmission is not triggered by a timeout, because $\epsilon_{y \to z}(d)$ is the last global event in a sequence of required synchronizations and no feedback from the car z can be expected. Therefore, a retransmission is initiated by receiving the last message μ from car z again, because this indicates that z has not received the message m_ϵ correctly. The message μ is taken from E, the set of global events that has last been received on the path through the CSA. Only if no such message is received is a timeout transition made, which indicates success by an upcall to the ASC. The global semantics of CSAs were carefully constructed so that this timeout is only taken if no message μ is received.

If $x = z$, then car x synchronizes ϵ by the system-triggered event $\underline{\epsilon}_{z \leftarrow y}(d)$:

(IV)

In any other case a trivial CSA with one state is returned.

Example 1 (Continued) The resulting CSAs from synthesizing the specification in (1) are shown in Fig. 5. The CSAs were generated by calling $\textsc{Synthesize}(\varphi, x, 0, \emptyset, n_\varphi)$ for $x \in \{A, B\}$. The retransmission bounds n_φ are calculated as explained in the next section according to the QoS requirements and to the bound on the drop probability δ. $\quad\Box$

5.2.2 Retransmission Bounds

Each global event $\epsilon_{x \to y}(d)$ gets assigned a unique message $m_\epsilon \in \mathrm{MSG}$ and a unique retransmission bound $n_\epsilon \in \mathbb{N}_0$. The retransmission bounds are evaluated according to the QoS requirements defined in the protocol specification φ.

Recall that the protocol specification φ induces a tree, cf. Fig. 4. Each edge of this tree is translated by the synthesis into a retransmission loop in the CSA of exactly one car, with a retransmission bound associated with that loop. The retransmission bounds have to be selected so that correctness as defined in Sec. 4.4 is guaranteed.

We use the semantics to find the conditions on the re-

SYNTHESIZE$(\varphi, x, i, E, n_\varphi)$
    If $\varphi = \varphi_1 \vee \varphi_2$ **Then**
        $(M_1, i_1) = $ SYNTHESIZE$(\varphi_1, x, i, E, n_\varphi)$
        $(M_2, i_2) = $ SYNTHESIZE$(\varphi_2, x, i_1, E, n_\varphi)$
        Return $(\langle S_{M_1} \cup S_{M_2}, s_{M_1}^{init}, S_{M_1}^f \cup S_{M_2}^f,$
                        $T_{M_1} \cup T_{M_2}\rangle[s_{M_1}^{init}/s_{M_2}^{init}], i_2)$
    Else If $\varphi = \epsilon_{y \to z}(d) \to \bigcirc \varphi'$ **Then**
        If $\exists \mu, d.\mu_{y \to z}(d) \in E$ **Then**
            Replace $\mu_{y \to z}(d)$ **By** $m_{\epsilon, y \to z}(d)$ **In** E
        Else
            Insert $m_{\epsilon, y \to z}(d)$ **Into** E
        If $x = y$ **Then**
            $(M, i') = $ SYNTHESIZE$(\varphi', x, i + 3, E, n_\varphi)$

(I) $\begin{cases} T_M(s_i, \underline{\epsilon}_{y \to z}(d)) := s_{i+1} \\ T_M(s_{i+1}, (!!m_{\epsilon, y \to z}(d), \nu_\epsilon \leq n_\epsilon)) := s_M^{init} \\ T_M(s_{i+1}, (\underline{fail}_\epsilon, \nu_\epsilon > n_\epsilon)) := s_{i+2} \\ T_M(s_M^{init}, (\text{T.O.}_\epsilon, \nu_\epsilon{+}{+})) := s_{i+1} \end{cases}$

            Return $(\langle S_M \cup \{s_i, s_{i+1}, s_{i+2}\}, s_i, S_M^f, T_M\rangle, i')$
        Else If $x = z$ **Then**
            $(M, i') = $ SYNTHESIZE$(\varphi', x, i + 1, E, n_\varphi)$

(II)$\{$ $T_M(s_i, (?m_{\epsilon, z \leftarrow y}(d), \underline{\epsilon}_{z \leftarrow y}(d))) := s_M^{init}$

            Return $(\langle S_M \cup \{s_i\}, s_i, S_M^f, T_M\rangle, i')$
        Else
            Return SYNTHESIZE$(\varphi', x, i, E, n_\varphi)$
    Else If $\varphi = \varphi = \epsilon_{y \to z}^p(d)$ **Then**
        If $x = y$ **Then**
            $\mu_{y \leftarrow z}(d) \in E$

(III)$\begin{cases} T(s_i, \underline{\epsilon}_{y \to z}(d)) := s_{i+1} \\ T(s_{i+1}, (!!m_{\epsilon, y \to z}(d), \nu_\epsilon \leq n_\epsilon)) := s_{i+2} \\ T(s_{i+1}, (\underline{fail}_\epsilon, \nu_\epsilon > n_\epsilon)) := s_{i+3} \\ T(s_{i+2}, (?\mu_{y \leftarrow z}(d), \nu_\epsilon{+}{+})) := s_{i+1} \\ T(s_{i+2}, (\text{T.O.}_\epsilon, \underline{success}_\epsilon)) := s_{i+4} \end{cases}$

            Return $(\langle \{s_i, s_{i+1}, s_{i+2}, s_{i+3}, s_{i+4}\}, s_i,$
                    $\{s_{i+4}\}, T\rangle, i + 5)$
        Else If $x = z$ **Then**
(IV)$\{$ $T(s_i, (?m_{\epsilon, z \leftarrow y}(d), \underline{\epsilon}_{z \leftarrow y}(d))) := s_{i+1}$
            Return $(\langle \{s_i, s_{i+1}\}, s_i, \{s_{i+1}\}, T\rangle, i + 2)$
        Else
            Return $(\langle \{s_i\}, s_i, \{s_i\}, \emptyset\rangle, i + 1)$

Table 3: Pseudocode of synthesis algorithm. The CSA is constructed from the diagrams explained in the text and referred to by Roman numerals.

transmission bounds that are sufficient for correctness. We can exploit the tree-like structure of the synthesized CSAs: Apart from the last two retransmission loops in each sequence, the message associated with a retransmission loop is never used at a later point in the same sequence.

Each sequence of global events $\sigma = \epsilon_1 \epsilon_2 \dots \epsilon_l$ is associated with a sequence of retransmission bounds $n_\sigma = n_{\epsilon_1} n_{\epsilon_2} \dots n_{\epsilon_l}$. Depending on the values of the retransmission bounds, the sequence σ is generated correctly with a certain *synchronization probability* $P^\sigma(n_\sigma)$ that depends on all retransmission bounds associated with any event in σ. Note that this probability is the likelihood of σ being generated correctly *given* that the calls are made by the ASCs that generate σ.

5.2.3 Deduction of $P^\sigma(n_\sigma)$

The synchronization probability $P^\sigma(n_\sigma)$ is evaluated as follows: The case of only one retransmission bound ($|n_\sigma| = 1$) never occurs. The case of exactly two retransmission bounds ($|n_\sigma| = 2$) means that the last retransmission loop

uses the message transmitted in the retransmission loop one before last, cf. Fig. 5. The example in this figure can be used to deduce the general expression for $P^\sigma(n_1, n_2)$. This is because the synthesis will always generate the same pattern for the last two global events in a specification.

We use the sequence $\sigma = \text{snd}_{A \to B}(d)\text{ack}_{B \to A}$ to deduce $P^\sigma(n_1, n_2)$ by evaluating the probability of correct synchronization by applying the deduction rules in Table 1 and Table 2. The CSAs $\langle A, B \rangle = \mathcal{M}$ start in the initial state $\langle s_1^A, s_1^B \rangle$. We omit writing the values of the retransmission counters within the states in our presentation. We let $P^\sigma(n_1, n_2) = p_{1,1}^\sigma(n_1, n_2)$, where $p_{i,j}^\sigma(n_1, n_2)$ are an auxiliary functions describing the probability of reaching a globally final state from $\langle s_i^A, s_j^B \rangle$ when the calls in σ are made.

Initially only M_A can execute by applying the [env] rule locally. Hence, we start the deduction at $\langle (\bullet)^1, \langle s_1^A, s_1^B \rangle, A \rangle$. We find $p_{1,1}^\sigma(n_1, n_2)$ by applying the global rules to deduce all sequences for which $\mathcal{M}$ ends in a globally final state and the calls in σ are made. First, apply [pr-e] globally and [env] locally and deduce

$$\langle (\bullet)^1, \langle s_1^A, s_1^B \rangle, A \rangle \Longrightarrow_{\mathcal{M}(\delta)} \langle (\underline{\text{snd}}_{A \to B}(d))^1, \langle s_2^A, s_1^B \rangle, A \rangle$$

globally from

$$\langle \bullet, s_1^A \rangle \xrightarrow{\ e\ }_{M_A} \langle \underline{\text{snd}}_{A \to B}(d), s_2^A \rangle$$

locally. This deduction step yields $p_{1,1}^\sigma(n_1, n_2) = p_{2,1}^\sigma(n_1, n_2)$, as the probability is not changed from state $\langle s_1^A, s_1^B \rangle$ to $\langle s_2^A, s_1^B \rangle$. From now on we omit the left hand side of the relation $\Longrightarrow_{\mathcal{M}(\delta)}$, since it is equivalent to the right hand side of the previous deduction. We further omit writing the local deductions.

At $\langle s_2^A, s_1^B \rangle$, globally only the [pr-e] rule can be applied. Locally, either [sys-c] or [b-c] can be applied, depending on the value of the retransmission counter ν_1. Applying [sys-c] corresponds to taking the transition labelled by $\underline{\text{fail}}_1$. Since then no final state can ever be reached, we only apply the [b-c] rule locally. So we apply locally the [b-c] rule:

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\underline{\text{snd}}_{A \to B}(d) + !!a_{A \to B}(d))^1, \langle s_3^A, s_1^B \rangle, A \rangle.$$

This transition leads to $p_{2,1}^\sigma(n_1, n_2) = p_{3,1}^\sigma(n_1, n_2)$. At this point the transmission medium is invoked and globally both the [trans] and [drop] rules can be applied. If [trans] is applied, M_B receives the message and we apply the [r-sys] rule locally. If [drop] is applied, merely the probability and prioritization changes. Hence, we can deduce either

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d))^{(1-\delta)}, \langle s_3^A, s_2^B \rangle, B \rangle, \text{ or}$$
$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho)^\delta, \langle s_3^A, s_1^B \rangle, B \rangle,$$

where $\rho = \underline{\text{snd}}_{A \to B}(d)$. Since two transitions may be taken, we get $p_{3,1}^\sigma(n_1, n_2) = (1-\delta)p_{3,2}^\sigma(n_1, n_2) + \delta\overline{p}_{3,1}^\sigma(n_1, n_2)$. After the application of [drop], M_B is prioritized but cannot make a transition. In state $\langle s_3^A, s_1^B \rangle$, globally only [npr] can be applied, with M_A making a timeout transition using [to-upd] locally:

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\underline{\text{snd}}_{A \to B}(d) + \text{T.O.}_1)^\delta, \langle s_2^A, s_1^B \rangle, A \rangle.$$

Since the application of [to-upd] increases the retransmission counter ν_1 by one, we get $\overline{p}_{3,1}^\sigma(n_1, n_2) = p_{2,1}^\sigma(n_1 - 1, n_2)$ and the base case $\overline{p}_{3,1}^\sigma(0, n_2) = 0$. This indicates that in state $\langle s_2^A, s_1^B \rangle$, the deduction may be repeated with the bound n_1 decreased by one, corresponding to a retransmission. If

$n_1 = 0$, i.e. in the base case, no more retransmissions are possible.

In state $\langle s_3^A, s_2^B \rangle$ after the transmission, M_B makes a transition in response to a call from its ASC. By applying [env] instead of [env'], we model that a is made immediately. Applying [pr-e] globally and [env] locally yields:

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + \underline{\text{ack}}_{B \to A})^{(1-\delta)}, \langle s_3^A, s_3^B \rangle, B \rangle,$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d)$. Then only [pr-e] with [b-c] can be applied (because again, applying [sys-c] does not conform with wanting to reach a final state). Therefore we get

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + !!b_{B \to A})^{(1-\delta)}, \langle s_3^A, s_5^B \rangle, B \rangle, \qquad (4)$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A}$. This generates the equalities $p_{3,2}^\sigma(n_1, n_2) = p_{3,3}^\sigma(n_1, n_2)$ and $p_{3,3}^\sigma(n_1, n_2) = p_{3,5}^\sigma(n_1, n_2)$.

In $\langle s_3^A, s_5^B \rangle$ we can apply either [trans] globally with [r-sys] locally on M_A, modelling a successful transmission, or we apply [drop] globally, modelling a dropped message. Hence we can either deduce

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + ?b_{A \leftarrow B})^{(1-\delta)(1-\delta)}, \langle s_5^A, s_5^B \rangle, A \rangle, \text{ or}$$
$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho)^{\delta(1-\delta)}, \langle s_3^A, s_5^B \rangle, A \rangle$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A}$. We get $p_{3,5}^\sigma(n_1, n_2) = (1-\delta)p_{5,5}^\sigma(n_1, n_2) + \delta \overline{p}_{3,5}^\sigma(n_1, n_2)$. In state $\langle s_5^A, s_5^B \rangle$, the sequence has been synchronized successfully. Here only [npr] with [to-sys] on M_B can be applied to yield

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + \text{T.O.}_2 + \underline{\text{success}}_2)^{(1-\delta)(1-\delta)}, \langle s_5^A, s_5^B \rangle, A \rangle,$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A} + ?b_{A \leftarrow B}$. This deduction ends in a globally final state and hence $p_{5,5}^\sigma(n_1, n_2) = 1$, because the sequence σ is correctly synchronized. In state $\langle s_3^A, s_5^B \rangle$ after the message has been dropped, only [pr-t] with [to-upd] locally on M_A can be applied:

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + \text{T.O.}_1)^{\delta(1-\delta)}, \langle s_2^A, s_5^B \rangle, A \rangle$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A}$. This step yields $\overline{p}_{3,5}^\sigma(n_1, n_2) = p_{2,5}^\sigma(n_1 - 1, n_2)$ with base case $\overline{p}_{3,5}^\sigma(0, n_2) = 0$. Now M_A retransmits (if its retransmission count is not yet exceeded) and we deduce with [pr-e] and [b-c] locally on M_A:

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + !!a_{A \to B}(d))^{\delta(1-\delta)}, \langle s_3^A, s_5^B \rangle, A \rangle$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A} + \text{T.O.}_1$. This yields $p_{2,5}^\sigma(n_1, n_2) = \overline{\overline{p}}_{3,5}^\sigma(n_1, n_2)$. Now [trans] can be applied with [r-upd] locally on M_B. However, when applying [drop], no final state can be reached by any sequence of applications of deduction rules. Hence we only apply [trans] and [r-upd] and get

$$\Longrightarrow_{\mathcal{M}(\delta)} \langle (\rho + ?a_{A \to B}(d))^{\delta(1-\delta)(1-\delta)}, \langle s_3^A, s_5^B \rangle, A \rangle$$

where $\rho = \underline{\text{snd}}_{A \to B}(d) + ?a_{B \leftarrow A}(d) + \underline{\text{snd}}_{B \leftarrow A}(d) + \underline{\text{ack}}_{B \to A} + \text{T.O.}_1$. This yields $\overline{\overline{p}}_{3,5}^\sigma(n_1, n_2) = (1-\delta)p_{3,3}^\sigma(n_1, n_2 - 1)$ with base case $\overline{\overline{p}}_{3,5}^\sigma(n_1, 0) = 0$.

5.2.4 Optimization Problem

For notational convenience, we drop the σ superscript if the context is clear and we are not referring to a particular sequence. When the sequence of global events σ has exactly two elements ($|\sigma| = 2$), we get

$$P(n_1, n_2) \triangleq \varrho(1 - \delta^{n_1+1}) + \frac{\varrho^3}{1 - \delta\varrho} \sum_{i=1}^{n_1} \delta^i \left[1 - (\delta\varrho)^M\right],$$

where $\varrho = (1 - \delta)$ is the reception probability and $M = \min(n_1 + 1 - i, n_2)$. When σ has more than two elements ($|\sigma| > 2$), the synchronization probability can be similarly deduced:

$$P(n_1, n_2, \ldots, n_l) \triangleq \varrho \sum_{i=0}^{n_1} \delta^i \underline{P}(n_1 - i, n_2 \ldots, n_l),$$

$$\underline{P}(n_1, n_2, \ldots, n_l) \triangleq \begin{cases} \varrho \sum_{i=0}^{M} \delta^i \underline{P}(n_2 - i, \ldots, n_l) & \text{if } l > 2 \\ P(n_1, n_2) & \text{if } l = 2, \end{cases}$$

where $M = \min(n_1, n_2)$. Ideally, we want to find the smallest retransmission bounds that ensure correctness. Each p-sequence $(\sigma)^p$ that satisfies the specification φ induces a condition on the retransmission bounds associated with the elements of σ. For example, a sequence $(\epsilon_1 \epsilon_2 \ldots \epsilon_l)^p$ induces the condition $P^{\epsilon_1 \epsilon_2 \cdots \epsilon_l}(n_{\epsilon_1}, n_{\epsilon_2}, \ldots, n_{\epsilon_l}) \geq p$. This inequality ensures that the sequence σ is generated by the CSAs with high enough probability as required by the correctness criterion set out above.

We can find the retransmission bounds by solving an optimization problem:

$$\text{(OPT)} \quad \min_{n_{\epsilon_1}, n_{\epsilon_2}, \ldots, n_{\epsilon_l}} \sum_{j=1}^{l} n_{\epsilon_j}$$
$$\text{s.t. } P^\sigma(n_\sigma) \geq p \quad \text{for all } (\sigma)^p \in \mathcal{S}_\varphi,$$

where $\mathcal{S}_\varphi = \{(\sigma)^p | (\sigma)^p \models \varphi\}$ is the set of p-sequences σ that satisfy the protocol specification φ.

Example 1 (Continued) Checking realizability of a specification amounts to checking well-posedness of the specification and feasibility of the optimization problem. For our example specification (1), the optimization problem is

$$\min_{n_{\text{snd}}, n_{\text{ack}}, n_{\text{nack}}} n_{\text{snd}} + n_{\text{ack}} + n_{\text{nack}}$$
$$\text{s.t. } P(n_{\text{snd}}, n_{\text{ack}}) \geq p_1$$
$$P(n_{\text{snd}}, n_{\text{nack}}) \geq p_2.$$

In the case that $p_1 = 0.7$, $p_2 = 0.8$ and $\delta = 0.35$, we get $n_{\text{snd}} = 3$, $n_{\text{ack}} = 1$, and $n_{\text{nack}} = 2$. $\square$

5.3 Correctness of Synthesis

Take any protocol specification φ, drop probability bound δ, and any p-sequence σ for which $(\sigma)^p \models \varphi$ holds. Then, correctness of the synthesis method is established by showing that $(\sigma)^{r(\sigma, \delta, \mathcal{M})} \models \varphi$, where $\mathcal{M}$ is the result of synthesis.

The definition of the feasible region of the optimization problem (OPT) contains the inequality $P(n_\sigma) \geq p$ for each such sequence σ. By the semantics of protocol specifications $(q \geq p \wedge (\sigma)^p \models \varphi) \Rightarrow (\sigma)^q \models \varphi$ for any sequence σ. It is therefore sufficient to show that $(\sigma)^{P(n_\sigma)} \models \varphi$ and $r(\sigma, \delta, \mathcal{M}) \geq P(n_\sigma)$, because then $(\sigma)^{r(\sigma, \delta, \mathcal{M})} \models \varphi$, as required to establish correctness.

First, if the retransmission bounds n_σ are part of a feasible solution to (OPT), then we necessarily have $P(n_\sigma) \geq p$, and

so $(\sigma)^{P(n_\sigma)} \models \varphi$ follows from $(q \geq p \wedge (\sigma)^p \models \varphi) \Rightarrow (\sigma)^q \models \varphi$.

Second, we have $P(n_\sigma) = r(\sigma, \delta, \mathcal{M})$ by construction of P (note that the superscript σ has been dropped from P^σ): $r(\sigma, \delta, \mathcal{M})$ is the sum of all probabilities p for which $\llbracket \rho \rrbracket = \sigma \wedge (\rho)^p \models \mathcal{M}(\delta)$, i.e. the environment-triggered events and system-triggered events in ρ synchronize to the sequence of global events σ and the p-sequence $(\rho)^p$ is generated by $\mathcal{M}$ and drop probability δ. By definition, $(\rho)^p \models \mathcal{M}(\delta) \Leftrightarrow \exists s^f \in S_\mathcal{M}^f . \exists x, y \in \mathfrak{C}. \langle (\bullet)^1, s^{init}, x \rangle \Longrightarrow_{\mathcal{M}(\delta)}^* \langle (\rho)^p, s^f, y \rangle$. It is therefore sufficient to show that in the deduction of the expression for $P(n_\sigma)$ exactly those p-sequences $(\rho)^p$ are taken into account that end in a globally final state $s^f \in S_\mathcal{M}^f$ (the prioritization of x and y can safely be ignored) and for which $\llbracket \rho \rrbracket = \sigma$.

The deduction of $P(n_\sigma)$ in Sec. 5.2.3 is essentially done by constructing a product automaton of all CSAs in $\mathcal{M}$ using the global semantics, and adding the probabilities along all paths that end in a globally final state corresponding to the sequence σ having been executed.

Note that it would have been enough to show $P(n_\sigma) \leq r(\sigma, \delta, \mathcal{M})$. A synthesis method that generates CSAs with $P(n_\sigma) = 0$ would be perfectly correct, but not very useful: The larger $P(n_\sigma)$ gets, the greater the feasible region of (OPT) gets and the more specifications can be synthesized. So by having $P(n_\sigma) = r(\sigma, \delta, \mathcal{M})$, we have maximized the capabilities of the synthesis method.

5.4 Computational Considerations

The time required to generate a CSA from a protocol specification φ by the SYNTHESIZE algorithm is proportional to the number of global events and disjunctions ($\vee$) in φ (ignoring the set operations on E), which can easily be seen from Table 3, where the implementation of SYNTHESIZE is shown as a simple structural recursion on φ. When also the set operations on E are taken into account, the algorithm is quadratic in the number of global events in φ.

The main computational complexity arises from the optimization problem OPT, which is an integer program and in general is NP-hard. There are however a few points to be noted that may simplify finding a solution. First, both the objective function and the function $P^\sigma(n_\sigma)$ are monotonous in their arguments along any dimension. Hence, if OPT is feasible for some $n = (n_{\epsilon_1}, n_{\epsilon_2}, \ldots, n_{\epsilon_l})$, it is also feasible for any $n' \geq n$.

Second, since correctness depends on $P^\sigma(n_\sigma) \leq r(\sigma, \delta, \mathcal{M})$, it is sufficient to solve an optimization problem with a strictly smaller feasible set than that of (OPT). This is helpful if a function Q^σ can be found s.t. for all σ, $Q^\sigma(n_\sigma) \leq P^\sigma(n_\sigma)$ while still maintaining that there exist retransmission bounds n_σ s.t. $Q^\sigma(n_\sigma) \geq p$ for all $(\sigma)^p \in S_\varphi$. The solution to the resulting optimization problem might not be optimal, but the resulting CSAs are still correct.

Lastly, since any suboptimal solution to OPT still gives rise to correct CSAs, the retransmission bounds may be chosen to be arbitrarily high as long as they are feasible. Note however that there might not be a solution to OPT at all, in which case the specification was unrealizable in the first place.

5.5 Discussion

The implementations of a communication protocol specification provide the ASCs with sufficient information on
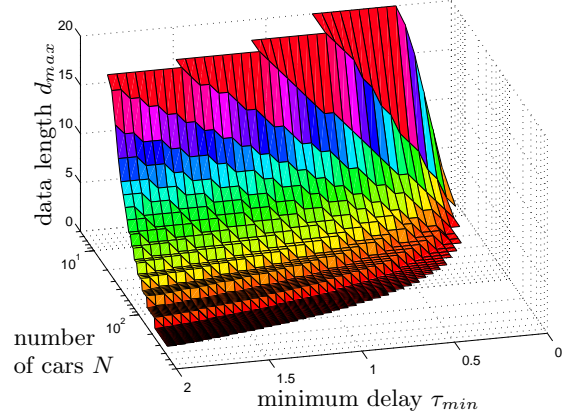

Figure 6: Feasible region of OPT for the protocol specification in (1) **with** $p_1 = p_2 = 0.9$**. The drop probability bound** δ **is calculated as a function of** N**,** d_{max} **and** τ_{min}**. All points on and under the surface are feasible.**

what messages are received so that accidents can effectively be prevented. In this section we develop the continuing example of the cars at an intersection further by explaining how our protocol can be embedded in an active safety application.

Example 1 (Continued) When transmitting data d from car A to car B, six cases can occur. We distinguish the cases by the final system-triggered events that generate upcalls to the ASCs on either car. The case we call "correct" is when B receives d, A knows about it and B assumes correctly that A knows. B then correctly receives a "success" upcall, which is consistent with A's last upcall. The ASCs can then correctly react in a consistent way, e.g. by one car gracefully decelerating.

In other cases the ASCs can still react in a safe way even if A and B have inconsistent information about each other: If B receives d correctly, A never receives an acknowledgement and B assumes A never did, then both ASCs receive "fail" upcalls and can react accordingly. If B receives d correctly and A receives the acknowledgement, but B assumes A did not receive it, then B receives a "fail" upcall and can react conservatively. If B does not receive d and A holds that it did not, then the ASC on A can react conservative on its "fail" upcall. If B receives d correctly, A misses the acknowledgement but B holds that A received it, then A incorrectly assumes the worst case but yet reacts conservatively.

The only problematic case is when B does not receive d but A holds that it did. Then the ASC on neither A nor B takes conservative action, potentially resulting in an accident. However, the synthesis method constructs the CSAs so that this case never occurs under the given assumptions. □

We now conclude the example by presenting numerical results that illustrate in which hypothetical scenarios protocols that we are considering are realizable.

Example 1 (Continued) As introduced above, the drop probability bound δ on the transmission medium may be calculated from other more readily available parameters. The realizability of a given protocol specification φ depends on the drop probability bound δ. For demonstrative purposes, we calculate δ from the number of cars N at the intersection that may use the transmission medium simultaneously,

the minimum time τ_{min} it may take for a message to be sent between two cars and the maximum amount of data d_{max} that may be carried in a message. Given an empirically obtained function $\delta(r)$ that maps a data-rate r to a drop-probability of the transmission medium, we calculate $\delta(r)$ with $r = (N-2)d_{max}/\tau_{min}$ (we take $N-2$ as we consider the environment to be all cars except the two that are communicating).

We illustrate the effectiveness of our synthesis method by asserting the sigmoid $\delta(r) = (1 + a \cdot \exp(-br))$ with $a = 4$ and $b = 0.002$. Using the protocol specification in (1), we illustrate how realizability changes with different values for the number of cars N, minimum time to deliver a message τ_{min} and maximum amount of data in a message d_{max}. Fig. 6 shows the feasible region of (OPT) for φ with $p_1 = p_2 = 0.9$, i.e. for which values of δ calculated as a function of N, τ_{min} and d_{max} the synthesis problem is realizable.

It is clearly visible from Fig. 6 that the more cars are sharing the transmission medium, the smaller the delay, and the larger the packets, the higher the worst-case data rate could be on the network, and the specification becomes harder to realize. If moreover the requirements p_1 and p_2 are made more stringent, the feasible region decreases even further. □

6. CONCLUSION

This work demonstrates a framework for reliable communication protocols for intervehicular communication in active safety applications. The framework, consisting of a precisely defined specification language and execution model (in the form of CSAs), allows for correct-by-construction synthesis of protocol implementations that satisfy the specifications even in the presence of several other cars sharing the transmission medium.

In our synthesis method we only take into account the drop probability of the transmission medium and assume that this is sufficient to synthesise reliable protocols. This also only enables to guarantee QoS requirements on the reception probability. Furthermore, in the current formulation, only two cars can participate in a dialogue, but some active safety applications might require to extend this. Also, note that if a communication is under way, the arrival of another message cannot directly be handled even if it is required to satisfy the QoS requirements.

Our approach permits several extensions: (i) Allowing the higher level to specify the QoS requirements and the destination address at runtime (i.e. for each transmission), (ii) Guaranteeing QoS requirements on the end-to-end delay of the communication and more general assumptions on the transmission medium dynamics in order to widen the range of applicability, and (iii) including the capability to relay messages over several cars to create a routed network. The latter would also require a rigorously developed synthesis method for protocols to discover the network topology, which we are currently working on.

Acknowledgements

7. REFERENCES

[1] H. Barringer and R. Kuiper. Hierarchical development of concurrent systems in a temporal logic framework. In S. Brookes, A. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 35–61. Springer, 1985.

[2] K. Bhargavan, D. Obradovic, and C. A. Gunter. Formal verification of standards for distance vector routing protocols. *Journal of the ACM*, 49(4):538–576, July 2002.

[3] S. Biswas, R. Tatchikou, and F. Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine*, 44(1):74–82, 2006.

[4] D. Caveney. Cooperative Vehicular Safety Applications. *IEEE Control Systems Magazine*, 30(4):38–53, 2010.

[5] P.-Y. Chu and M. Liu. Protocol synthesis in a state-transition model. In *Proc. of the International Computer Software and Applications Conference*, pages 505–512, 1988.

[6] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31(1):591–656, 2008.

[7] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. *Electronic Notes in Theoretical Computer Science*, 128(6):195–214, 2005.

[8] K. Farkas, J. Heidemann, L. Iftode, T. Kosch, M. Strassberger, K. Laberteaux, L. Caminiti, D. Caveney, and H. Hada. Vehicular communication. *IEEE Pervasive Computing*, 5:55–62, 2006.

[9] H. Hartenstein and K. Laberteaux. A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171, 2008.

[10] Y. Hatanaka, M. Nakamura, Y. Kakuda, and T. Kikuno. A synthesis method for fault-tolerant and flexible multipath routing protocols. In *Proc. of the IEEE International Conference on Engineering of Complex Computer Systems*, page 96, 1997.

[11] C. Huang, R. Sengupta, H. Krishnan, and Y. Fallah. Implementation and evaluation of scalable vehicle-to-vehicle safety communication control. *IEEE Communications Magazine*, 49(11):134–141, 2011.

[12] K. Ishida, Y. Kakuda, M. Nakamura, T. Kikuno, and K. Amano. A protocol synthesis method for fault-tolerant multipath routing. *Information and Software Technology*, 41(11–12):745–754, 1999.

[13] S. Khurana, N. Gupta, and N. Aneja. Reliable ad-hoc on-demand distance vector routing protocol. In *Proc. of the IEEE Conference on Networking*, page 98, 2006.

[14] F. Li and Y. Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22, 2007.

[15] C. Maihofer. A survey of geocast routing protocols. *IEEE Communications Surveys & Tutorials*, 6(2):32–42, 2004.

[16] J.-C. Park and R. Miller. Synthesizing protocol specifications from service specifications in timed extended finite state machines. In *Proc. of the International Conference on Distributed Computing Systems*, pages 253–260, 1997.

[17] K. Saleh and R. Probert. Automatic synthesis of protocol specifications from service specifications. *IEEE Transactions on Computers*, 40(4):615–621, 1991.

[18] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, 1997.

[19] A. Wang, P. Basu, B. Loo, and O. Sokolsky. Declarative network verification. In *Proc. of the International Symposium on Practical Aspects of Declarative Languages*, pages 61–75, 2009.

[20] O. Wibling, J. Parrow, and A. Pears. Automatized verification of ad hoc routing protocols. In *Proc. of the International Conference on Formal Techniques for Networked and Distributed Systems*, pages 343–358, 2004.

[21] X. Yang, L. Liu, N. Vaidya, and F. Zhao. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Proc. of the International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 114–123, 2004.

[22] F. Ye, M. Adams, and S. Roy. V2V wireless communication protocol for rear-end collision avoidance on highways. In *IEEE International Conference on Communications Workshops*, pages 375–379, 2008.

[23] P. Zafiropulo, C. West, H. Rudin, D. Cowan, and D. Brand. Towards analyzing and synthesizing protocols. *IEEE Transactions on Communications*, 28(4):651–661, 1980.

[24] H. Zimmermann. OSI reference model–the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425 – 432, 1980.