UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE

RENAN SEBEM

DISTRIBUTED CONTROL OF CONNECTED AND AUTOMATED VEHICLES: A DISCRETE EVENT SYSTEMS APPROACH

JOINVILLE
2022

RENAN SEBEM

DISTRIBUTED CONTROL OF CONNECTED AND AUTOMATED VEHICLES: A DISCRETE EVENT SYSTEMS APPROACH

Tese apresentada como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica pelo Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas - CCT, da Universidade do Estado de Santa Catarina - Udesc.
Orientador: André Bittencourt Leal
Coorientador: Douglas Wildgrube Bertol

JOINVILLE

2022

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

Sebem, Renan
Distributed Control of Connected and Automated Vehicles :
a Discrete Event Systems Approach / Renan Sebem. -- 2022.
115 p.

Orientador: André Bittencourt Leal
Coorientador: Douglas Wildgrube Bertol
Tese (doutorado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2022.

1. Multiple Connected and Automated Vehicles. 2. Distributed and Scalable Control. 3. Reconfigurable Path Planning. 4. Multiple Intersections Management. 5. Discrete Event Systems. I. Bittencourt Leal, André. II. Wildgrube Bertol, Douglas. III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. IV. Titulo.

RENAN SEBEM

DISTRIBUTED CONTROL OF CONNECTED AND AUTOMATED VEHICLES: A DISCRETE EVENT SYSTEMS APPROACH

Tese apresentada como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica pelo Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas - CCT, da Universidade do Estado de Santa Catarina - Udesc.
Orientador: André Bittencourt Leal
Coorientador: Douglas Wildgrube Bertol

BANCA EXAMINADORA

Prof. Dr. André Bittencourt Leal
Univ. do Estado de Santa Catarina

Membros:

Profª. Drª. Patrícia Nascimento Pena
Univ. Federal de Minas Gerais

Profª. Drª. Patricia Della Méa Plentz
Univ. Federal de Santa Catarina

Prof. Dr. Yuri Kaszubowski Lopes
Univ. do Estado de Santa Catarina

Prof. Dr. Benjamin Grando Moreira
Univ. Federal de Santa Catarina

Joinville, 19 de julho de 2022.

Dedico este trabalho à minha família que me apoia incondicionalmente nas minhas escolhas. À minha Esposa Laís, à minha filha Yasmin e aos meus pais Wilson e Raquel.

AGRADECIMENTOS

Em primeiro lugar sou grato pelo dom da vida e suas possibilidades.

Agradeço à minha esposa Laís que me ajudou em vários aspectos, principalmente pessoais, durante todo o período de realização desta tese, dedicando seu tempo para que eu pudesse dedicar tempo ao desenvolvimento da tese. Também tenho que agradecer a minha filha Yasmin de um ano e 10 meses, que muitas vezes me ajudou a escrever a tese digitando “asdfghjkl” no teclado. Também agradeço aos meus pais Wilson e Raquel pois sempre me ajudam em tudo que peço.

Agradeço ao meu orientador André por toda a sua dedicação e compromisso comigo, com outros orientados e com a pesquisa científica. Ele é uma pessoa excepcional que acredita no potencial das pessoas e que permite o desenvolvimento do trabalho com liberdade. E isto é peça fundamental para a pesquisa científica, que requer criatividade e inovação. Espero ter rendido (e ainda render) frutos à altura da orientação recebida.

Agradeço ao meu coorientador Douglas, Tchê, pela parceria. As conversas com mate sempre rendem, seja uma solução ou pelo menos uns bons risos. O Tchê também é uma pessoa excepcional que contribuiu muito com o meu desenvolvimento na pesquisa. Seu semblante sempre descontraído e despreocupado parece até fácil de se conseguir.

Agradeço aos meus colegas de laboratório Maurício e Matheus, dos quais fui coorientador de seus trabalhos de conclusão de curso. Trabalhos estes que foram propostas provenientes desta tese. Agradeço pela confiança e principalmente por todo esforço dedicado à nossa colaboração.

Agradeço ao grupo estudantil GERM, pelo empréstimo dos robôs utilizados nos experimentos desta tese.

Agradeço à Georgia Tech College of Engineering pelo desenvolvimento, manutenção e disponibilização do Robotarium.

Agradeço a CAPES e ao FUMDES pelas bolsas de doutorado concedidas à mim (em períodos distintos). Agradeço à Fundação de Amparo à Pesquisa e Inovação do Estado de SC - FAPESC 2021TR930, pelo suporte financeiro investido no grupo de pesquisa.

Agradeço aos colegas de laboratótio Lucas e Áureo, pelas conversas filosóficas e elucidantes, pela companhia e amizade.

Agradeço à colega (e minha professora na época de graduação) Ana, e ao professor Marcelo pela colaboração na escrita de um artigo, que surgiu como um trabalho na disciplina do Marcelo e rendeu um artigo publicado em periódico internacional qualificado, depois de muito esforço, é claro.

Agradeço ao colega Daniel, pela colaboração na escrita de um artigo baseado em sua dissertação de mestrado, que rendeu um artigo publicado em periódico internacional.

Agradeço aos colegas de laboratório Hebert, Rangel; e aos profesores Celso e Fabrício, pela companhia e pelas conversas.

Agradeço aos professores e profissionais do PPGEEL e do SECEPG.

“Existem apenas dois dias no ano em que nada pode ser feito. Um é chamado Ontem e o outro é chamado Amanhã. Hoje é o dia certo para Amar, Acreditar, Fazer e principalmente Viver."

Dalai Lama

RESUMO

Esta tese propõe uma solução para o problema de controle de CAVs em grandes centros urbanos. A arquitetura de controle para CAVs proposta neste trabalho combina três características: Distribuída, Escalável e Reconfigurável. O termo *Distribuída* é usado no sentido de que o controle está totalmente embarcado nos CAVs, sem um gestor/coordenador central. O termo *Escalável* é usado no sentido de que o número de CAVs pode mudar no tempo de execução. E o termo *Reconfigurável* é usado no sentido de que os CAVs podem alterar seus destinos e caminhos a qualquer momento. Com esta arquitetura, garante-se o comportamento não bloqueante e livre de colisões do sistema, e também a privacidade do caminho para cada CAV. A estrutura de controle local é composta por uma planta local, um controlador de caminho e um controlador de coordenação, todos modelados por autômatos. A disputa pela ocupação da via entre os CAVs é tratada para evitar colisões, com base em uma comunicação sem fio onde cada CAV impõe desabilitações de eventos (ações) aos demais CAVs. Propõe-se um tratamento online para prevenção de bloqueios, sem a necessidade de computar um controlador monolítico em nenhuma fase do projeto. Os controladores de caminho e coordenação são propostos para serem embarcados e sintetizados no CAV em tempo de execução, o que permite a reconfiguração. A escalabilidade é alcançada explorando a similaridade entre as plantas dos CAVs, por meio do uso de um mapa de reetiquetagem. A arquitetura é validada através de um ambiente de simulação no Robotarium e testes práticos experimentais com robôs Lego Mindstorms. Dois exemplos são dados, assim como duas simulações e dois experimentos são executados. Uma das simulações foi executada por 54h, variando o número de CAVs de 1 até 36, e na qual foi encontrado que a arquitetura é mais eficiente em 22,2% da taxa de ocupação dos CAVs no mapa. Os resultados mostram que todos os requisitos foram atendidos.

Palavras-chave: Múltiplos Veículos Conectados e Automatizados, Controle Distribuído e Escalável, Planejamento de Caminhos Reconfiguráveis, Gerenciamento de Múltiplas Interseções, Sistemas a Eventos Discretos.

ABSTRACT

This thesis proposes a solution to the problem of CAVs control in large urban environments. The control architecture for CAVs proposed in this work combines three features: Distributed, Scalable, and Reconfigurable. The term *Distributed* is used in the sense that the control is totally embedded in the CAVs, without a central manager/coordinator. The term *Scalable* is used in the sense that the number of CAVs can change in the runtime. And the term *Reconfigurable* is used in the sense that the CAVs may change their destinations and paths at any time. This architecture ensures the system's nonblocking and collision-free behavior, and the path's privacy for each CAV. The local control structure comprises a local plant, a path controller, and a coordination controller, all modeled by automata. The dispute for road occupation between CAVs is handled to avoid collisions, based on a wireless communication where each CAV imposes disablings of events (actions) to other CAVs. We propose an online handling for deadlock avoidance, with no need to compute a monolithic controller in any design phase. The path and coordination controllers are proposed to be embedded and synthesized in the CAV at runtime, which allows the reconfiguration. The scalability is achieved by exploring the similarity between the CAVs' plants, through the use of a relabeling map. The architecture is validated through a simulation environment in Robotarium and a practical experimental testbed with Lego Mindstorms robots. Two examples are given, as well as two simulations and two experiments are executed. One of the simulations was run for 54h, varying from 1 to 36 CAVs, where it was found that the architecture is most efficient at 22.2% of the CAVs occupation rate on the map. The results achieved all the proposed objectives for the architecture, showing that it is not only a new and efficient way to solve the problem of CAVs control, but also provides the only distributed, scalable and reconfigurable solution for CAVs control.

Keywords: Multiple Connected and Automated Vehicles. Distributed and Scalable Control. Reconfigurable Path Planning. Multiple Intersections Management. Discrete Event Systems.

LIST OF FIGURES

LIST OF TABLES

LIST OF THEOREMS

ALGORITHMS LIST

LIST OF ABBREVIATIONS

CAV	Connected and Automated Vehicle
CMA	Colored Marking Automaton
DES	Discrete Event System
HDV	Human-Driven Vehicle
IoT	Internet of Things
MPC	Model Predictive Control
MTDES	Multitasking Discrete Event System
RMTDES	Reconfigurable Multitasking Discrete Event System
SCT	Supervisory Control Theory
V2I	Vehicle to Infrastructure communication
V2V	Vehicle to Vehicle communication
V2X	Vehicle to Vehicle and Vehicle to Infrastructure communication

SYMBOLS LIST

RG	Root generator
$\mathbf{C}_{\mathrm{pa}_i}$	Path controller automaton of cav_i
$\mathbf{C}_{\mathrm{co}_i}$	Coordination controller automaton of cav_i
$\mathbf{C}_{\mathrm{co}}$	Coordination controller automaton
$\mathbf{C}_{\mathrm{co}_1}$	Coordination controller automaton of cav_1
$\mathbf{C}_{\mathrm{co}_2}$	Coordination controller automaton of cav_2
$\mathbf{C}_{\mathrm{pa}_1}$	Path controller automaton of cav_1
$\mathbf{C}_{\mathrm{pa}_2}$	Path controller automaton of cav_2
$\mathbf{G}$	Plant model automaton
$\mathbf{S}$	Supervisor automaton
$\mathbf{K}$	Control specification automaton
$\mathbf{E}$	Control specification automaton
$\mathbf{G}_i$	Local plant model of cav_i
$\mathbf{K}_i$	Specification automaton for the path controller of cav_i
$\mathbf{K}_2$	Specification automaton for the path controller of cav_2
$\mathbf{S}_{\mathrm{pa}_i}$	Path supervisor automaton of cav_i
i**,** j**,** k	CAV's indexes
cav_i	Variable to identify the ith CAV
cav_1	Variable to identify the first CAV
cav_2	Variable to identify the second CAV
Σ	Finite set of events
Σ_i'	Enabled event set for cav_i
Σ_i	Event set of cav_i plant model
Σ_{co_1}	Event set for the coordination controller of cav_1
Σ_{co_i}	Event set for the coordination controller of cav_1
$D_{i\downarrow}$	Set of disabled events received by cav_i
$D_{i\uparrow}$	Set of disabled events sent by cav_i
$D_{1\uparrow}$	Set of disabled events sent by cav_1
$D_{2\downarrow}$	Set of disabled events received by cav_2

$D_{2\uparrow}$	Set of disabled events sent by cav_2
δ	Transition function
Γ	Active event function
$L(\mathbf{G})$	Language of automaton $\mathbf{G}$
$L_m(\mathbf{G})$	Marked language of automaton $\mathbf{G}$
R_i	Relabeling map to $_i$
Q	State set
q_0	Initial state
Q_m	Marked states set
C	Set of colors in a colored marking generator
χ	Marking function in a colored marking generator
ε	Empty string

CONTENTS

1 INTRODUCTION

This work aims at the problem of automated vehicles replacing Human-Driven Vehicles (HDVs). Recent advances in technology support the possibility of this replacement. In this sense, Fully Automated Autonomous Vehicles (AAVs) have improved safety, efficiency, and convenience in the driving experience in comparison to HDVs. These advantages may be improved even further with the exchange of traffic information between vehicles. This kind of vehicle is often referred to as Connected and Automated Vehicles (CAVs) (PARENT, 2013). One advantage of CAVs over AAVs is that the exchanged information can be used to coordinate CAV's movements to reduce or even extinguish the need for stopping at intersections. Also, another advantage is that the path planning of a CAV may consider the information of the traffic, and therefore improve efficiency. From another perspective, CAVs may be connected to signalized intersections, making street crossings safer for pedestrians.

Research in Connected and Automated Vehicles (CAVs) is a recent topic, and it is rapidly growing as the technology allows to foresee its application in a global scale, in a not-so-distant reality. The reason of this growth is based on three main motivations: safety, efficiency and convenience (GUANETTI; KIM; BORRELLI, 2018). Comparing CAVs with HDVs, it is easily noted that the CAVs receive information of the traffic beyond human capability. Thus, CAVs can provide better safety, considering that human errors are the greater cause of accidents in traffic. Also, CAVs control may reach better efficiency in many ways, such as fuel and time economy, due to optimal control of dynamics and optimal trajectory planning. Finally, it is clearly more convenient allowing humans to safely "fall asleep at the wheel". However, the CAV's control architecture must be designed to achieve these improvements, which is the subject of this work.

One aspect of the CAV's control is the structure, which may be classified as centralized (GUAN et al., 2020), decentralized (XIAO; CASSANDRAS, 2019; ZHANG; CASSANDRAS, 2019), or distributed (MIRHELI et al., 2019; LIU et al., 2018; ROSZKOWSKA; REVELIOTIS, 2013). In the centralized structure, all decisions are made by a central processing system, which may be unfeasible depending on the number of CAVs. In the decentralized structure, some of the decisions may be taken locally and some decisions are made by a coordinator; this approach improves the feasibility

for a large number of CAVs, but there is still a need for a coordinator (or decentralized coordinators), which has some disadvantages such as the cost and maintenance of it. In the distributed structure, all decisions are taken locally without the need for a coordinator, however, the complexity of the control system is increased. In this work, the term *Distributed* is used in the sense that the control is totally embedded in the CAVs, without a central manager/coordinator. In the literature, the term “in-vehicle control” is used.

Another problem in the control of CAVs is how to adapt the control system when the specification is changed at runtime. In other words, this problem deal with the recomputation of the control when the destination, or the path of the CAV is changed. Most methods in the literature consider that the control specification is fixed and cannot change at runtime (QUEIROZ, 2000, 2004; CAI; WONHAM, 2010; GUO; LI; BAN, 2019; GUANETTI; KIM; BORRELLI, 2018). That is, a change in the route requires a manual offline intervention, previous to the runtime. Furthermore, there is a research field which deals with dynamic path planning of CAVs, however, this kind of research is focused in the optimization algorithm, instead of the control architecture (JIANG et al., 2022).

In summary, a change in the system requires a respective change in the models of the CAVs and their controllers. In most cases, this change must be done offline, i.e., with the system turned off. In this work, we study two kinds of change: scalability and reconfiguration. We use the term *Scalable* in the sense that the number of CAVs can change in runtime and the control architecture is able to adapt to this change while in execution. The term *Reconfigurable* is used in the sense that the CAVs may change their paths (or tasks) at runtime, and at any time; and the control architecture is able to adapt to this change while in execution.

Considering specific problems in the traffic of CAVs, many solutions in different situations are currently under research, such as cruise control (MOSER et al., 2018), intersections management (MIRHELI et al., 2019; ZHANG; CASSANDRAS, 2019; GUAN et al., 2020; KAMAL et al., 2015; STEINMETZ et al., 2018; WANG; ZHAO; YIN, 2019; CHEN et al., 2020), mission assignment (BASILE; CHIACCHIO; MARINO, 2019), road merging (XIAO; CASSANDRAS, 2019; RIOS-TORRES; MALIKOPOULOS, 2017a; ITO et al., 2019; JING et al., 2019; DING et al., 2020; ZHENG et al., 2020), path planning (FRANSEN et al., 2020), platooning (CHEN et al., 2020b; GUO et al., 2020),

roundabouts (DEBADA; GILLET, 2018), warehouse automation (TATSUMOTO et al., 2018), etc. These approaches require complex solutions and most works present independent solutions for each problem, and rare are the works that treat more than one problem with the same solution.

1.1 MULTI-AGENT ASPECT OF CAVS SYSTEMS

The transit of CAVs in large-scale can be seen as a multi-agent system. A multi-agent system is characterized by the existence of countless independent and autonomous subsystems which can interact with each other, forming a complex system. A multi-agent system can be defined as a complex system, which can be segregated into different subsystems, called agents, which perform decisions and actions independently of the others. The agents interact in the same environment, thus forming the multi-agent system. Examples of multi-agent systems are manufacturing plants with industry 4.0 technology (KOVALENKO; TILBURY; BARTON, 2019); computer networks; Internet of Things (IoT); the transit of autonomous vehicles; and robot swarms (LOPES et al., 2016).

An important characteristic of multi-agent systems is the uniformity of the agents. Agents that perform the same functions are classified as similar or identical, composing a homogeneous system, i.e., a network formed only by computers is a multi-agent system with similar agents. This can provide a simplification in the modeling and controlling of systems formed by a large number of agents.

Another important feature is the number of agents. It is interesting that the number of agents of the system is variable at runtime. This provides flexibility to the system, for example, it does not need to stop the global system for the maintenance of an agent.

Other characteristic of multi-agent system regards the tasks of the agents. When agents execute a part of the work to achieve a global objective, the multi-agent system is called collaborative. In collaborative systems, generally, the order of task completion by each agent matters, and in some cases it has to be synchronized. When agents execute their own task, independent of the tasks of other agents, to achieve a local objective, the multi-agent system is called competitive. In competitive systems, the order of tasks completion does not matter, and each agent desires to finish its task first. It is interesting to note that competitive multi-agent systems is, in another perspective, a

multitasking system.

In this work, the term “CAVs system” refers to a multi-CAV system, seen from a perspective of a multi-agent system.

1.2 SUPERVISORY CONTROL FOR CAVS SYSTEMS

In terms of high-level control (i.e., coordination, path planning, road merging, etc), a multi-CAV system has a discrete state-space and it is adequate to be modeled as a Discrete Event System (DES). Also, DES modeling and control are well known in the literature and many structures of control have already been proposed, such as centralized, decentralized, distributed, and hierarchical; many control solutions have been implemented in the scope of DES, such as consensus-based and auction-based (BASILE; CHIACCHIO; MARINO, 2019); scalable control (LIU; CAI; LI, 2019a) and multitask control (QUEIROZ; CURY; WONHAM, 2005; QUEIROZ; CURY, 2005) has been also proposed for DESs in independent works. In this work, the structure of multi-agent systems is combined with modeling through DESs within the scope of Supervisory Control Theory (SCT) (RAMADGE; WONHAM, 1987), in which the control system is formally verified.

A problem in the distributed control of DES is that the exponential explosion of states still exists in the design phase (LIU; CAI; LI, 2018; CAI; WONHAM, 2010; QUEIROZ; CURY, 2000). In works by Liu, Cai and Li (2018, 2019b), the monolithic supervisor is computed first, and then the localization (CAI; WONHAM, 2010) is performed, that is, the distribution from a global to local supervisors. This localization method may become unfeasible due to the need of computing the monolithic supervisor. In the work of Queiroz and Cury (2000), the test of non-conflict is made after computing the local modular supervisors, and in this design phase the problem of exponential explosion of states still exists.

However, Liu, Cai and Li (2018, 2019b) exploit the similarities between agents to avoid the exponential explosion of states, thus calculating a monolithic supervisor called scalable. Another way to solve this problem is through the online calculation of the supervisor using the truncated models of the system, that is, periodically only a part of the supervisor is calculated (CHUNG; LAFORTUNE; LIN, 1993; HADJ-ALOUANE; LAFORTUNE; LIN, 1996). An example of the application of this technique, related to the subject of this work, can be found in the work by Tatsumoto et al. (2018). Although

the exponential explosion of states is addressed in these works, it is important to note that the control structure is centralized.

Given the distributed nature of multi-agent systems, it is pertinent that the control of these systems is also distributed. Furthermore, considering an urban traffic of CAVs, i.e., a large-scale system, it is unfeasible to implement a centralized control structure (QUEIROZ, 2000).

In a real application, the number of CAVs in CAVs systems can change over time. That is, a CAV that was idle (or powered off) can start its task at any time, or a CAV that has finished its task could stay on stand-by. So the control must be able to adapt each time the number of CAVs is changed, i.e., the control must be scalable at runtime. In terms of scalability, in the multi-agent control perspective, the similarity between agents can be explored to reduce the complexity and allow the scaling of the solution (SU; LIN, 2013; LIU; CAI; LI, 2019a). This kind of solution is feasible to be applied in CAVs systems, considering CAVs are similar, and depending on the modeling can be considered identical.

In terms of reconfiguration, the controller (supervisor) model must be computed at runtime. This way, the algorithms used to compute the controllers models must be feasible to be embedded in the CAV. We have found some works with reconfigurable control of DES, and they are presented and compared to the work in this thesis in Chaper 4.

1.3 MOTIVATION

The CAV's technology has the potential to improve many aspects in the traffic of vehicles, such as comfort, safety, time, and energy efficiency. To do so, the control system must be able to achieve these requirements. In Table 1, a correlation of the potential improvements in CAVs traffic vs. the technical requirements in the control system is presented. For example, the technical requirement of *collision-free behavior* assured by the control system, results in an improvement on the comfort and safety for the passenger.

It is very complex to achieve all these technical requirements in one control scheme. It is clear that through the use of conventional control schemes, these requirements cannot be achieved. Most works in the literature focus on one of these requirements (see Chapter 3). To the best of the author's knowledge, it was not found

Table 1 – Correlation between features that can be improved vs. technical requirements in CAVs traffic.

	Comfort	Safety	Time Efficiency	Energy Efficiency
Collision Free	Yes	Yes		
No stopping at intersections	Yes		Yes	Yes
No traffic jam	Yes		Yes	Yes
Dynamic path planning			Yes	Yes
Reliable communication		Yes		
Platooning				Yes

Source: designed by the author (2022).

in the literature a control architecture for CAVs which achieves all these requirements. This is evidence that research on control systems for CAVs is not established and further research is required.

There are other aspects of the control of CAVs which must be achieved by the control system to assure that it can be implemented in a very large-scale system. These aspects can be noted when observing how the traffic of HDV works. Firstly, there is not a central coordinator sending orders for each human driver, i.e., each human driver is autonomous on his decisions. However, every driver is subject to common rules or traffic laws. This rules apply to all drivers and they are made to ensure the collision-free behavior. This observation have motivated that the coordination of CAVs should be distributed.

In a second observation, vehicles may become active or inactive at any time, e.g., in the morning the vehicle is used to travel to the office, then it becomes inactive all day in the parking lot, then it is used to travel back home at evening. Thus, the number of active vehicles at the traffic are varying all the time and scalability is a way to solve this problem.

The third observation is that drivers/passengers may choose a route and change it at any time. For example, a driver may choose to travel to the restaurant at noon, but in the middle of the route, the driver observes that the fuel is almost empty, then he changes his route to the gas station. In terms of control, a reconfiguration at runtime is needed, i.e., an in-vehicle recomputation of the controller must be feasible.

In a fourth observation the only information traded by HDVs is brake and turn signals. This means that the only information shared by an HDV is “I am stopping” and “I am turning” left or right. Otherwise, it is assumed that the HDV is going forward.

Sharing the path or the destination violates the privacy of the user, and therefore we propose a communication based on the disabling of actions of other CAVs, which is sufficient to avoid collisions.

Regarding the modeling formalism, it was observed that most solutions in the literature are based on continuous state-space modeling. These solutions are most adequate to the vehicle's low level control, such as speed control. To the best of the author's knowledge, it was noted the lack of solutions of high level control, and in this case, the discrete event system approach, with a discrete state space, is most adequate to abstract away part of the dynamics, focusing on the high level of the path planning.

1.4 PROBLEM STATEMENT AND SCOPE

In this work the problem of controlling multiple CAVs in a shared environment is solved by proposing a control architecture, in which the requirements in the following statements are fulfilled:

(S1) The control structure distributed in each CAV assures that two or more CAVs cannot occupy the same space at the same time, i.e., the control structure assures a collision-free behavior;

(S2) The control structure distributed in each CAV assures the nonblocking of the whole system;

(S3) Each CAV computes its own path, optimized independently, considering an available shortest route;

(S4) CAVs do not share their paths with others;

(S5) The solution is feasible for large-scale systems, avoiding the exponential explosion of states[1];

(S6) The number of CAVs may change at runtime (scalable solution);

(S7) Each CAV may change its control specification at runtime, with no need to recompute other CAVs' control.

[1] In terms of DES models, the composition of many subsystems causes the exponential explosion of states w.r.t. the number of subsystems, in which case, a monolithic controller could be unfeasible.

The proposed solution considers the following assumptions:

(A1) The event sets for each CAV are pairwise disjoint;

(A2) Each CAV has only one destination at a time, i.e., $|Q_m| = 1$;

(A3) There is no need for the CAVs to arrive at their destinations at the same time, synchronously;

(A4) The local plant model of each CAV is based on the same map, i.e., they share the same resource;

(A5) The CAV's destination is an input to its control system. After arriving at a destination, the CAV: will have a new input to move to a new & different destination; or, will be parked outside the road;

(A6) The communication[2] topology is Vehicle to Vehicle (V2V), i.e., is done directly between CAVs, wirelessly, and has a sufficient and limited range;

(A7) The delay and packet loss in the communication are negligible;

(A8) CAVs are fully automated.

The exponential explosion of states, and the scalability at runtime are problems that, in other contexts, have already been solved in the literature (QUEIROZ, 2000, 2004; CAI; WONHAM, 2010). However, part of the purpose of this work is to perform the distributed synthesis of the control, so it is necessary to propose a new method for solving these problems.

1.5 OBJECTIVES

The **general objective** of this thesis is to propose a distributed control architecture for CAVs modeled as DES, with solutions for scalability and reconfiguration. Also, in this architecture, many solutions for CAVs can be modeled jointly, such as intersection management, road merging, dynamic path planning, and roundabouts. In summary, we present a distributed architecture with a scalable solution for both problems

[2]The communication technology is not the focus of this work. The most common solutions use the IEEE 802.11 protocol, however, the research on Ultra-Reliable Low Latency Communication (URLLC) over 5G, is in constant growth and may be adequate for CAV's communications.

of coordination of intersection management and dynamic path planning assuring the nonblocking and collision-free behavior, for multiple similar CAVs, with reconfigurable path at runtime. Furthermore, the privacy of the CAV's path is guaranteed through the proposed architecture.

The following **specific objectives** are highlighted:

- Develop algorithms for the embedded synthesis CAV's controllers;
- Develop an algorithm for automatic generation of the control specification;
- Perform the formal proof of the proposed architecture;
- Create didactic examples;
- Develop illustrative examples in a simulation environment;
- Implement the proposed architecture in an experimental testbed;
- Analyze the results achieved with simulations and experiments to validate the proposed control architecture.

1.6 CONTRIBUTIONS AND ORGANIZATION OF THE DOCUMENT

In Chapter 2, a summary of the fundamental concepts of the SCT which are related to this work are presented. We start by presenting the modeling formalism of languages & automata, the definition of blocking and the synchronous composition operation. Then we present three control structures in the framework of DES: monolithic, distributed and multitasking. These concepts are the basis for the control architecture proposed in this work.

In Chapters 4 and 3, two overviews are made to better understand the state of the art of the *supervisory control of multi-agent DES* and *CAVs control methods*, respectively. The overviews were carried out through a systematic search in the literature with the aim to find the maximum number of works on the respective topics. The topic in Chapter 4 is ampler, with many different nomenclatures which makes it difficult for the search, and thus, two different searches were performed. Then, in each chapter, we present the search methodology, and then realize the qualitative and quantitative analysis over the works found through the search, regarding the characteristics studied in this work.

In Chapter 5, the distributed, scalable and reconfigurable control architecture for CAVs is presented, which is the main contribution of this work. To design this architecture, we have taken advantage of the characteristics of CAVs systems to propose a novel solution to the blocking problem, without the exponential growth of states in any phase of the design. In the context of CAVs control, the architecture is developed to have a collision-free behavior, simultaneously providing the intersection management and the dynamic path planning.

In Chapter 6, detailed examples are provided to support the understanding of the proposed architecture. In the sequence, we present simulations which were designed to stress the proposed architecture. We provide statistics to evaluate the efficiency of the control. For the simulation we have adapted the Robotarium simulation environment in Python. And finally, we have developed an infrastructure to perform experiments with multiple CAVs. The infrastructure is composed by five Lego Mindstorms robots, configured as line followers; and a road environment with 5×6 intersections. We perform two experiments to demonstrate the feasibility of implementing the proposed architecture.

Finally, in Chapter 8, the conclusions and a summary of future works are presented.

2 DISCRETE EVENT SYSTEMS AND SUPERVISORY CONTROL

The objective of this chapter is to present the main concepts of supervisory control of Discrete Event System (DES), which are used as the basis for the control architecture proposed in this work (CASSANDRAS; LAFORTUNE, 2021; CURY, 2001). DES can represent many kinds of systems, including continuous and hybrid systems at a higher level of abstraction. Considering the interest in this work, the CAVs' paths and behavior are adequately represented as DES.

Discrete event systems are systems in which the state-space is discrete and the dynamic is driven by the asynchronous occurrence of discrete events. The concept of DES is presented in Definition 1 (CASSANDRAS; LAFORTUNE, 2021).

Definition 1 (Discrete Event System)**.** A discrete event system is an event-driven system, with discrete states, i.e., its evolution depends entirely on the asynchronous occurrence of discrete events over time. ◊

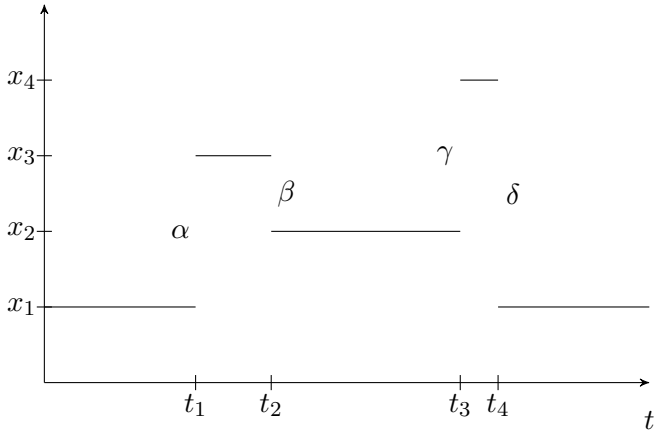
The states represent the system's situation, which may be a discrete physical quantity, an abstract condition, or a combination of situations of various subsystems. DES are dynamical systems, their states evolve over time and this evolution is given through the occurrence of stimuli, which may originate from the system itself, or from the external environment. These stimuli are called events (CURY, 2001). The transitions between states are associated with events. One event can be associated with many transitions, wich necessarily have different origin states, but not necessarily have different destination states.

Because DES have discrete state-space, events have an instantaneous duration in time, and are also asynchronous in time. Events represent the occurrence of a certain phenomenon in the system and can cause a change in its state. Figure 1 shows the state trajectory of any DES with states x_1, x_2, x_3 and x_4, and events α, β, γ and δ.

Note that state transitions, given by events, are instantaneous. States remain unchanged until an event occurs. The times of occurrence of events t_1, t_2, t_3 and t_4 are asynchronous in time. In theory, the events occcur instantaneously, i.e., they have zero duration.

The definition of states and events can be interpreted in many ways. And over

Figure 1 – State evolution of a DES in time.



Source: Ramadge and Wonham (1989).

these interpretations there are many modeling formalisms such as Petri nets, Markov chains, queuing theory, max-plus algebra, and languages & automata. In this work, languages & automata are adopted as the modeling formalism.

2.1 LANGUAGES & AUTOMATA: DES MODELING FORMALISMS

Languages & automata are two complementary modeling formalisms. They both can be used to represent the same behavior of a system, each one with a particular characteristic. Philosophically, languages are to differential equations as automata are to the Laplace transform.

2.1.1 Languages

Language is a simple way of representing the behavior of a DES through equations and text, based on the set theory. As with any language, the representation of a DES by language has an alphabet composed by letters or symbols, denoted by Σ, and words are formed by concatenating them. Thus, a language, denoted by L, is formed by the set of words of this alphabet (CASSANDRAS; LAFORTUNE, 2021). The words, or sequence of symbols, are usually called strings. A string is a finite-length sequence of events in Σ.

The notation $|\Sigma|$ represents the number of elements in Σ, and is valid for any kind of set. The notation $\Sigma_a = \Sigma_b \setminus \Sigma_c$, means that Σ_a is formed by all elements of Σ_b which are not contained in Σ_c.

Given a string s, its length, i.e., number of events including repetitions is denoted by $\|s\|$. Given the strings s and t, the concatenation of them is denoted by st. The set of all finite strings that can be formed with the elements of Σ is denoted by Σ^*, including the empty string ε (CURY, 2001). The set Σ^* is also called the Kleene-closure of Σ. Any subset of Σ^* is called a language over Σ.

Language theory is based on set theory, because language is a set of words, just as the alphabet is a set of symbols. Thus, some operations on sets can be performed on languages, such as union, intersection, and difference, among others. Important operations on languages are defined next.

The concatenation of two languages $L_a, L_b \subseteq \Sigma^*$ is defined as:

$$L_a L_b = \{s \in \Sigma^* : (s = s_a s_b) \text{ and } (s_a \in L_a) \text{ and } (s_b \in L_b)\}. \tag{2.1}$$

The prefix-closure of a language $L \subseteq \Sigma^*$, is denoted by $\overline{L}$, and defined by:

$$\overline{L} = \{s \in \Sigma^* : \exists\, t \in \Sigma^* \mid (st \in L)\}. \tag{2.2}$$

In words, the prefix-closure of a language $L \subseteq \Sigma^*$ is the set of all prefixes of the strings of the language L. Thus, $L \subseteq \overline{L}$. A language is said to be prefix-closed if $L = \overline{L}$, i.e., all prefixes of the language L are contained in L.

The Klenee-closure of a language $L \subseteq \Sigma^*$, denoted by L^*, is result of the concatenation of a finite number of strings in L, and is given by:

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup ... \tag{2.3}$$

In other words, Kleene-closure transforms a non-recursive language $L \subseteq \Sigma^*$ into a recursive language L^*.

A important concept for this works is the post-language of L after s, which denoted by L/s and defined as $L/s = \{t \in \Sigma^* : st \in L\}$.

An important classification is the linevess of languages. The language L is live if every string in L can be extended to another string in L. This represents a system which has recursive actions.

To represent the behavior of DES by languages, each possible action of the

system is associated with an event which can be represented by a letter (or symbol), that all together form the event set Σ.

The physical behavior of the system is a sequence of actions, represented by strings that form a language L, i.e., the physical behavior determines which sequences of events are possible to occur.

In some cases, it is difficult to visualize and understand a DES represent by languages. For example, a language with infinite number of strings. To overcome this difficulty, the representation of a DES through an automaton can be used.

2.1.2 Finite State Deterministic Automata

The representation of automata can be done in graphical form through a directed graph, or in textual form through a six-tuple. These two methods will be discussed next.

An automaton is a six-tuple $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$, where Q is the set of states; Σ is the finite set of events; δ: $Q \times \Sigma \rightarrow Q$ is the partial transition function; Γ is the active event function, in other words, it the set of events which has an output transition in a given state and can be obtained from δ; q_0 is the initial state, and; Q_m is the marked state set. The transition function δ is extended for strings such that δ: $Q \times \Sigma^* \rightarrow Q$, and:

$$\delta(q_0, s) = q_n; \tag{2.4}$$

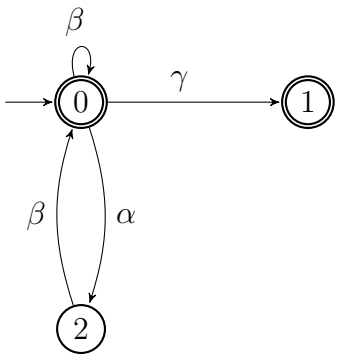
in which $s = \sigma_1\sigma_2 \ldots \sigma_n$, $\delta(q_0, \sigma_1) = q_1$, $\delta(q_1, \sigma_2) = q_2$, ..., $\delta(q_{n-1}, \sigma_n) = q_n$.

It is important to note that the transition function may not be defined for all elements of the event set Σ in every state of Q. This means that the transition function can be partial, which is usually the case. The initial state q_0 is unique and $q_0 \in Q$. The marked states $Q_m \subseteq Q$ can be multiple and represent a completed task in the system.

In a directed graph representation, states are represented by nodes (circles) and events by labeled arcs (arrows). The initial state is represented as the destination of an arrow with no source state, not characterizing an event. Marked states are represented with a double circle.

As an example, Figure 2 represents an automaton $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$, where $Q = \{0,\ 1,\ 2\}$; $\Sigma = \{\alpha,\ \beta,\ \gamma\}$; $\delta(0, \alpha) = 2,\ \delta(0, \gamma) = 1,\ \delta(0, \beta) = 0,\ \delta(2, \beta) = 0$; $\Gamma(0) = \{\alpha,\ \beta,\ \gamma\}$, $\Gamma(1) = \varnothing$, $\Gamma(2) = \{\beta\}$; $q_0 = 0$; $Q_m = \{0,\ 1\}$.

Figure 2 – Example of deterministic finite-state automaton.



Source: designed by the author (2022).

Finite state deterministic automata can be represented by regular languages. The automaton $\mathbf{G}$ is associated with two languages, $L(\mathbf{G})$ called the generated language of $\mathbf{G}$ and $L_m(\mathbf{G})$ called the marked language. $L(\mathbf{G})$ represents all physically possible behavior of an automaton $\mathbf{G}$, and $L_m(\mathbf{G})$ represents the behavior of $\mathbf{G}$ in which tasks are completed. Formally:

- $L(\mathbf{G}) = \{s \in \Sigma^* : \delta(q_0, s) \text{ is defined}\}$;
- $L_m(\mathbf{G}) = \{s \in \Sigma^* : \delta(q_0, s) \in Q_m\}$.

Basically, $L_m(\mathbf{G}) \subseteq L(\mathbf{G}) \subseteq \Sigma^*$. A language can be defined from a given state q, such as $L(\mathbf{G}, q)$ and $L_m(\mathbf{G}, q)$ which represent, respectively, the language and the marked language of $\mathbf{G}$ starting from state q, in which $L(\mathbf{G}, q) = \{s \in \Sigma^* : \delta(q, s) \in Q\}$ and $L_m(\mathbf{G}, q) = \{s \in \Sigma^* : \delta(q, s) \in Q_m\}$.

2.1.2.1 Accessible Part of an Automaton

An automaton can have inaccessible states starting from the initial state. Thus it is important to define the accessibility of $\mathbf{G}$. A state $q \in Q$ is accessible if $q = \delta(q_0, s)$ for some $s \in \Sigma^*$.

The $\mathbf{G}$ automaton is said to be accessible if all states of $\mathbf{G}$ are accessible. If an automaton is not accessible, the accessible component of an automaton can be obtained by eliminating the non-accessible states and the transition functions associated with them, this operation is known as $Ac(\mathbf{G})$.

The accessible part of $\mathbf{G}$ with respect to a state q is:

$$Ac(\mathbf{G}, q) = (Q_{ac}, \Sigma, \delta_{ac}, \Gamma_{ac}, q, Q_{m_{ac}}),$$

where $Q_{ac} = \{q' \in Q : (\exists s \in \Sigma^*)(\delta(q, s) = q' \text{ is defined})\}$, and $\delta_{ac} = \delta|_{Q_{ac} \times \Sigma \to Q_{ac}}$.

2.1.2.2 Coaccessible Part of an Automaton

A state q is said to be coaccessible if there is $s \in \Sigma^*$ such that $\delta(q, s = q_m)$ in which $q_m \in Q_m$. An automaton is coaccessible if all states are coaccessible.

In other words, the automaton $\mathbf{G}$ is said to be coaccessible, or nonblocking, if each string $s \in L(\mathbf{G})$ can be completed by $t \in \Sigma^*$ such that $st \in L_m(\mathbf{G})$, i.e., each string of $L(\mathbf{G})$ is a prefix of $L_m(\mathbf{G})$. This means that from any state of $\mathbf{G}$ there is at least one string that takes $\mathbf{G}$ to a marked state.

If an automaton is not coaccessible, the coaccessible component of an automaton can be obtained by eliminating the non-coaccessible states and the transition functions associated with them, this operation is known as $CoAc(\mathbf{G})$.

The coaccessible part of $\mathbf{G}$ with respect to q is:

$$CoAc(\mathbf{G}, q) = (Q_{coac}, \Sigma, \delta_{coac}, \Gamma_{coac}, q_{0_{oac}}, Q_m),$$

where $Q_{coac} = \{q \in Q : (\exists s \in \Sigma^*)(\delta(q, s) \in Q_m\}$; $q_{0_{oac}} = q,$ if $q \in Q_{coac}$, undefined otherwise; and $\delta_{coac} = \delta|_{Q_{coac} \times \Sigma \to Q_{coac}}$, which means that δ is restricted to the smaller domain of the accessible states Q_{coac}.

2.1.2.3 Trim Component of an Automaton

The trim part of $\mathbf{G}$ with respect to q is obtained by computing both accessible and coaccessible parts.

2.1.2.4 Blocking and Nonblocking

An automaton is said to be nonblocking if the prefix-closure of the marked language of this automaton is equal to the automaton's own language. In other words, every *string* belonging to the automaton language has at least one suffix that leads to a marked state. Below is a definition of blocking based on the conditions presented in this paragraph.

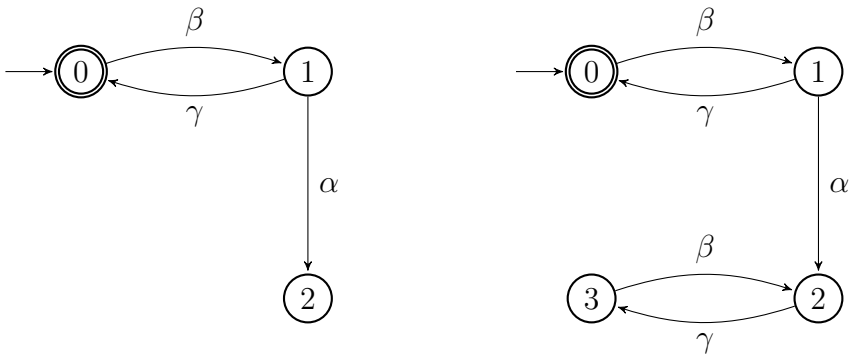
Definition 2 (Blocking)**.** (CASSANDRAS; LAFORTUNE, 2021). An automaton is considered to be blocking if there is at least one state, reached from the initial state, from which a marked state cannot be reached. ◊

Formally, having the language $L(\mathbf{G})$ as the closing prefix of the language $L_m(\mathbf{G})$, guarantees that $\mathbf{G}$ is nonblocking, i.e., it is necessary that $L(\mathbf{G}) = \overline{L_m(\mathbf{G})}$. If this equality is not satisfied, the automaton $\mathbf{G}$ is said to be blocking.

There are two different cases of blocking that occur depending on the conditions presented. These cases are widely known in the literature as *deadlock* and *livelock*. *Deadlock* occurs when an accessible state is not marked, and from that state, there is no possibility of any event occurring. In other words, the system is literally locked in the state and it means that the generated language of this system is not live. In the case of *livelock*, it is possible that from an accessible state there are infinite transitions between states, but a marked state will never be reached from these states.

In Figure 3, two cases of blocking are represented graphically, one automaton with a deadlock and one automaton with a livelock.

Figure 3 – Blocking automata with: (a) deadlock in state 2 and (b) livelock between states 2 and 3.



(a) (b)

Source: designed by the author (2022).

2.1.2.5 Synchronous Composition

The composition of automata allows the system to be modeled by several subsystems. Thus, instead of modeling the system as a whole, making the automaton complex, it is possible to model several subsystems by simpler automata and then ob-

tain the global model by the composition of the automata. Even so, if the system is relatively large, the automaton composition of the subsystems may lead to an automaton with a large number of states and transitions.

Consider $\mathbf{G}_1 = (Q_1,\ \Sigma_1,\ \delta_1, \Gamma_1, q_{0_1},\ Q_{m_1})$ and $\mathbf{G}_2 = (Q_2,\ \Sigma_2,\ \delta_2, \Gamma_2,\ q_{0_2},\ Q_{m_2})$, then, their synchronous composition, denoted by $\mathbf{G}_1 \parallel \mathbf{G}_2$, is given by:

$$\mathbf{G}_1 \parallel \mathbf{G}_2 = Ac(Q_1 \times Q_2,\ \Sigma_1 \cup \Sigma_2,\ \delta_{1\|2},\ \Gamma_{1\|2}, (q_{0_1},\ q_{0_2}),\ Q_{m_1} \times Q_{m_2}), \tag{2.5}$$

where:

$$\delta_{1\|2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if} \quad \sigma \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, \sigma), q_2), & \text{if} \quad \sigma \in \Gamma_1(q_1) \backslash \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)), & \text{if} \quad \sigma \in \Gamma_2(q_2) \backslash \Sigma_1 \\ \text{undefined}, & \text{otherwise} \end{cases} ; \tag{2.6}$$

$$\Gamma_{1\|2}(q_1, q_2) = (\Gamma_1(q_1) \cap \Gamma_2(q_2)) \cup (\Gamma_1(q_1) \backslash \Sigma_2) \cup (\Gamma_2(q_2) \backslash \Sigma_1). \tag{2.7}$$

To exemplify the synchronous composition take both automata from Figure 4, $\mathbf{G}_1 = (Q_1,\ \Sigma_1,\ \delta_1, \Gamma_1,\ q_{0_1},\ Q_{m_1})$ and $\mathbf{G}_2 = (Q_2,\ \Sigma_2,\ \delta_2, \Gamma_2,\ q_{0_2},\ Q_{m_2})$.

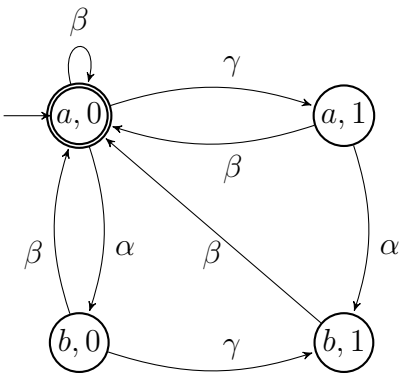
Figure 4 – Automata $\mathbf{G}_1$ and $\mathbf{G}_2$.

Source: designed by the author (2022).

The synchronous composition of $\mathbf{G}_1$ e $\mathbf{G}_2$, is given by $\mathbf{G}_1 \parallel \mathbf{G}_2 = Ac(Q_1 \times Q_2,\ \Sigma_1 \cup \Sigma_2,\ \delta_{1\|2}, \Gamma_{1\|2},\ (q_{0_1},\ q_{0_2}),\ Q_{m_1} \times Q_{m_2})$, and is represented in Figure 5.

One important note to make regards the number of states in the composition. In the worst case, considering pairwise disjoint event sets, the number of states in the composed automaton is the multiplication of the number of states of each individual automaton used in the composition. For example, considering a composition of n au-

Figure 5 – Composition of automata $\mathbf{G}_1$ and $\mathbf{G}_2$.



Source: designed by the author (2022).

tomata with pairwise disjoint event sets and 2 states each, the resulting number of states is 2^n. This phenomenon is known as the exponential explosion of states (CASSANDRAS; LAFORTUNE, 2021).

2.1.3 Representation of DES by Languages and Automata

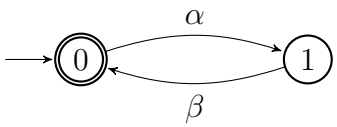
Next, a brief example is provided to compare the representation of a DES by a language and by an automaton. The representation by languages describes the complete behavior of the system with the generated language and the possible tasks that can be completed by the system. However, in order to represent the recursiveness of a system, the number of strings becomes innumerable, thus making the analysis of the system difficult. To deal with recursion, finite automata or regular languages can be used.

Considering how an automatic door works, the system behavior is such that the door can be opened and closed repeatedly, but it can never be opened twice without first being closed. Consider α and β representing the actions of opening and closing, respectively. The automaton $\mathbf{G}$ in Figure 2 represents the behavior of the door. This generates a language $L(\mathbf{G}) = \{\varepsilon,\ \alpha,\ \alpha\beta,\ \alpha\beta\alpha, \ldots\}$, where ε is the empty string. And the marked language is given by $L_m(\mathbf{G}) = \{\varepsilon, \alpha\beta, \alpha\beta\alpha\beta, \ldots\}$. It is intuitive that the automaton representation is easier to understand than the representation by languages.

2.2 SUPERVISORY CONTROL OF DES

The Supervisory Control Theory (SCT), which was first developed by Ramadge and Wonham (1987), is based on the theory of formal languages and automata and is

Figure 6 – Automaton representation of a door functioning.



Source: designed by the author (2022).

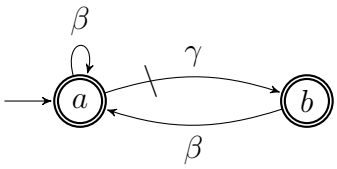
applied to the control of DES. Specific terms are used in SCT, such as *plant*, *specification*, *supervisor*, *controllable event*, and *uncontrollable event*.

A system generally has inputs and outputs, and depending on the case, they can be intuitively associated with reading and writing, or receiving and acting, respectively. In a DES control system (i.e. supervisor), generally, a control system input is associated with an uncontrollable event, such as a sensor signal, and a control system output is associated with a controllable event, such as turning a motor on.

It is important to clarify that in the Ramadge and Wonham (1987) framework, all events (controllable and uncontrollable) are spontaneously generated by the plant, i.e., the supervisor only enables/disables which controllable events can be generated by the plant.

Controllable events are graphically represented with a labeled arc sectioned by a slash (CURY, 2001). In Figure 7, an example is given, where the event γ is controllable, and the event β is uncontrollable.

Figure 7 – Controllable event representation in a finite state automaton.



Source: designed by the author (2022).

Thus, events are classified into controllable and uncontrollable. The event set can be divided such that $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c represents the set of controllable events, which can be enabled/disabled by the supervisor. And Σ_{uc} represents the set of uncontrollable events, which cannot be disabled by the supervisor. Then, the evolution of states over time is controlled by the supervisor through the enabling/disabling of the occurrence of controllable events (RAMADGE; WONHAM, 1987).

The control action performed by the supervisor over DES is defined as:

Definition 3 (Control Action)**.** The control action on DES is to enable or disable the occurrence of controllable events in determinate states. ◊

The plant is the physical system to be controlled. In open loop, the plant may present undesirable behaviors such as vehicles collisions or a vehicle moving in zigzag. Thus, in order to restrict the behavior of the plant to desirable behaviors, control specifications are created.

The goal in the SCT is to compute a supervisor, which will effectively control the plant. It is ideal that the supervisor is minimally restrictive, i.e., it allows the maximum occurrence of events assuring the control specifications, under a nonblocking behavior. The supervisor works in a closed loop, receiving the events generated by the plant, following the evolution of the plant through states, in which it enables or disables the occurrence of controllable events, according to the specifications.

The existence of a nonblocking supervisor is conditioned to the existence of a controllable sublanguage $L_c \subseteq L(\mathbf{G})$, such that $L_m(\mathbf{S}/\mathbf{G}) = L_c$. The notation $\mathbf{S}/\mathbf{G}$, means S controlling $\mathbf{G}$. The Definition 4 presents the conditions needed for a language to be controllable.

Definition 4 (Controlability)**.** Consider a DES $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$ with the set of uncontrollable events $\Sigma_{uc} \subseteq \Sigma$. Let $L(\mathbf{K}) \subseteq L(\mathbf{G})$, where $L(\mathbf{K}) \neq \varnothing$. Then there exists supervisor S such that $L(\mathbf{S}/\mathbf{G}) = \overline{L(\mathbf{K})}$ iff $\overline{L(\mathbf{K})}\Sigma_{uc} \cap L(\mathbf{G}) \subseteq \overline{L(\mathbf{K})}$. ◊

The SCT has formal procedures defined for synthesizing supervisors in different kinds of architectures. Some of these architectures are the monolithic, presented by Ramadge and Wonham (1987), the modular presented by Ramadge and Wonham (1989), the local modular, presented by Queiroz (2000) and the supervisor localization presented by Cai and Wonham (2010). The first, third and fourth methods will be explained in the next sections, as they are of great importance for this work.

2.2.1 Controller vs. Supervisor

It is necessary to define the distinction between controller and supervisor, as these concepts will be used later in the proposal of this work.

The control logic obtained through SCT is called supervisor due to some characteristics, presented in the previous section, which are summarized in the following definition.

Definition 5 (Supervisor)**.** Supervisor is the name given to the element of control that acts on a plant, enabling or disabling controllable events generated by the plant, in which the closed-loop behavior must be nonblocking and minimally restrictive. ◊

The controller definition is shown below.

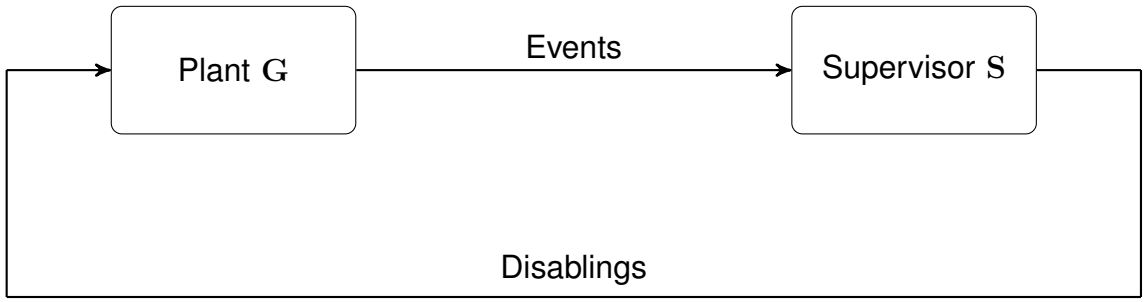
Definition 6 (Controller)**.** Controller is the general name given to a control element that performs control actions on a plant, in which the closed-loop behavior must be nonblocking, however is not minimally restrictive. ◊

The difference evidenced is relative to the minimally restrictive behavior, which is a characteristic assured by the supervisor and not by the controller.

2.2.2 Centralized or Monolithic Control

Presented by Ramadge and Wonham (1987), this approach consists of computing a single supervisor from one plant and one control specification. The plant may be a composition of several models of subsystems. And the specification may be a composition of several specifications for the subsystems. The most important characteristic is that the supervisor is obtained as a whole, and is most adequate to be implemented in a centralized architecture. The control system obtained by this approach is illustrated in Figure 8.

Figure 8 – Monolithic control architecture of DES in closed-loop.



Source: Ramadge and Wonham (1989).

However, when modeling a large DES, it may be formed by several subsystems. These subsystems can be modeled locally and then composed in a global plant model. The synchronous composition is performed on the local models $\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_n$, to obtain the global plant $\mathbf{G} = \mathbf{G}_1 \| \mathbf{G}_2 \| \ldots \| \mathbf{G}_n$, where n is the number of subsystems.

This process is similar for the control specifications, which can be modeled for each subsystem, or interaction of subsystems, and then composed into a global specification. Thus, the synchronous composition is performed on the specifications $\mathbf{E}_1, \mathbf{E}_2, ..., \mathbf{E}_m$, to form the specification $\mathbf{E} = \mathbf{E}_1 \| \mathbf{E}_1 \| \ldots \| \mathbf{E}_m$, where m is the number of specifications.

Then the automaton $\mathbf{K}$ is obtained through the synchronous composition of $\mathbf{E}$ and $\mathbf{G}$, i.e., $\mathbf{K} = \mathbf{E} \| \mathbf{G}$. The automaton $\mathbf{K}$ has the maximally permissive language which respects the control specification. If all events in Σ are controllable, then $\mathbf{K}$ is the supervisor. The problem is that the language of $\mathbf{K}$, may be trying to disable uncontrollable events, thus, it is necessary to compute the maximally sublanguage of $L_m(\mathbf{K})$ that is controllable with relation to $L(\mathbf{G})$, which is denoted by $supC(L(\mathbf{G}), L_m(\mathbf{K}))$.

In the synthesis of the supervisor, a state in which an uncontrollable event is disabled by the language of $\mathbf{K}$ is a *bad state*. The algorithm of the supervisor searches for these states and removes then, restricting the behavior of $\mathbf{K}$ through the disablings of controllable events. Then, the trim component is computed over the resulting automaton. This process is done recursively, until there are no bad states in the resulting automaton. Thus, the resulting automaton, generally called $\mathbf{S}$, is the supervisor which has the supremal controllable language over $\mathbf{G}$.

It is important to note that as the number of specifications and the number of plants increase, the number of states in $\mathbf{K}$ grows exponentially, due to the synchronous composition of these automata, considering pairwise disjoint event sets. This exponential explosion of states can become a problem when it comes to computing the models and implementing SCT, as the number of states directly affects the amount of memory used to execute the algorithms.

2.2.3 Distributed Control of DES

There are two distributed control approaches that stand out in the literature, one is called the local modular approach (QUEIROZ, 2000, 2004), and the other is called the supervisor localization (CAI; WONHAM, 2010).

Basically, in the local modular approach, local supervisors are first calculated and the conflict test is performed on the composition of the obtained supervisors. For this reason, this technique can be characterized as *bottom-up*.

When localizing supervisors, the monolithic supervisor is first calculated and

then calculations are performed to locate the supervisors. Characterizing the approach as *top-down*.

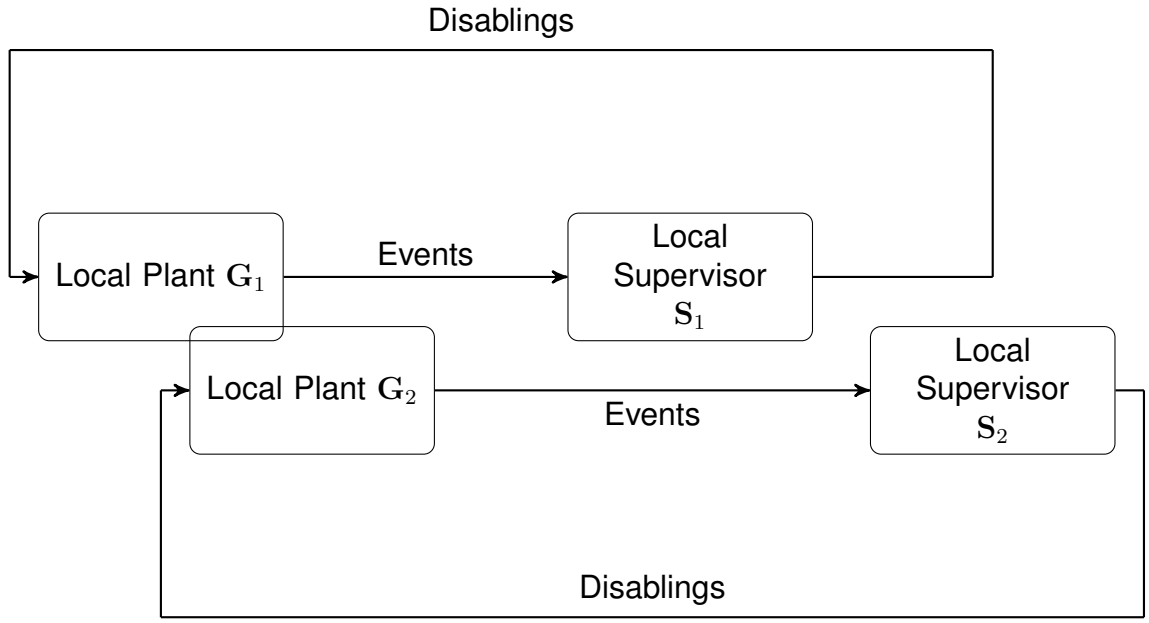
Next, the details of the local modular control of DES are given, which is used as the basis to the architecture proposed in this work.

2.2.3.1 Local Modular Control

The local modular approach is an extension of the classic modular approach (RAMADGE; WONHAM, 1989), as denoted by Queiroz (2000). This approach reduces the problem existent in the monolithic approach, in which the number of states increases exponentially in relation to the number of subsystems, considering pairwise disjoint event sets for each subsystem. Therefore, this approach reduces the computational complexity in the synthesis, in the implementation and in the maintenance of the supervisors.

Instead of a supervisor, several supervisors are obtained which control the behavior of the plant (QUEIROZ; CURY, 2000). Figure 9 illustrates the architecture of this control approach.

Figure 9 – Local modular control architecture of DES in closed-loop.



Source: Queiroz (2000).

The first step in the local modular approach is to obtain the most refined representation by product system, which consists of not composing the subsystems' plant

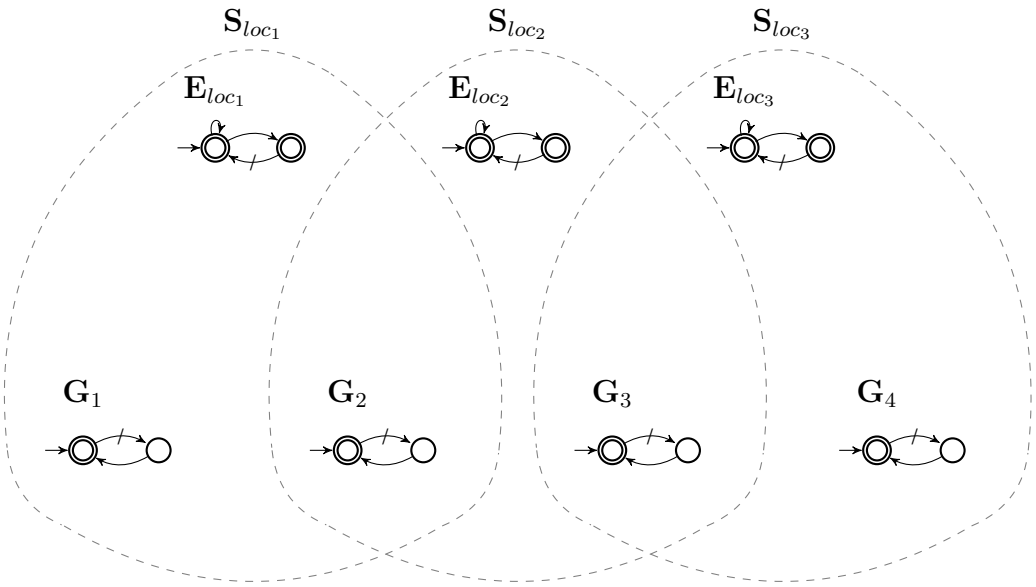
models unless they share events. In other words, if there are multiple subsystems that share events, then they must be composed and, therefore, must be considered as one subsystem. To adopt the local modular approach it is necessary that there are at least two subsystems which have pair-wise disjoint events set (do not share events). Each local subsystem model is called $\mathbf{G}_{loc_i}$

To design the control system, a control specification must be made for each local subsystem model.

The steps to obtain local supervisors are the same as in the monolithic approach, but the process is repeated for each local supervisor $\mathbf{S}_{loc_i}$, where i varies from 1 to the total number of local modules. The local supervisors $\mathbf{S}_{loc_i}$ are given by $\mathbf{S}_{loc_i} = supC(L(\mathbf{G}_{loc_i}), L(\mathbf{K}_{loc_i}))$. Where $\mathbf{K}_{loc_i} = \mathbf{G}_{loc_i} \| \mathbf{E}_{loc_i}$.

Figure 10 illustrates how the models are disposed in the local modules. The local plants are given by $\mathbf{G}_{loc_1} = \mathbf{G}_1 \| \mathbf{G}_2$, $\mathbf{G}_{loc_2} = \mathbf{G}_3 \| \mathbf{G}_2$, $\mathbf{G}_{loc_3} = \mathbf{G}_3 \| \mathbf{G}_4$.

Figure 10 – Disposal of the models in a local modular architecture.



Source: designed by the author (2022).

2.2.4 Multitasking Control

In the work of Queiroz, Cury and Wonham (2005), an approach to model DES with multiple tasks and to solve problems of multitasking supervisory control of DES

is introduced. Basically, a DES has multiple classes of tasks when it is composed by subsystems which have different objectives, that do not need to be accomplished at the same time, nor in any sequence. They introduced the Multitasking Discrete Event System (MTDES), which is a DES with multiple classes of tasks.

They introduced the notion of colored marking, where each color represents a different class of tasks. The colors are indicated in the set of colors C and the colored markings are represented as a function χ in the Colored Marking Automaton (CMA) tuples: $\mathbf{G} = (Q, \Sigma, C, \delta, \Gamma, q_0, \chi)$, where C is the set of colors and $\chi : Q \rightarrow 2^C$ is the coloring function.

When composing automata with one class of task (i.e. one color), a state represents the completion of a task only when the states of each subsystem also completes a task (see Figure 5). In the case of composing CMAs, the rule is:

- If both subsystems have the same class of tasks, then the composed state represents the completion of this task when the states of both subsystems complete the task;
- If one subsystem $\mathbf{G}_i$ has one class of task, which is not present in any other subsystem, then the composed state represents the completion of this task when the states of $\mathbf{G}_i$ completes the task.

Formally, the composition of markings in a CMA is given by:

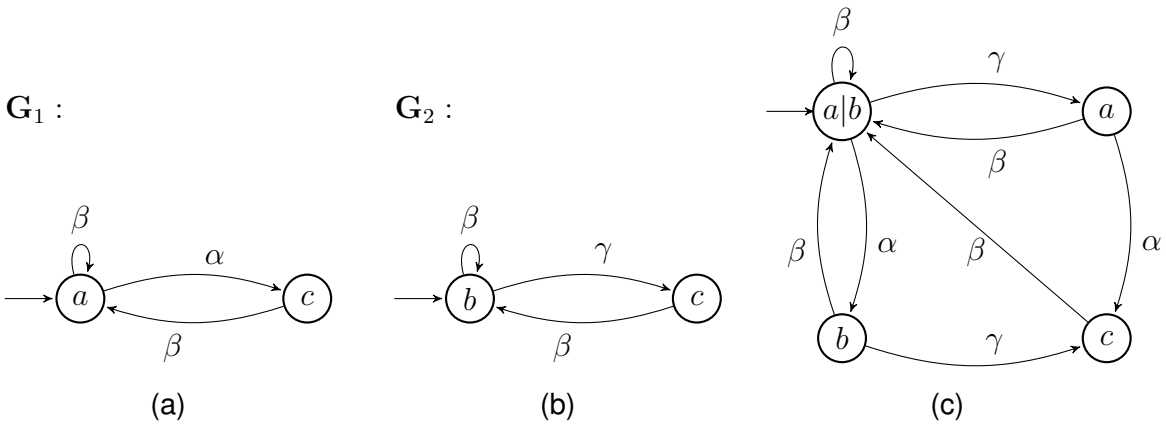
$$\chi_{1\|2}(q_1|q_2) = (\chi_1(q_1) \cap \chi_2(q_2)) \cup (\chi_1(q_1) \backslash \Sigma_2) \cup (\chi_2(q_2) \backslash \Sigma_1). \tag{2.8}$$

In Figure 11, a composition of two CMAs is illustrated. Note that $C_1 = \{a,\ c\}$, $C_2 = \{b,\ c\}$, $\chi_1(0) = \{a\}$, $\chi_1(1) = \{c\}$, $\chi_2(0) = \{b\}$, and , $\chi_2(1) = \{c\}$. Note that the symbols in the states represent the colors, and not the name of the state.

The result is that $C_{1\|2} = \{a,\ b,\ c\}$, $\chi_{1\|2}(0|0) = \{a|b\}$, $\chi_{1\|2}(0|1) = \{a\}$, $\chi_{1\|2}(1|0) = \{b\}$, and , $\chi_{1\|2}(1|1) = \{c\}$.

To design supervisory control for a plant modeled as a CMAs, it is interesting that the specifications and the supervisor be modeled as CMAs. Furthermore, it is interesting that new classes of tasks (new colors) can be introduced in the modeling of the control specification. For example, a buffer specification could introduce a new class of task: the desired quantity of elements in the buffer. In the implementation, the

Figure 11 – Composition of colored marking generators: (a) $\mathbf{G}_1$, (b) $\mathbf{G}_2$, and (c) $\mathbf{G}_{1\|2} = \mathbf{G}_1 \| \mathbf{G}_2$.

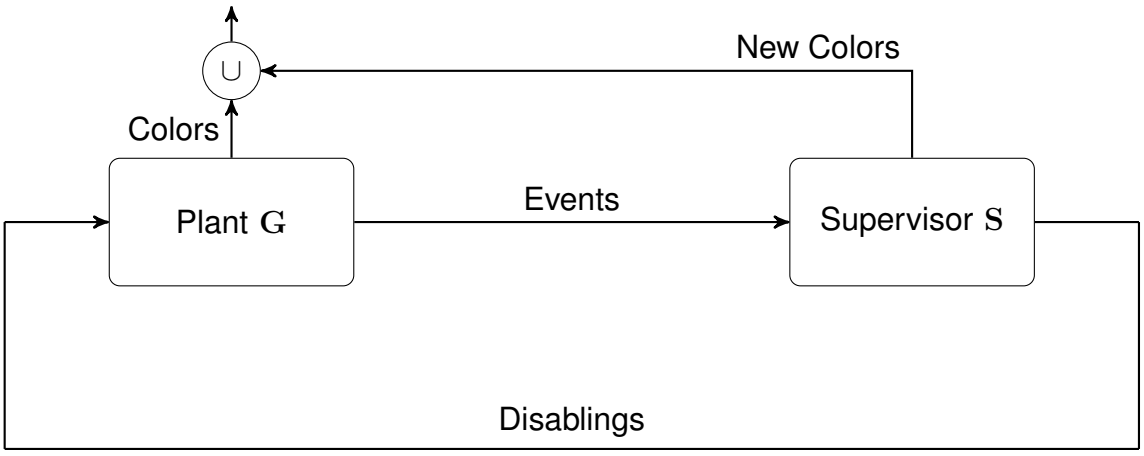


(a) (b) (c)

Source: designed by the author (2022).

supervisor introduces the new colors provided in the specifications. This architecture is depicted in Figure 12.

Figure 12 – Multitasking with colored marking control architecture of DES in closed-loop.



Source: Queiroz, Cury and Wonham (2005).

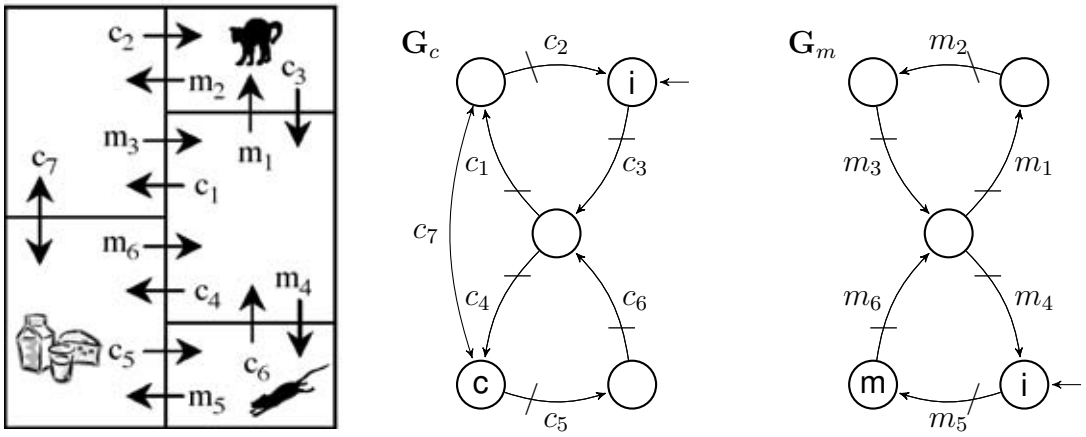
2.2.4.1 Example of Supervisory Control of MTDES

To illustrate the solution of supervisory control of MTDES, the problem of the cat and the mouse maze is recalled (RAMADGE; WONHAM, 1989).

The problem consists in a maze, in which the cat and the mouse cannot be in the same room. The cat needs to feed from milk and the mouse needs to feed from

cheese. However, the problem is that both milk and cheese are disposed in the same room at the maze. The maze is shown in Figure 13.

Figure 13 – Maze and CMA models.



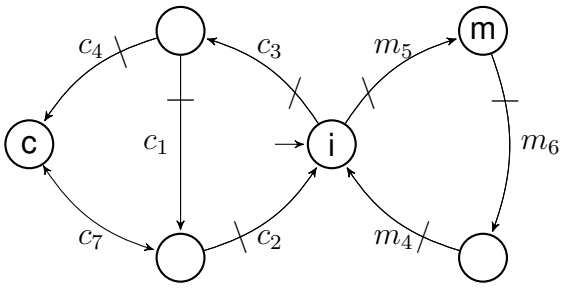
(a) (b) (c)

Source: Queiroz, Cury and Wonham (2005).

The passage through the doors between rooms are named c_i and m_j for the cat and the mouse, respectively, where $i = 1, 2, \ldots, 7$ and $j = 1, 2, \ldots, 6$. Note that all passages are controllable, except for c_7. The models for the cat and mouse are defined as $\mathbf{G}_c$ and $\mathbf{G}_m$, respectively. $C_c = \{\mathrm{i},\ \mathrm{c}\}$ and $C_m = \{\mathrm{i},\ \mathrm{m}\}$ represent the marking colors for the cat and the mouse, respectively, where i represents the return to the initial state for both mouse and cat, which is a "rest" task; c and m represent the task of the cat drinking the milk and the mouse eating the cheese, respectively.

The control solution given by a CMA supervisor, which contains the maximal controllable language and nonblocking w.r.t. the control specifications is shown in Figure 14.

Figure 14 – CMA supervisor solution for the maze problem.



Source: Queiroz, Cury and Wonham (2005).

2.2.5 Computation of the Supervisor

The supervisor's computation is performed through algorithms that evaluate the plant, generating a new automaton that represents the part of the plant's behavior which is in accordance with the control specifications, by disabling controllable events.

The control specifications generate a language which represents all feasible actions for the system. The problem is that the behavior of this language may not be controllable, i.e., this language may be trying to disable an uncontrollable event. In the automaton, a state in which an uncontrollable event is disabled is called a *bad state*.

A bad state can be defined as an undesirable state of the system, from which the execution of an uncontrollable event causes an undesired behavior, such as an accident, or a collision in a CAV system. In the SCT, a bad state means a state in which there is an uncontrollable event which would lead to an undesirable behavior.

In the *supC* algorithm, the *bad states* are identified and removed from the supervisor automaton. Each time a bad state is found, the Trim operation is applied and bad states are searched again, until there are no bad states in the trimmed automaton.

In the *supC* algorithm, the supremal controllable language is obtained, which means a nonblocking maximally permissive language under the desired behavior.

The traditional way of computing the supervisor is performed *offline*, i.e., the computing of the supervisor is performed prior to the execution of the system, and the supervisor found contains all the nonblocking *strings* allowed by the control specification. It should be noted that in case of changes in the plant models or control specifications, the supervisor must be recomputed, or reconfigured.

Another technique found in the literature is the supervisor's *online* calculation with the *limited lookahead* strategy (CHUNG; LAFORTUNE; LIN, 1992). In this case, the supervisor is calculated at runtime, but only a truncated part of the supervisor is calculated. In each state of the supervisor a new truncated supervisor is computed up to N transitions ahead.

In this work, it is proposed that the controllers be computed *online*, but not as in the limited lookahead technique. It is proposed that the supervisor be computed as a whole. To do this, an architecture with an algorithm which computes the control specification is proposed. More details are given in Chapter 5.

2.3 DISCUSSION

In this chapter, the concepts that serve as a basis for understanding the proposal of this work were presented. The concepts about DES were briefly presented, while other works show the same concepts extensively (CASSANDRAS; LAFORTUNE, 2021; CURY, 2001). Two important supervisor synthesis methods were detailed, the monolithic and the local modular.

Comparing the techniques of local control (local modular and supervisor localization), both allow the calculation of local supervisors, and the joint action of modular supervisors results in a closed-loop behavior equal to the behavior imposed by the monolithic supervisor. Therefore, it is feasible and appropriate that the control of multi-agent DES is performed in a distributed manner. However, through the techniques mentioned above, the control of multi-agent systems is limited regarding the characteristics of multi-agent DES. For example, these distributed control techniques do not allow the controlled multi-agent system to have a variable number of agents over time.

Due to the multi-agent nature of CAVs sytems, it is essential that the control architecture proposed in this work considers a solution for multitasking. In the architecture proposed in this work we combine the ideas from the modular local approach with the multitasking approach, exploring multi-agent characteristics of CAVs to create a specific approach for them.

3 RELATED WORKS ON CONTROL OF CAVS

In this chapter, we aim to determine the state-of-the-art of CAVs control techniques (GUANETTI; KIM; BORRELLI, 2018). To do this, a systematic literature review is made regarding the characteristics of interest of this work.

Regarding the control of CAVs, there are many aspects which can be explored, for example: the structure, which can be centralized, decentralized or distributed; the formation control, which can be tri-dimensional, bi-dimensional, uni-dimensional or none; the scope of the coordination which can be an isolated intersection, a roundabout or lane changing; traffic control (GUO; LI; BAN, 2019); and the path planning, which may be optimized individually or globally. The uni-dimensional formation is often called platooning control, and the tri-dimensional and bi-dimensional are commonly applied to unmanned aerial vehicles, such as drones. The path planning may be classified into low level, which considers the continuous dynamics; or into high level which abstracts the continuous dynamics. Due to this complexity with the characteristics of CAVs, we first start this chapter with a brief review on the basic concepts of CAVs.

The organization of this chapter is similar to the Chapter 4, but focused in existent control methods for CAVs. In Section 3.1, the basic concepts of CAVs control are presented, and in the following sections, the systematic review is presented.

3.1 CONCEPTS AND TERMINOLOGY

In this section, the main concepts and common terms in the control solutions for CAVs in the literature are presented.

Considering the complexity of CAVs control, it is usual to delimit a specific scope in each solution presented in the literature. These scopes are presented in the following.

3.1.1 Cruise Control

The objective in cruise control is to control the speed of the vehicle. It is popular in Human-Driven Vehicles (HDVs). Usually, it allows the driver to set a constant speed reference for the controller. In sophisticated HDVs, the cruise control allows to follow

the preceding vehicle, with the same speed, through distance sensors. In CAVs, it allows to follow the preceding vehicle through Vehicle to Vehicle (V2V) communication. An example of cruise control of CAVs can be seen in the work of Moser et al. (2018).

3.1.2 Platooning

Platooning is a very common control solution for CAVs due to its appeal in the improvement of the efficient use of energy. Basically, a platoon is a queue of vehicles, traveling one behind the other to reduce the aerodynamic drag. The communication is usually V2V, however specific topologies are used, such as: predecessor following, predecessor following leader, bidirectional, and bidirectional leader (GUO et al., 2020).

3.1.3 Lane Changing

The objective in the lane changing problem is to plan the trajectory from one lane to another of one CAV, considering the position and speed of other vehicles in the road. An example of lane changing control of CAVs is found in the work of Goulet and Ayalew (2021).

3.1.4 Road Merging

The road merging problem is quite similar to the lane changing problem. The difference is that the merging has to be done in a specific point in the road. Thus, vehicles have to match their velocity to merge with a collision-free behavior. A setup of a highway (main lane) with a slip road is used in many works. More details of road merging control of CAVs can be seen in the work of Rios-Torres and Malikopoulos (2017b).

3.1.5 Roundabouts

The coordination of CAVs in roundabouts can be considered as a mix of road merging and lane changing. Basically, when entering the roundabout, the problem is similar to road merging. And, considering a multi-lane roundabout, there is a problem of lane changing. Furthermore, there is another problem that is treated in the literature, which is the trajectory optimization in a multi-lane roundabout. An example of maneu-

vers planning for CAVs in a single-lane roundabout is found in the work of Debada and Gillet (2018).

3.1.6 Intersections Management

The intersection management for CAVs is basically a coordination of their movement, to pass through the intersection with a collision-free behavior. Most of the works in the literature present solution for an isolated intersection (MIRHELI et al., 2019; ZHANG; CASSANDRAS, 2019; GUAN et al., 2020; KAMAL et al., 2015; STEINMETZ et al., 2018). Some works present solutions for multiple intersections management, such as the work of Chen et al. (2020), but it is interesting to note that this solution is independent of path planning.

3.1.7 Path Planning

Path planning is a generic term in the research field of CAVs, and it is used to refer to many different situations. For example, there may be a path planning in the solutions of lane changing, road merging, intersections management. The problem of path planning may be modeled in a low level approach, where the dynamics of the CAVs are modeled in a continuous state-space; or, in another point of view, it could be modeled in a high level approach, with a discrete state-space, in which the continuous dynamics are abstracted. Furthermore, there are another similar terms used to refer to the same problem, such as trajectory planning and navigation planning (FRANSEN et al., 2020).

3.1.8 Communication Topology

There are different topologies for CAVs communication. The *vehicle-to-vehicle* (V2V) communication is used when there is no infrastructure, i.e., each vehicle trades information directly to others. Another one, is the *vehicle-to-infrastructure* (V2I) communication, where each vehicle trades information with the infrastructure. There is also the V2X communication, which is an acronym for *vehicle-to-all*, considering that the vehicle can communicate to other vehicles and also to the infrastructure. And finally, the *infrastructure-to-infrastructure* (I2I) communication, where infrastructures trade information (SINGH; NANDI; NANDI, 2019) .

The V2V communication is often used in distributed architectures, in which CAVs need to communicate to each other due to the absence of infrastructure. The V2I communication is used in centralized architectures, in which the coordination is made in a centralized infrastructure. And, the I2I communication is used in decentralized architectures, in which the coordination is made within many infrastructures (KHAN et al., 2022).

3.2 METHODOLOGY FOR THE SEARCH OF PAPERS IN THE LITERATURE

In this section we present the methodology used to perform the review on control of CAVs. The steps of this review are the same presented in Section 4.2. As we are searching for control solutions which includes a distributed architecture, scalability and automatic reconfiguration, we have defined a general search phrase: **("connected and automated vehicles") AND (reconfigurable) AND ("path planning" OR trajectory) AND ("distributed control" OR "distributed coordination") AND scalable**. However, we were not able to find any work with this global search phrase, applied to all fields (title, abstract, whole document, etc). Furthermore, as we have tried to discard some of the terms, we have found hundreds of works, which would be unfeasible to select manually. To solve this, we have divided the search in two searches, which are explained next.

3.3 FIRST SEARCH FOR PAPERS

In the first search, we have discarded many terms, to simplify the search phrase, however, the search was limited to the title of the papers. The search phrase applied was: **distributed AND ("connected and automated vehicles" OR "connected and automated vehicle" OR "connected automated vehicle" OR "connected automated vehicles")**. This way, we have found related papers, which may not fulfill all three features. The details of the search in each base is shown in Table 2.

3.3.1 Analysis on the Papers Found in the First Search

In this section a brief analysis is made to contextualize the solutions found in the first search.

Table 2 – Information on the first search for works in the literature.

Base	Search Type	Fields	Quantity
Engineering Village	Quick Search	Title	18
IEEExplore	Command Search	Document Title	8
Science Direct	Advanced	Title	3
SCOPUS	Advanced	Title	13
Springer	Advanced	Title	6
Wiley	Advanced	Title	2
Web of Science	Advanced	Title	12
Total (excluding repeated and unrelated)			12

Source: designed by the author (2022).

Li et al. (2017) propose a framework for distributed platooning control of CAVs. Their framework is applicable to continuous state-space systems. The interesting point is that they use neighbors' information to implement the feedback control for the distributed controllers. They comment about the unstructured distributed control in which an all-to-all communication is required; and the structured control law which may be implemented in an explicit or implicit way.

Du, HomChaudhuri and Pisu (2018) propose an interesting architecture, in which the distributed control is in the infrastructure, not in the CAVs. Thus, the control structure is classified as decentralized. They propose an intersection manager which communicates with other intersection managers next to it, to coordinate the traffic. There is a problem of having a coordinator control at intersections, because every intersection would require an infrastructure installation. If we observe, in HDVs traffic's, not all intersections have traffic lights, because it has a cost involved, and in some cases it is not the most efficient solution.

Feng et al. (2018) propose a distributed method for the formation control of CAVs based on a continuous state-space modeling. Their solution is distributed in the sense of CAVs in a group which are traveling in formation. In the sense of the whole transit, the leading vehicle needs to communicate with a central coordinator. In other words, the solution of path planning for diverse groups of vehicles is centralized. The communication is made in such a way that there is a certain privacy of information between CAVs.

Guo et al. (2020) have developed a distributed method for CAVs platoon control based on continuous state-space modeling. Their method includes model-based predictive control, among other techniques which are not related with the interest of

this thesis. The interesting part for this work, is related with the distribution of control. Their solution is called *distributed adaptive triple-step nonlinear control strategy*, which includes: a distributed reference generator; a distributed adaptive updating law for kinematic and dynamic uncertainties; and a distributed adaptive triple-step nonlinear control law. Their focus is on the control of the dynamics of the platoon. The control of the speed dynamics is made on the leader of the platoon, and passed by as reference for the following CAVs. Therefore, the CAVs share a model structure, but each one has its own parameters, which allow the distribution of the control. Yet, they have to share some variables between adjacent CAVs, in an online manner, such as the position, velocity, acceleration and the constant headway time.

Chen et al. (2021b) present a control architecture for CAVs on a ramp merging situation based on continuous state-space modeling. Their method is a virtual rotation approach, where the merging is considered as a virtual platoon. Thus CAVs adapt their velocity as they were following the CAV at the other lane. They propose a two layer control, in which the virtual rotation process is considered an upper level control that is centralized in the infrastructure. And a low-level distributed control where the trajectories are computed. Therefore, their method is not entirely distributed.

Chen et al. (2021a) propose an adaptive control method for a mixed platoon formation consisted by CAVs and HDVs based on continuous state-space modeling. They propose a distributed architecture with a V2V communication. They use the information shared between CAVs to update adaptive controller parameters. In order to consider HDVs in the platoon CAVs are equipped with distance sensors to measure the distance without communication.

Goulet and Ayalew (2021) propose a distributed control method for CAVs traveling in a multi-lane road based on a continuous state-space modeling. Their method controls the CAVs' maneuvers and speed based on observations and communications. Their method is based on a distributed model-based predictive control.

Wu et al. (2021) develop a distributed control method comprising the merging of CAVs platoons in multilane roads. Their method is based on continuous state-space modeling. The solution is made with a consensus-based algorithm with intrinsic collision avoidance. They propose a two layer architecture in which one layer is responsible for the coordination of the movements and the other layer is responsible for the trajectory tracking control of the CAVs.

Katriniok, Rosarius and Mähönen (2022) propose a distributed control method for coordination of CAVs at intersections. Their method is based on continuous state-space modeling, using the model predictive control technique. They use a V2V communication in which the path of each vehicle is shared with others. In the trajectory computation, an optimization control problem is solved. The objective of the control architecture is to provide ride comfort and efficiency with collision avoidance behavior, while observing the constraints of the vehicle.

Luo et al. (2022) propose a distributed control method for multiobjective CAVs platooning. Their method is based on continuous state-space modeling, using a distributed model predictive control technique. The proposed architecture is formed by a cooperative driving control layer and an energy efficiency optimization control layer.

Prayitno and Nilkhamhang (2022) develop a distributed model reference controller to solve the platooning of CAVs considering a communication topology is modeled by an adjacency matrix, which indicates the flow of information between CAVs. This way the information flow can be redundant, overcoming external disturbances and bounded leader input.

Wu et al. (2022) present a distributed control architecture to manage the trajectory of multiple CAVs in an isolated roundabout with multiple lanes. Their technique is based on the continuous state space and the objective is to optimize the CAVs trajectories considering the difference length of lanes in the roundabout. They propose an interesting approach to the problem with a globally near-optimal solution. When applying a control solution in a distributed architecture, it is unfeasible or inadequate to achieve a globally optimal solution. To overcome this problem, they propose to separate the coordination of CAVs between lanes to the trajectory planning algorithm to ensure globally near-optimal trajectories.

To summarize the analysis of the papers found in the first search, in Table 3 the comparison of the solutions is made regarding the three features proposed in this work. It is important to note that none of the papers proposes a complete solution of path planning, therefore, none of them could be reconfigurable in terms of path planning. Another important observation is that all solutions are based on continuous state-space, which is an evidence that the solutions are most based in the vehicle's continuous dynamics, such as speed and torque. Only two solutions are not distributed and only one solution is not scalable. Therefore, the proposal of this work must focus

efforts on the reconfiguration feature.

Table 3 – Comparison of solutions in the first search, regarding the three features: Distributed, Scalable, and Reconfigurable

Work	Distributed	Scalable	Reconfigurable
Katriniok, Rosarius and Mähönen (2022)	Yes	Yes	No
Luo et al. (2022)	Yes	Yes	No
Prayitno and Nilkhamhang (2022)	Yes	Yes	No
Wu et al. (2022)	Yes	Yes	No
Wu et al. (2021)	Yes	Yes	No
Chen et al. (2021b)	No	Yes	No
Chen et al. (2021a)	Yes	No	No
Goulet and Ayalew (2021)	Yes	Yes	No
Guo et al. (2020)	Yes	Yes	No
Du, HomChaudhuri and Pisu (2018)	No	Yes	No
Feng et al. (2018)	Yes	Yes	No
Li et al. (2017)	Yes	Yes	No

Source: designed by the author (2022).

3.4 SECOND SEARCH FOR PAPERS

For the second search we have used the general search phrase as basis and made some observations:

1. Some works present an intrinsic scalable solution without mentioning it;
    - To solve this, we have removed *scalable* from the search phrase, with the odd of having to read many works with solutions that are not in fact scalable.
2. Some works use synonyms for *reconfigurable path planning*, such as *dynamic path planning* and *online path planning*;
    - To solve this, we have added the terms *online* and *dynamic*.

It is important to note that this is an evidence that scalability and reconfiguration are not established research fields.

Finally, the search phrase applied in the second search was: **("connected and automated vehicles") AND (online OR dynamic OR reconfigurable) AND ("path**

planning" OR trajectory) AND ("distributed control" OR "distributed coordination"). As this search phrase is more restrictive, it was applied to search in the whole document. The details of the second search in each base is shown in Table 4.

Table 4 – Information on the second search for works in the literature.

Base	Search Type	Fields/ Filtering	Quantity
Engineering Village	Quick Search	-	3
IEEExplore	Command Search	-	2
Science Direct	Quick	Articles & Proc.	31
SCOPUS	Quick	T.A.K.	1
Springer	Quick	-	14
Wiley	Quick	-	3
Web of Science	Quick	-	4
Total (excluding repeated and unrelated)			20

Source: designed by the author (2022).

3.4.1 Analysis on the Papers Found in the Second Search

In this section, the analysis of the papers found in the second search is presented.

Liu, Ozguner and Zhang (2017) propose a distributed control method for cooperative highway driving. Their method is applied to a cooperative lane changing, however they assert that lane changing is the basis to build more complex maneuvers, and thus, their method can be extended to other maneuvers such as road merging and roundabouts merging. Their solution is based on a distributed Model Predictive Control (MPC) with continuous state-space modeling. Their architecture is based on three layers: decision consensus; cooperation strategy; and distributed controllers. In the decision consensus layer, vehicles trade information whether the desired maneuver is feasible or not. In the cooperation strategy layer, a vehicle is chosen to send the cooperation command of the desired maneuver for the distributed controller layer. In the distributed controller layer, the trajectory of the maneuver is optimized.

The objective in the work of Petrillo et al. (2018) is to provide a distributed control system for CAVs platooning which is insusceptible to communication problems such as multiple time-varying delays. They use the continuous state-space modeling for the vehicles dynamics and the discrete state-space modeling for the communication

topology. Their architecture consist in one layer controller in cav_i which receives inputs from all CAVs and generate a control action for cav_i.

Wang (2018) proposes an adaptive control for mixed CAV's platoons. The objective is to assure the string stability when one CAV is placed among a platoon of HDVs. Their method is based on the continuous state-space modeling. They use a centralized control structure which receives data from CAVs and estimates the parameters of HDVs.

Xu et al. (2018) present a distributed feedback control scheme to manage CAVs at an unsignalized intersection. Their method is based on the continuous state-space modeling. They propose to manage the intersection with a virtual platoon, which is a similar method to the virtual rotation approach (CHEN et al., 2021b). The virtual platoon method is basically to create a queue, in which the first CAV in the queue is the closer to the center of the intersection, considering any direction.

Zhu and Zhu (2019) propose a barrier-function-based distributed adaptive control for CAVs platoon, considering its non-linearities. They claim that the linearization requires the complete a priori knowledge of the plant, which makes it unfeasible in practice. Their method is based on the continuous state-space, modeled as differential equations.

Li et al. (2020) propose a centralized scalable control for CAVs regarding route planning and trajectory following. Their method is based on parallel computing in a cloud center to support the computational burden of large-scale systems. On interesting observation is that they consider CAVs as fully controllable and observable. One advantage of this method is to obtain a global optimization regarding the path planning, however the costs of operation and maintenance of the cloud computing center may make it unfeasible.

Hu et al. (2021a) propose a control method for intersection management of CAVs. The control architecture is composed of two layers, a centralized supervision level and a distributed execution level. Their method is based on the virtual platoon approach.

Liu et al. (2022) propose a joint control of the traffic signal and trajectories of vehicle platoons. The control structure is based on a single layer, in which the optimization of both problems of cycle timing in the signal and the trajectories of the platoons are modeled. Their method is based on continuous state-space modeling.

Zhang et al. (2020) propose a control structure of CAVs platoon control. The structure is based on two layers, where one layer deals with the internal control of the CAV, regarding the position and velocity of it. The other layer deals with the cooperation of CAVs considering the multi-agent aspects of CAVs systems. The structure is based on continuous state-space through Laplace transfer function modeling.

Wei et al. (2017) propose a control scheme for CAVs trajectories, which can be applied in cruise control and platooning. The modeling is based on the continuous state-space. The communication method relies on a structure installed aside of the roads.

Chen et al. (2020a) present a joint control system for CAVs platoon in the proximity of a connected traffic light. The control structure is distributed, considering a Vehicle to Vehicle and Vehicle to Infrastructure communication (V2X) communication in which the communication of the platoon is made in a V2V and the communication between the traffic light and the platoon leader is made in a Vehicle to Infrastructure communication (V2I) topology.

Katriniok (2020) proposes a consensus-based distributed control method for CAV's lane changing. The structure has two layers, a low level layer, which is related to the local optimization and is run locally in each agent, and a coordination layer for the consensus decision, which is run on the CAV which desires to change lane.

Kneissl et al. (2020) propose a combined scheduling-control method for managing CAVs at an isolated intersection. The structure is formed by two layers, a infrastructure which is in the layer of the scheduling management of the intersection; and the distributed MPC which is in the low level control layer.

Shi et al. (2022) presents a learning based distributed scheme for the longitudinal control of CAVs. The objective of the method is to stabilize traffic oscillations of platoons under communication failures.

Wang and Chen (2021) present a hierarchical control scheme for CAVs 2D flocking. The structure is composed by two layers, in which the low level layer deals with the control of the CAVs trajectory, and the high level layer deals with the flocking coordination between CAVs.

Yang, Feng and Liu (2021) propose a control structure for CAVs coordination at multiple intersections. The environment is considered to be mixed, with CAVs, connected vehicles (not automated) and HDVs. The proposed structure is hierarchical

with three layers. At the lower layer, the vehicle trajectory is controlled regarding the collision-free behavior, optimal trajectories and cruise control. The second layer regards the intersection management. And the top layer regards the corridor level, which manages the link between intersections.

Tajalli, Mehrabipour and Hajbabaie (2021) propose a decentralized control structure for intersection management CAVs, with speed optimization. In the structure, there is a controller for each intersection which communicates to other intersection controllers and to vehicles. It is developed a distributed optimization solution considering a trade off between maximal throughput and minimal speed variation.

Zhuang, Xu and Yin (2020) propose a platoon control for CAVs. The control structure is based on two layers. The higher layer is responsible for planning the platoon formation, and the lower level is responsible for the dynamics of the CAV.

Bakibillah et al. (2019) propose a distributed control scheme for CAVs platoons. The structure is based on MPC with a continuous state-space modeling.

Hu et al. (2021b) propose a control structure for on-ramp merging of CAVs. The control structure is based on two layers. The decision for the coordination of the merging is in the centralized high layer. And the dynamics of the CAVs are controlled in the lower layer, based on the continuous state-space modeling.

In Table 5, a summary considering the features of interest of this works on the control structure of each paper is presented. Similarly to the papers found in the first search, it was not found any work which solves the high-level path planning in a complete manner, i.e., give a starting point calculate the control actions to generate a path which achieve a destination. The path planning found in the works are related to the low level control of position and speed.

3.5 OTHER SEARCHES

Besides the systematic search, some works were found empirically, by performing isolated searches on the databases. An objective analysis is made on these works and presented in Table 6.

Table 5 – Comparison of solutions in the second search, regarding the three features: Distributed, Scalable, and Reconfigurable

Work	Distributed	Scalable	Reconfigurable
Liu, Ozguner and Zhang (2017)	Yes	Yes	No
Petrillo et al. (2018)	Yes	Yes	No
Wang (2018)	No	No	No
Xu et al. (2018)	Yes	No	No
Zhu and Zhu (2019)	Yes	Yes	No
Li et al. (2020)	No	Yes	No
Hu et al. (2021a)	No	No	No
Liu et al. (2022)	No	No	No
Zhang et al. (2020)	Yes	Yes	No
Wei et al. (2017)	No	No	No
Chen et al. (2020a)	Yes	Yes	No
Katriniok (2020)	Yes	No	No
Kneissl et al. (2020)	No	No	No
Shi et al. (2022)	Yes	No	No
Yang, Feng and Liu (2021)	No	No	No
Wang and Chen (2021)	No	No	No
Tajalli, Mehrabipour and Hajbabaie (2021)	No	No	No
Zhuang, Xu and Yin (2020)	Yes	Yes	No
Bakibillah et al. (2019)	Yes	Yes	No
Hu et al. (2021b)	No	No	No

Source: designed by the author (2022).

3.6 DISCUSSION

While executing the research in this chapter, the objective was to find the maximum number of works to comprehend the state of the art of the control of CAVs.

Through the study presented in this chapter, it was possible to gather various important characteristics of the control of a multi-CAV system. One interesting observation is that many works propose a similar strategy where the architecture is divided in, at least, a coordination (or cooperative) control layer and a trajectory optimization control layer (FENG et al., 2018; WU et al., 2021; LUO et al., 2022; WU et al., 2022; LIU; OZGUNER; ZHANG, 2017; ZHUANG; XU; YIN, 2020; HU et al., 2021b).

The classification of the scalability is a complex task, because in this work, we are interested in an architecture which allows the escalation of the control system at runtime. Most works present a modeling technique which depends on the number of vehicles, which allows its escalation. However, they do not present an architecture which allows the escalation at runtime. Nevertheless, we have classified these works

Table 6 – Comparison of solutions in the literature, regarding three features: Distributed, Scalable, and Reconfigurable

Work	Distributed	Scalable	Reconfigurable
Chalaki and Malikopoulos (2021)	No	Yes	No
Chen et al. (2020)	Yes	Yes	No
Ding et al. (2020)	No	Yes	No
Fransen et al. (2020)	No	No	Yes
Guan et al. (2020)	No	Yes	No
Basile, Chiacchio and Marino (2019)	No	Yes	No
Mirheli et al. (2019)	Yes	No	No
Wang, Zhao and Yin (2019)	No	Yes	No
Yu et al. (2019)	No	No	No
Zhang and Cassandras (2019)	No	Yes	No
Steinmetz et al. (2018)	No	Yes	No
Clark et al. (2017)	Yes	Yes	No
Hill and Lafortune (2017)	No	Yes	No
Rios-Torres and Malikopoulos (2017a)	No	Yes	No
Wuthishuwong and Traechtler (2017)	Yes	Yes	No
Kamal et al. (2015)	No	Yes	No
Vemulapalli, Dasgupta and Kuhl (2008)	Yes	Yes	No

as scalable.

We have found very interesting, sophisticated and reliable techniques, however, most (if not all) treat an isolated problem of the control of CAVs. For example, highway merging, intersection management and roundabout merging. Besides, we have searched in all found works for the three features of interest in this work (distributed architecture, scalability, and automatic control reconfiguration), and to the best of our knowledge, we have not found one control architecture for CAVs which embraces them all. This fact support the need for the development of an architecture with the three features, which is presented in the Chapter 5.

4 RELATED WORKS ON SUPERVISORY CONTROL OF MULTI-AGENT DES

The objective of this chapter is to present techniques of control for multi-agent DES. First, the concept of multi-agent is presented, and then a study is carried out to determine the state of the art in the control of multi-agent DES, through a systematic literature review. The purpose of this analysis is to present the existing techniques in the literature, which can serve as a basis for the ideas of this work and align the development of the work with the existing gaps in the literature, corroborating to the originality of the work.

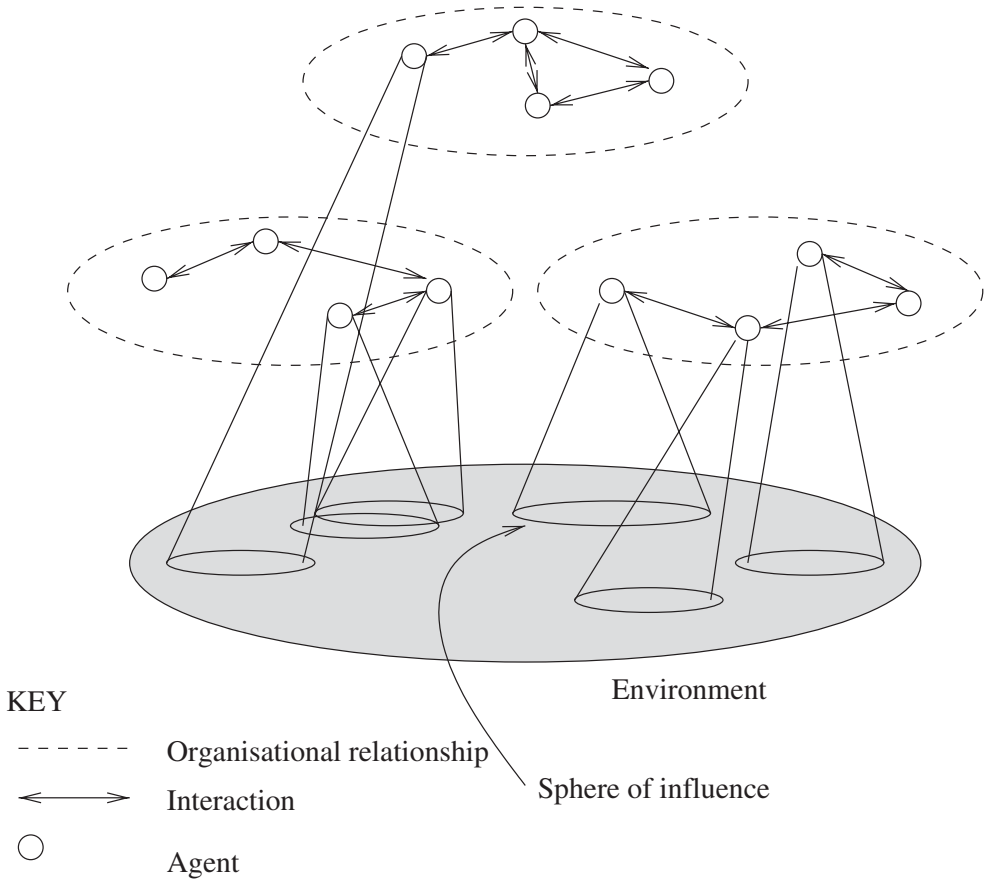
The systematic review of the literature is focused on finding in the literature, the maximum number of related works in order to recognize all existing solutions. It is noteworthy that, as it is "systematic", the review is free from biases that may limit the number of studies found. On the other hand, it is possible that some related works are not found through a systematic search. Thus, the systematic search is complemented with an empirical search, based on the author's experience.

4.1 MULTI-AGENT SYSTEMS

The concept of multi-agent systems was created in the field of computer science, in the 80's. In this field of knowledge, the multi-agent systems can be defined as: "[...] a weakly coupled network of problem solvers which interacts to solve problems which are beyond the capacity or knowledge of each individual problem solver" (DURFEE; LESSER, 1989 apud HüBNER, 2017). Another way to define is: "[...] an organization of agents interacting together in a shared environment" (HüBNER, 2014).

In Figure 15 it is shown that the interaction of an agent can be performed with only a few agents. Thus, it is possible to form interaction groups between agents. Still, in the same figure, it is illustrated that each agent has an area of action within the environment, which characterizes a system with limited accessibility to the environment. If there is no overlap between the areas, the control of each agent is relatively simple, as it is independent of other agents. When there is an overlap, the control of the multi-agent system as a whole becomes more complex because it is necessary for the control to avoid possible conflicts between agents (BORDINI; HüBNER; WOOLDRIDGE, 2007). Another detail in Figure 15 is the fact that not all agents are acting in the environment, that is, over time agents may stop acting, reaching a waiting

Figure 15 – Illustration of a multi-agent system.



Source: Bordini, Hübner and Wooldridge (2007).

state until they receive a new task.

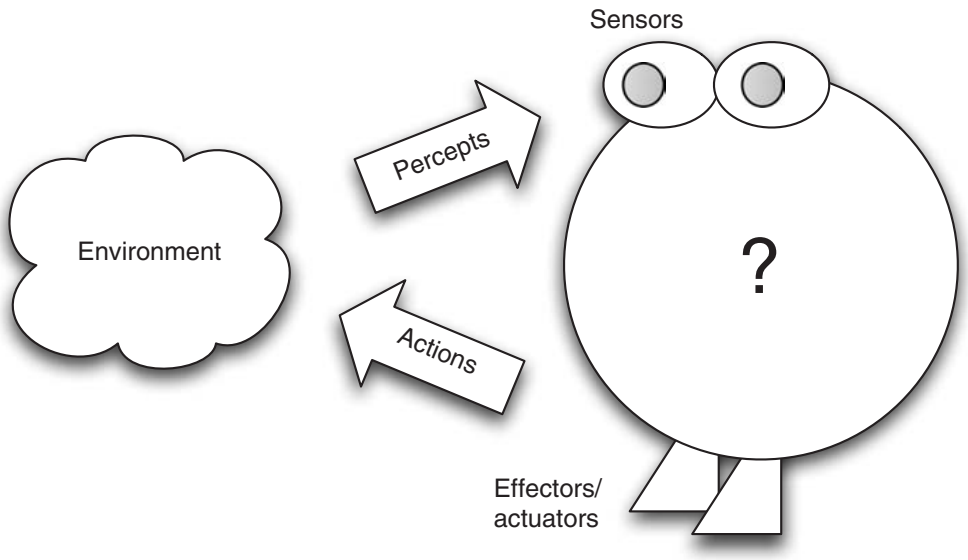
In the engineering research field, multi-agent systems can represent manufacturing plants, communication networks, and robot swarms, among others. The Definition 7 is summarized considering generic multi-agent systems in engineering.

Definition 7 (Multi-agent System)**.** Multi-agent system is a system formed by autonomous agents that interact in the same environment through sensors and actuators, sharing resources and/or dividing tasks. ◊

Agents are individuals who act (interact) in the environment, therefore they can be considered active. These can represent machines, robots, and devices that interact in an environment, composing the multi-agent system. Figure 16 illustrates the fundamental characteristics of an agent.

In Figure 16, the agent's sensors are represented through a pair of eyes, which visualize the state of the environment. Embedded to the agent, decisions are made,

Figure 16 – Components of an agent.



Source: Bordini, Hübner and Wooldridge (2007).

represented by the question mark in the figure. And finally, the actions are carried out in the environment through the actuators, represented by a pair of feet.

The Definition 8 of Weiss (2013, p. 1) completely defines an agent:

Definition 8 (Agent)**.** An agent is a computational entity, such as software or a robot, that can act and sense its environment and is autonomous in the sense that its behavior depends, at least partially, on its own experience. As an intelligent entity, an agent operates flexibly and rationally in a variety of environmental circumstances given its actuators and sensors. Behavioral and rational flexibility are achieved by an agent through fundamental processes such as problem solving, planning, decision making, and learning. As an interactive entity, an agent can be affected in its activities by other agents and even by humans. ◊

Multi-agent systems are naturally complex, so they have several characteristics. In the literature, there are several different ways to characterize them. Also, some authors propose different nomenclatures, but with similar meanings (WEISS, 2013; SINGH; HUHNS, 2006). The most complete and adequate characterization for this work is given by Weiss (2013). In this characterization, multi-agent systems can be divided into three parts: the agents, the environment, and the interaction between the agents. Table 7 lists the characteristics of multi-agent systems. From Table 7, it can be

concluded that multi-agent systems are complex and have several characteristics that specify them.

Table 7 – Summary of characteristics of multi-agent systems.

	Item	Characteristic
Agents	fixed or variable	number
	uniformity	homogeneous or heterogeneous
	objectives	contradictory or complementary
	architecture	reactive or deliberative
	skills	simple to advanced
Interaction	frequency	low to high
	persistence	short to long term
	information exchange level	simple to complex
	standard	decentralized to hierarchical
	variability	fixed or variable
	purpose	competitive to collaborative
Environment	predictability	predictable to unpredictable
	accessibility	unlimited to limited
	dynamic	static or variable
	diversity	little to a lot
	resource availability	restricted to broad

Source: adapted from Weiss (2013).

4.2 METHODOLOGY FOR THE SEARCH FOR PAPERS IN THE LITERATURE

The systematic review carried out in this chapter combines techniques from three different works, and a methodology adapted to the interest of this work was developed (KITCHENHAM; CHARTERS, 2007)(PETERSEN et al., 2008) (OKOLI, 2015). The review was performed according to the following protocol:

1. Determine the search phrase;
    - Find keywords related to the topic and combine them with boolean logic (e.g. control AND discrete event system);
2. Check the search phrase;
    - Perform experimental searches in the databases and check the papers to see if they are related to the topic;
3. Perform the systematic search;

- Perform the final search and make a list with all articles found;

4. Include works found empirically;
    - Add works known by experience, or found by non-systematic searches to the list;
5. Read the abstracts of all articles in the list;
6. Remove repeated articles & unrelated from the list;
7. Read the articles, which remained in the list, in full;
8. Search on the references of the read articles and repeat items 5, 6 and 7 (once);
9. Write a review based on the read articles.

The phrase should be generalist as the objective of the systematic search is to map the largest number of related works. The terms that define the most important characteristics of this work are used, and the tests are carried out in order to verify the realization of the search phrase. So the following sentence was defined: **((multi AND agent) OR multiagent) AND "supervisory control" AND ("discrete event system" OR "discrete event systems")** . Note that in search engines, the following phrases are equivalent: “discrete event system” or “discrete-event system”. Thus, the hyphen does not influence the search.

From the author’s previous experience in the use of search engines for the area of interest, it is known that the following engines (which were used to carry out the search) contain the vast majority of works: IEEExplore; Science Direct; Engineering Village; SCOPUS.

The search was performed considering the conditions presented, and some information is presented in Table 8. After merging the works found in each search base and removing those that appear repeatedly in more than one base, a total of 52 works were found.

After discarding unrelated articles, 27 articles have remained in the systematic search. Adding the empirically found articles, in total, 30 articles were analyzed. Some articles are a continuation of the same work, performed by the same authors, in these cases, they are considered as one work for counting purposes.

Table 8 – Information on the process of searching for works in the literature.

Base	Search Type	Fields	Quantity
IEEExplore	Advanced	Metadata	32
Science Direct	Advanced	Title, abstract, keywords	7
Engineering Village	Quick Search	Subject, abstract, keywords	29
SCOPUS	Advanced	Article title, abstract, keywords	31
Diverse	Empiric	-	4
Total (excluding repeated and unrelated)			30

Source: designed by the author (2019).

4.3 ANALYSIS ON THE PAPERS FOUND IN THE SEARCH

The first work involving the modeling of multi-agent systems was published by Hubbard and Caines (1999). Later, the work was extended by Romanovski and Caines (2003). The technique developed is called *multi-agent product*, as it is based on a vector mapping of the models of each agent. In summary, it is a concurrent association between the automata of each agent, which is similar to the parallel composition of two automata, with the difference that the execution of the events of each agent is synchronous and independent of the others. Thereby, while the model of an agent i transits with the event α, the model of another agent j can transit with the event β, synchronously. One of the limiting problems of this technique, identified through this review, is that transitions in the multi-agent product are defined only if there is a simultaneous transition in all agents. This makes the practical application not very flexible and very limited because, in addition to making synchronicity between agents necessary, it is impossible for an agent to be inactive. Subsequently, a supervisory control method and conditions for distributing supervisors to systems modeled as a multi-agent product were presented (ROMANOVSKI; CAINES, 2006, 2007).

There are three related works that develop fault-tolerant control techniques for multi-agent systems (CHO, 2000; CHO; LIM, 2000, 2001). However, the multi-agent architecture used in these works is based on the synthesis of a local supervisor for each agent, which is independent of the other agents. Therefore, an architecture or conditions for the control of a multi-agent system are not presented. They claim that multi-agent systems differ from decentralized systems in the sense that multi-agent systems are more autonomous and that agents have little or no intersection in control objectives. This is an important feature of multi-agent systems.

The work by Queiroz and Cury (2000) is one of the first works to present a fully distributed control structure, based on SCT. In this technique, local modular supervisors are obtained, which are synthesized from subsystems of the global plant. The idea is that the composition of the local modular supervisors should be equivalent to the monolithic supervisor that would be obtained for the global plant. This technique is quite widespread in works in the literature. When the global system is on a large scale, non-conflict testing can be a problem due to the exponential explosion of states. In this sense, Pena, Cury and Lafortune (2009) propose a method of testing the non-conflict on abstracted models of the supervisors, which reduces the computational complexity of the test.

Gordon and Kiriakidis (2000) present ideas about adaptive control of DESs, which can be applied in multi-agent systems. However, an architecture or conditions for the control of a multi-agent system are not presented. It is evident that adaptability can refer to a failure or the connection of a new subsystem. Thus, the connection of a new subsystem can be explored as the variable number of agents in a multi-agent system. One problem identified is that the adaptability of the system depends on changes that need to be modeled. This makes the solution expensive, or with little flexibility.

The work of Hiraishi (2002) proposes an interesting control architecture for multiagent DESs. The proposed architecture is based on models in tagged Petri nets, where the tags (or *tokens*) are models of subsystems (or agents). This architecture has some important considerations to this thesis that will be discussed here. Agent events can occur concurrently. The system as a whole is asynchronous and is controlled by several controller agents that can observe and control subsets of events. If an event is not controllable by a given agent, it can ask other agents to disable that event. Given the possibility of simultaneous occurrence of events, the technique used depends on a communication structure to avoid errors due to communication problems. A similar but centralized control architecture is presented in works by Čapkovič and Serin (2008) and Čapkovič (2009, 2010).

Wang et al. (2003) present a hierarchical control architecture for multi-agent DESs. The hierarchy simplifies the control of multi-agent systems because decisions can be sent to a coordinator. On the other hand, the need for a coordinator reduces the flexibility of the system.

The strategy adopted in the work by King, Pretty and Gosine (2003) is interesting

because it allows the control of multi-agent DES to be reconfigurable at runtime. Basically, an algorithm is proposed that plans the missions of multiple robots online, generating a model in Petri nets at runtime. This means that if the environment changes, the robots' missions can be reconfigured. The work does not make it clear how these possible changes are passed on to the planner so that it is possible to generate a new model in Petri nets.

The work of Takai and Ushio (2003) proposes improvements in relation to the work of Hubbard and Caines (1999). A comparison between the two methods is demonstrated in a practical application of a manufacturing cell (CHEN; MOREL; RAKOTO-RAVALONTSALAMA, 2004). The improvement proposed by Takai and Ushio (2003) is to relax the constraint on the occurrence of simultaneous events, allowing the asynchronous occurrence of events in the subsystems. This is an important characteristic, because in practical applications of multi-agent systems it is not necessary for the agents to be synchronized in their actions. The nomenclature of "Concurrent Discrete Event Systems" is used to show that the system is formed by subsystems whose state evolution is concurrent, i.e., the evolution of states occurs in parallel and independently of the other subsystems. Takai and Ushio (2003) propose several mathematical operations for DES competitors, as well as the supervisor synthesis. A disadvantage is that the supervisor found is monolithic, acting globally, which highlights the need for a coordinator. Later the method is extended considering the partial observation of events (TAKAI; USHIO, 2005).

Seow, Ma and Yokoo (2004) propose a technique called multi-agent planning. In summary, multi-agent planning can be considered as a monolithic supervisor, which acts in a global way ensuring the desired functioning of the system assuring a non-blocking behavior. It is a simple technique and does not present advantages in relation to the other analyzed techniques.

The work by Karimadini, Lin and Lee (2009) demonstrates a different approach from those presented in the previous paragraph. The approach deals with the supervisory control of multi-agent systems modeled by non-deterministic automata. This approach can be interesting depending on how agents are modeled. For example, given limited communication between agents, some events may be unobservable by some agents. This could mean non-determinism in transitions of agent models. A criticism of this work is that the example used is purely mathematical, not illustrating

a practical application of this technique. Also, although the title emphasizes that the control is distributed, in reality, a monolithic supervisor is proposed for systems where the plant has a distributed nature.

Pham and Seow (2012) introduce a consideration of disjunctive alphabets for the models of each agent, i.e., there are no events in common between the agents. This consideration allows the global behavior of the system to be modeled by independent local models. This solves the problem of modeling multi-agent systems because if there are events in common, it means that the occurrence of these events is synchronous between all agents, which generates synchronization problems in practical applications. They also introduce the concept of minimum necessary communication between agents, whereby only the events involved in the coordination of agents are communicated between them. The control structure is formed by two layers, with a lower layer which considers the local goals of the agent, and a higher layer which solves the coordination between agents.

Cai and Wonham (2010, 2012, 2015) develop an approach called “supervisor localization”. This approach consists of two steps. Initially, the calculation of the monolithic supervisor on a global plant is performed, which is the synchronous composition of the agent models. Next, the supervisor is submitted to an algorithm that “localizes” it, and as an output of this algorithm, the local supervisors for the agents are obtained. In the first publications of this approach, they proposed that the agents’ alphabets be disjoint, but with the evolution of the work, this restriction was relaxed. The main advantage of using this technique is the guarantee of nonblocking and that the language of the closed-loop system is maximally permissive. In addition, the nonblocking feature is achieved in the design phase, without the need to carry out a non-conflict test, as in the local modular approach. Another difference to the local modular approach is that localized supervisors only disable controllable events from the subsystem where the supervisor is localized. However, the disadvantage of the supervisor localization arises when the number of agents is very large, which makes it impossible to obtain a monolithic supervisor due to the exponential explosion of states. Zhang and Cai (2016, 2018) continued the work, exploring the partial observation of events between agents. Cai and Wonham (2014) also explored the same proposal considering the state tree structure, with the justification that the localization algorithm is more efficient using this structure.

Furci, Paoli and Naldi (2013) propose a supervisory control structure applied to the navigation of autonomous robots in search and rescue tasks. The structure allows the control to be reconfigured at runtime since in search tasks the environment is unknown. Therefore, it is desirable that the control be reconfigurable. Still, the presented control is centralized with a monolithic supervisor. Although it is evident in the work that the control is designed for multiple agents, the example presented uses only one agent.

Tatsumoto et al. (2018) demonstrate a multi-agent DES control application. The control is performed by a monolithic supervisor, but calculated with the *limited lookahead* strategy in order to avoid the exponential explosion of states. Although the application in the automation of a warehouse with robots is interesting to demonstrate the control of a multi-agent system, the technique is relatively simple and has no advantages over other techniques.

The first work, in chronological order, within that explored the **similarity** of DES applied to the control of modular systems is the work by Rohloff and Lafortune (2006). It is considered that the models of modules are isomorphic, i.e., their models in automata have the same shape. A global blocking (non-conflicting) test method is presented, without the need to compose all modules, avoiding the exponential explosion of states. Conditions for obtaining local or global supervisors are demonstrated.

The work of Su and Lin (2013) relaxes the isomorphism condition of Rohloff and Lafortune (2006), and the restriction, in this case, is equality between the alphabets of the agents. The technique proposes that the control is carried out through local supervisors for each agent and a global supervisor that governs the behavior of all agents. It is demonstrated that the language of the global supervisor obtained is not maximally permissive, and although the language is not maximally permissive, it is evidenced that there is a gain in flexibility when using this technique. In this case, it is evident that a task coordinator is needed, from which the global supervisor is executed, and there would be greater flexibility if there was no need for a coordinator. Two alphabets are used for each agent, one called private and the other global. This strategy facilitates the structuring of control, as the agents only interact through the global alphabet. However, there is the restriction that the local alphabets must be disjoint, i.e., there are no conflicts between agents in the execution of local events. Furthermore, the global alphabet is formed by events that must be executed synchronously by all

agents, in which case there is also no conflict between agents. Therefore, it is not clear whether the technique used solves the problem of resource disputes. In the example used, all events are considered controllable, but it is not clear if the proposed technique is adequate for alphabets with controllable and uncontrollable events. It can be considered that the proposed control method is suitable for multi-agent systems with the purpose of cooperative interaction in the case where there is no dispute for resources. Therefore, the issue of security in the exchange of information between agents does not apply.

The work of Liu, Cai and Li (2018, 2019b, 2019a) explores some interesting features. One of these characteristics is the similarity between agents (homogeneous multi-agent systems), this allows the modeling of agents to be performed in an identical way, and thus some assumptions can be made. In this way, scalable supervisors were proposed that exploit this similarity between agents, and allow the number of agents to be variable at runtime. Although the variation in the number of agents over the runtime is a certain type of reconfiguration, there are still other more important issues in the reconfiguration that must be addressed, such as changing the control specification and changes to the plant at runtime, which were not explored in their work. Another important feature of their work is the distributed implementation, where initially a monolithic supervisor is calculated and then its location is performed.

The subject of Auer, Dingel and Rudie (2014)'s work is on controlling *multi-threads* in concurrent computing. Although the application used in this work is not related to physical dynamical systems, nor explicitly related to multi-agent systems, there are ideas that can be explored within the scope of this work. It is noteworthy that the proposal of this work is based on the theory of supervisory control and the *multi-thread* computation structure is similar to the structure of a multi-agent system. The supervisor's online calculation is used with the *limited-lookahead* strategy in order to avoid the exponential explosion of states. Auer, Dingel and Rudie (2014) solves the issue of dynamically allocated *threads*, which, in the scope of this work, is equivalent to the variable number of agents. The term "concurrent computing" has nothing to do with the purpose of competitive interaction in the case of multi-agent systems since concurrent *threads* indicate a parallel computation that can be performed on different hardware, instead, *threads* solve part of a global problem. Therefore, in the work of Auer, Dingel and Rudie (2014), the issue of security in the exchange of information

between agents does not apply. The reconfiguration issue does not apply either, because despite the *threads* being dynamically created, they are based on the same set of functions originally determined, as in the case of the example used for the read and write permission control of *threads*.

Dulce-Galindo et al. (2019) present a hybrid control structure applied to the navigation of multiple robots. The implementation of control within the scope of SCT is carried out with local modular supervisors (QUEIROZ; CURY, 2000), however, there is a centralized algorithm for task assignment. Due to these characteristics, the control structure is considered decentralized. Although part of the control structure is made up of algorithms, this work, among the analyzed in this chapter, is the one that presents the greatest advantages in relation to reconfiguration, since: the number of robots can vary over time; there may be unknown obstacles in the navigation environment; tasks can be assigned to robots at runtime. Also, the solution to avoid collisions between robots in competitive cases is performed through a reactive planner. This reactive planner is not based on SCT but on mechanical physics, and knowledge of the robot's global position is necessary to solve the problem. A disadvantage of the reactive planner is that its solution is not deterministic, and there is a possibility of not obtaining a solution due to local minimum.

The work of Dulce-Galindo et al. (2019) is extended in Dulce-Galindo et al. (2022). They enhance the solution regarding the problem of local minimum in the previous work. They enhance the control architecture to obtain a scalable solution regarding the number of CAVs. The structure of control regarding the reactive and deliberative planners is very similar to the previous work. The architecture is solved in two structures, in a centralized structure, and in a distributed structure. In both cases the task assignment is made by a scheduler, which is in an infrastructure. The difference is that in the distributed structure, the scheduler is considered as an agent aside to the robots and therefore the control is distributed and embedded in each agent.

4.4 DISCUSSION

A discussion is made on the works found in the literature considering the three aspects (distributed, scalable and reconfigurable), which are important to show the originality of the thesis proposal of this work.

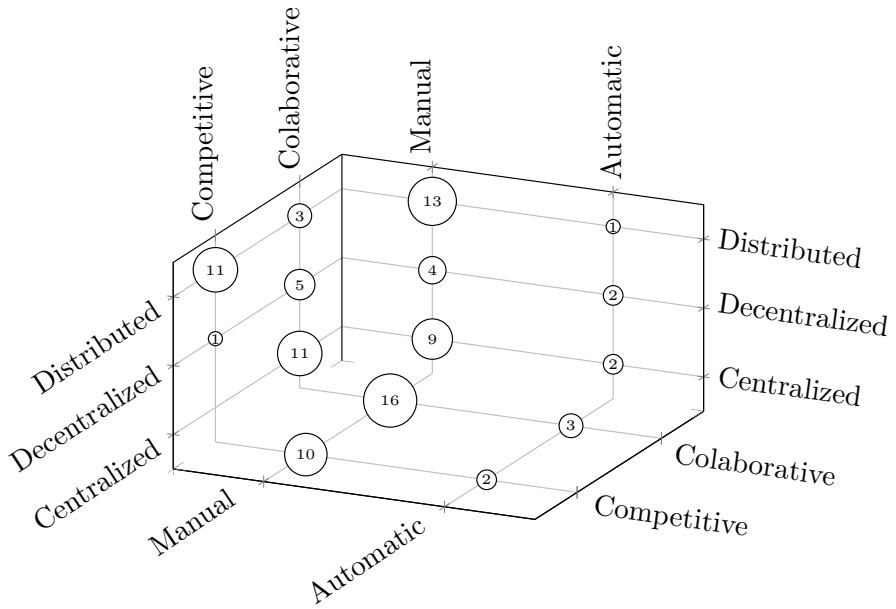
The first aspect refers to the implementation of the control architecture of the

solutions found, which can be centralized, decentralized, or distributed. The second aspect refers to the purpose of interaction in multi-agent systems, which can be competitive or collaborative. And, the third aspect is the possibility of reconfiguration of the control system, which can be performed manually (offline) or Automaticlly (at runtime).

In Table 9 a classification is made on the solutions analyzed in this chapter, regarding the four characteristics: Structure, Scalable, Reconfigurable and Interaction. It is important to analyze the interaction characteristic of the control solutions, because due to the competitive nature of CAVs, collaborative control solutions are not applicable. It is important to note that the purpose of interaction (collaborative or competitive) is not explicited in the literature, making it difficult to find a search phrase which excludes works focused in collaborative interaction.

Figure 17 shows a bubble chart, summarizing the number of works found with the characteristics analyzed.

Figure 17 – Bubble chart, where the characteristics of the works found in the literature are summarized.



Source: designed by the author (2019).

The analysis on the bubble graph of Figure 17 is made considering two aspects at a time. It is possible to observe that the largest number of works found (16 works), are the works that present a centralized architecture for collaborative multi-agent systems, without reconfiguration at runtime. In descending sequence, 13 works present a

Table 9 – Comparison of solutions on control of DES, regarding the four characteristics: Structure, Scalable, Reconfigurable and Interaction

Work	Structure	Scalable	Reconf.	Interaction
Hubbard and Caines (1999)	Decentralized	No	Manual	Collab.
Cho (2000),	Distributed	No	Manual	Collab.
Gordon and Kiriakidis (2000)	Decentralized	No	Automatic	Collab.
Cho and Lim (2000)	Distributed	No	Manual	Collab.
Cho and Lim (2001)	Distributed	No	Manual	Collab.
Hiraishi (2002)	Decentralized	No	Manual	Collab.
Wang et al. (2003)	Decentralized	No	Manual	Collab.
King, Pretty and Gosine (2003)	Centralized	No	Automatic	Collab.
Romanovski and Caines (2003)	Decentralized	No	Manual	Collab.
Takai and Ushio (2003)	Centralized	No	Manual	Collab.
Chen, Morel and Rakoto-Ravalontsalama (2004)	Centralized	No	Manual	Collab.
Seow, Ma and Yokoo (2004)	Centralized	No	Manual	Collab.
Takai and Ushio (2005)	Centralized	No	Manual	Collab.
Romanovski and Caines (2006)	Distributed	No	Manual	Compet.
Rohloff and Lafortune (2006)	Distributed	No	Manual	Compet.
Romanovski and Caines (2007)	Distributed	No	Manual	Compet.
Čapkovič and Serin (2008)	Centralized	No	Manual	Collab.
Čapkovič (2009)	Centralized	No	Manual	Collab.
Karimadini, Lin and Lee (2009)	Centralized	No	Manual	Collab.
Čapkovič (2010)	Centralized	No	Manual	Collab.
Pham and Seow (2012)	Distributed	No	Manual	Compet.
Cai and Wonham (2012)	Distributed	No	Manual	Compet.
Furci, Paoli and Naldi (2013)	Centralized	No	Automatic	Collab.
Su and Lin (2013)	Distributed	No	Manual	Compet.
Cai and Wonham (2014)	Distributed	No	Manual	Compet.
Cai and Wonham (2015)	Distributed	No	Manual	Compet.
Tatsumoto et al. (2018)	Centralized	No	Manual	Collab.
Liu, Cai and Li (2018)	Distributed	Yes	Manual	Compet.
Liu, Cai and Li (2019a)	Distributed	Yes	Manual	Compet.
Dulce-Galindo et al. (2019)	Decentralized	No	Automatic	Compet.
Dulce-Galindo et al. (2022)	Distributed	Yes	Automatic	Compet.

distributed structure without reconfiguration at runtime. We have found 11 works with distributed structure and competitive interaction; and 11 works with centralized structure with collaborative interaction. This is an evidence that collaborative interactions are most suitable to be coordinated by a central element, and competitive interactions are most suitable to be coordinated by themselves (distributed).

Finally, observing the graph in Figure 17, it is possible to state that only one work was found presenting a control system with a distributed structure, with scalability and automatic reconfiguration at runtime. Furthermore, most control systems are adequate

for collaborative multi-agent systems, which are not adequate for CAVs systems. This evidences the originality of the proposal of this work.

Considering the objectives of this thesis, the work of Dulce-Galindo et al. (2022) can be considered the only work found in the literature, which contemplates the solution to the three features of interest in this thesis: Distributed, Scalable and Reconfigurable.

Comparing the scalability in both works, the proposal differs in the application. In the work of Dulce-Galindo et al. (2022), the application is for autonomous robots in a determinate environment, such as warehouses and manufacturing industries. In these environments, the scale of the system is usually about dozens to hundreds robots, in an area from 1000 to 10000 square meters. With this application, it is adequate the use of a coordinator (scheduler). In this thesis, the application is for CAVs in urban environments which has a scale of thousands to millions of vehicles in areas up to 10000 square meters. In this application, the use of a coordinator within an infrastructure is unfeasible due to the number of vehicles and the size of the environment. An objective comparison between both works is made in Table 10, to show some differences.

Table 10 – Comparison of the solutions proposed in this thesis and by Dulce-Galindo et al. (2022).

Characteristic	This thesis	Dulce-Galindo et al. (2022)
Structure	Distributed	Distributed
Scalability	At runtime	At runtime
Reconfiguration	At runtime	At runtime
Modeling	Map based	Orientation based
Infrastructure	Not needed	Needed
Environment	Known	Partially Known

Source: designed by the author (2022).

5 CONTROL ARCHITECTURE

In this chapter we propose a distributed, scalable and reconfigurable control architecture for CAVs, ensuring the nonblocking and collision-free behavior. The architecture can be applied to solve problems such as dynamic path planning, multiple intersections management, roundabouts management and road merging.

To solve the exponential growth of states problem, it is proposed that the control be synthesized in a fully distributed way. For this to be possible, the characteristics of similarity between CAVs are explored, avoiding the composition of a global plant. The blocking and collision between CAVs problems would be easily solved by calculating a monolithic supervisor over the global plant, however, this method is not feasible to implement in this context. For the problem of collision between CAVs, it is proposed that the CAVs have the control action of disabling events of other CAVs. For the blocking problem, in order to avoid the costly verification of blocking, it is proposed that an alternative path be recalculated for each blocked agent, with the temporary removal of blocked paths. In this way, a new controller is computed at runtime, creating a control alternative that is nonblocking. For this, some necessary and sufficient conditions for the plant model (map) of the CAVs are defined.

Part of the proposed solution was inspired by the TCP networking protocol, in which there is a timeout to resend a packet. This means that not always a packet will be received, and when it is not received, after a while, it is resent. Note that there is no sophisticated method which identifies why the packet was lost or where it was lost, nor the infrastructure is robust enough to never lose a packet. The solution is simple, the sender just waits for confirmation, and if the acknowledgement is not received, the packet is sent again. It is not the most efficient use of the network, but the package loss rate is too low to justify a sophisticated solution. The inspiration was to look at the path (packet) of a CAV in such a way that it can be re-traced (resent), when a blocking (packet loss) occur.

5.1 PRELIMINARIES

In this section, some concepts and definitions are presented, in order to facilitate the understanding of the chapter. In the next paragraph, we summarize some common

concepts and definitions. Then, we present the relabeling function, the concept of circular accessible state & automaton, and the concept of reconfigurable multitask DES, all of them developed in the context of this thesis.

To be more convenient to the reader, we recall a summary of the definitions stated in Chapter 2. The finite set of events is called Σ. $|\Sigma|$ denotes the number of elements in Σ, and is valid for any kind of set. A string is a finite-length sequence of events in Σ. Given a string s, its length, i.e., number of events including repetitions is denoted by $\|s\|$. The set of all strings formed by events in Σ is denoted by Σ^*. Any subset of Σ^* is called a language over Σ. Let L be a language over Σ. The prefix-closure of L is denoted by $\overline{L}$. The notation $\Sigma_a = \Sigma_b \setminus \Sigma_c$, means that Σ_a is formed by all elements of Σ_b which are not contained in Σ_c. An automaton is a six-tuple $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$, where Q is the set of states, Σ is the finite set of events, δ: $Q \times \Sigma \to Q$ is the partial transition function, Γ is the active event function, q_0 is the initial state, and Q_m is the marked state set. δ is extended such that δ: $Q \times \Sigma^* \to Q$. The accessible part of $\mathbf{G}$ with respect to q is $Ac(\mathbf{G}, q) = (Q_{ac}, \Sigma, \delta_{ac}, \Gamma_{ac}, q, Q_{m_{ac}})$, where $Q_{ac} = \{q' \in Q : (\exists s \in \Sigma^*)(\delta(q, s) = q' \text{ is defined})\}$, and $\delta_{ac} = \delta|_{Q_{ac} \times \Sigma \to Q_{ac}}$. The coaccessible part of $\mathbf{G}$ with respect to q is $CoAc(\mathbf{G}, q) = (Q_{coac}, \Sigma, \delta_{coac}, \Gamma_{coac}, q_{0_{oac}}, Q_m)$, where $Q_{coac} = \{q \in Q : (\exists s \in \Sigma^*)(\delta(q, s) \in Q_m\}$; $q_{0_{oac}} = q$, if $q \in Q_{coac}$, undefined otherwise; and $\delta_{coac} = \delta|_{Q_{coac} \times \Sigma \to Q_{coac}}$. The trim part of $\mathbf{G}$ with respect to q is obtained by computing both accessible and coaccessible parts. $L(\mathbf{G})$ represents all physically possible behavior of an automaton $\mathbf{G}$, and $L_m(\mathbf{G})$ represents the behavior of $\mathbf{G}$ in which tasks are completed. $L(\mathbf{G}, q)$ and $L_m(\mathbf{G}, q)$ represent, respectively, the language and the marked language of $\mathbf{G}$ starting from state q.

In order to represent a modular DES in which the subsystems models have a similar structure, we introduce a labeling function. Consider a labeling function R_i : $\Sigma \to \Sigma_i$ with a biunivocal relation for every $\sigma \in \Sigma$, such that $|\Sigma| = |\Sigma_i|$ and $R_i(\sigma) = \sigma_i$ for $\sigma_i \in \Sigma_i$; and $R_i^{-1}(\sigma_i) = \sigma$; where $i \in \mathbb{N}$. Note that $R_i(\sigma) \neq R_j(\sigma)$, however $R_i^{-1}(\sigma_i) = R_j^{-1}(\sigma_j)$, for $i \neq j$.

We extend this function to languages & strings; and, state sets & states. We also extend this function to automata, by applying it individually for each tuple, i.e., for $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$:

$$R_i(\mathbf{G}) = (R_i(Q), R_i(\Sigma), R_i(\delta), R_i(\Gamma), R_i(q_0), R_i(Q_m)), \tag{5.1}$$

where $R_i(\delta) : R_i(Q) \times R_i(\Sigma) \rightarrow R_i(Q)$, and $R_i(\Gamma) : R_i(Q) \rightarrow R_i(\Sigma)$.

The concept of blocking means at least one CAV would be stuck, i.e., unable to achieve its destination (marked state). In the automata model, $\mathbf{G}$ is nonblocking when $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$ (CASSANDRAS; LAFORTUNE, 2021).

Next, we present the definitions of circular accessible state and automaton, in terms of DES, which are useful for the solution proposed in Section 5.2. It is important to note that this definition is different to the *strongly connected automaton*, which does not satisfies the requirements for the solution.

Definition 9 (Circular Accessible State)**.** A state $q_x \in Q$ is said to be circularly accessible if $\forall q_y \in Q$, with $q_y \neq q_x$, $\exists s_x, s_y \in \Sigma^*$, with $s_x = \sigma_{x_1}\sigma_{x_2}\ldots\sigma_{x_m}$, and $s_y = \sigma_{y_1}\sigma_{y_2}\ldots\sigma_{y_n}$ such that the following hold:
i) $\delta(q_y, s_y) = q_x$ and $\delta(q_x, s_x) = q_y$; and
ii) $\delta(q_y, \sigma_{y_1}) = q_{y_2}$, $\delta(q_{y_2}, \sigma_{y_2}) = q_{y_3}$, $\ldots$, $\delta(q_{y_n}, \sigma_{y_n}) = q_x$, $\delta(q_x, \sigma_{x_1}) = q_{x_2}$, $\delta(q_{x_2}, \sigma_{x_2}) = q_{x_3}, \ldots,$ $\delta(q_{x_m}, \sigma_{x_m}) = q_y$; and
iii) $q_y \neq q_{y_2} \neq \ldots \neq q_{y_n} \neq q_x \neq q_{x_2} \neq \ldots \neq q_{x_m}$. $\Diamond$

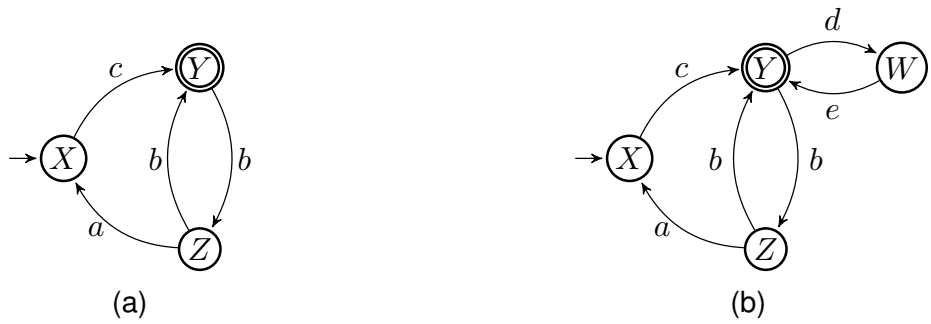
In words, a state q_x is circularly accessible if for all state q_y of the automaton there is a closed path, which passes through q_y and returns to q_x, without visiting any state more than once.

Definition 10 (Circular Accessible Automaton)**.** An automaton $\mathbf{G} = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$ is said to be circularly accessible if $\forall q \in Q$, q is circular accessible. $\Diamond$

For the illustration of the circular accessibility, consider the automaton $\mathbf{F}$ in Figure 18a, in which all states are circularly accessible, therefore, $\mathbf{F}$ is circularly accessible. Now, consider the automaton $\mathbf{H}$ in Figure 18b, state W does not fulfill the conditions for $\mathbf{H}$ to be circularly accessible, since all paths from state X to state W pass through state Y, and all paths from state W to X pass through the same state Y.

Queiroz, Cury and Wonham (2005) proposed a notion of multitasking DES. This notion is extended in Queiroz and Cury (2005), in which each module has its task and the tasks may be accomplished asynchronously, i.e., not at the same time. A task is defined as an objective of the control system, and therefore, a multitasking system, with different classes of tasks, has multiple distinct objectives.

Figure 18 – Examples of (a) circular accessible automaton $\mathbf{F}$ and (b) non-circular accessible automaton $\mathbf{H}$.



Source: designed by the author (2022).

We extend this notion of multitasking in Definition 11 by considering that the agent (module) receives a new, different task after the accomplishment of the current task.

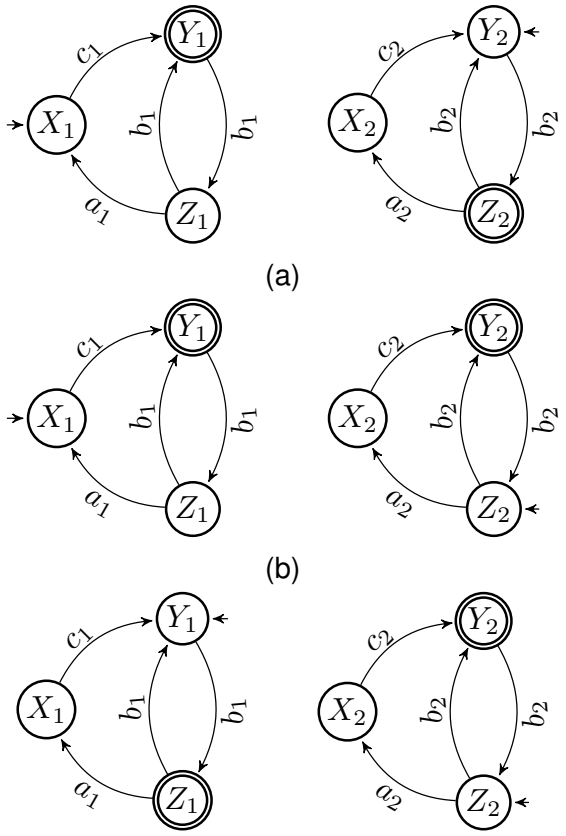
Definition 11 (RMTDES)**.** A DES is called a Reconfigurable Multitasking Discrete Event System if it has the following properties: (i) it is composed of two or more concurrent subsystems that have independent classes of tasks; and (ii) the initial and marked states of each subsystem may be instantly modified. ◊

It is important to note that multitasking means that the tasks of different subsystems do not need to be accomplished at the same time.

Queiroz, Cury and Wonham (2005), and Queiroz and Cury (2005) introduced the Colored Marking Automaton as a model that distinguishes classes of tasks in DES. The colored marking intends to distinguish different classes in a monolithic system. In our work, each class of task is biunivocally associated with each subsystem, and in our method, we do not compose the subsystems model, therefore, there is no need to use a label to represent different classes of tasks.

An example is provided in Figure 19 to illustrate the concept of RMTDES. In the initial instant $t_1 = 0$ (Figure 19a), the subsystem $\mathbf{F}_1$ has Y_1 as its marked state and subsystem $\mathbf{F}_2$ has Z_2 as marked state. In the second time instant $t_2 > t_1$ (Figure 19b), the subsystem $\mathbf{F}_2$ has completed its task and received a new one with Y_2 as its marked state. Observe that when a subsystem receives a new task, both initial and marked states change. In the third time instant $t_3 > t_2$ (Figure 19c), the subsystem $\mathbf{F}_1$ has completed its task and received a new one with Z_1 as its marked state.

Figure 19 – Illustration of automata of RMTDES system $\mathbf{F}$, formed by $\mathbf{F}_1 = R_1(F)$ and $\mathbf{F}_2 = R_2(F)$. Three different instants of time are shown: (a) time $t_1 = 0$; (b) time $t_2 > t_1$; and (c) time $t_3 > t_2$.



Source: designed by the author (2022).

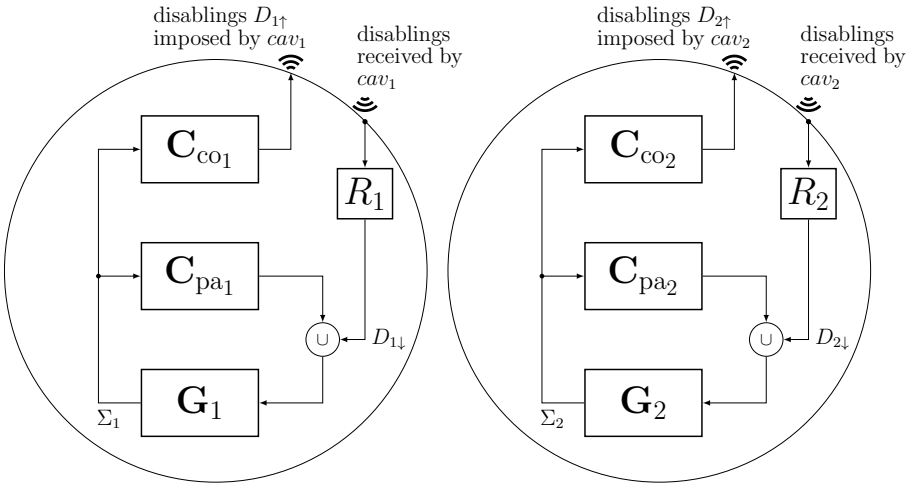
5.2 CONTROL ARCHITECTURE

In this section, we present the proposed architecture for the distributed control of multi-CAVs systems based on RMTDES. The control is distributed and embedded in each CAV, where the control actions are made, considering the information sent by other CAVs.

Given a multi-CAV system, a CAV is denoted by cav_i with $i = 1, 2, 3, \ldots, n$; where n represents the number of active CAVs. The controlled behavior of cav_i is represented by three automata models, a local plant $\mathbf{G}_i$, a path controller $\mathbf{C}_{\mathrm{pa}_i}$, and a coordination controller $\mathbf{C}_{\mathrm{co}_i}$. The proposed architecture is shown in Figure 20 with two CAVs, denoted by cav_1 and cav_2. This architecture is based on the framework of Ramadge and Wonham (1989), in which $\mathbf{G}_i$ represents the local plant and $\mathbf{C}_{\mathrm{pa}_i}$ acts as a local supervisor, observing events generated by $\mathbf{G}_i$ and disabling controllable events for $\mathbf{G}_i$. The controller $\mathbf{C}_{\mathrm{co}_i}$ acts as a supervisor who observes events generated by $\mathbf{G}_i$ and disables

controllable events of other CAVs plants. Thus, the local plant $\mathbf{G}_i$ receives the union of disablings from $\mathbf{C}_{\mathrm{pa}_i}$ and $\mathbf{C}_{\mathrm{co}}$ of other CAVs.

Figure 20 – Control structure for a multi-CAV system.



Source: designed by the author (2022).

The local plant model of a CAV is based on the map, modeled as an automaton. The states represent regions in space and transitions represent movement between regions. The path controller is responsible for the control actions concerning the destination of the CAV. The specifications for the path controller are individual for each CAV. The coordination controller is responsible for the control actions which avoid collisions with other CAVs. The specifications for the coordination controller are individual for each CAV, however, they are based on a common rule for all CAVs which provides the same level of priority for all CAVs in the coordination.

We propose the use of modular RMTDES for the modeling of a multi-CAV system, in which, each CAV has its automaton model, and may be considered a module of the whole system. We assume that all events are controllable, but we foresee that uncontrollable events could be included in the model of this architecture. We also consider that two or more events cannot occur at the same instant of time (in terms of implementation).

Consider a directed graph $RG = (Q, \Sigma, \delta, \Gamma)$, called *root graph*, which represents the map model, and the generator $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, \Gamma_i, q_{0i}, Q_{mi})$, which is the local plant for cav_i, where $Q_i = R_i(Q)$, $\Sigma_i = R_i(\Sigma)$, $\delta_i = R_i(\delta)$ and $\Gamma_i = R_i(\Gamma)$. This means that $\mathbf{G}_i$ has the same state transition structure based on the RG, but could have different initial

and marked states (CAI; WONHAM, 2012). The model of the $\mathbf{G}_i$ for each CAV has its states and events labeled by its index. In other words, two events with the same name, but different indexes, represent the same action but are executed by different CAVs.

We explore the map sharing between CAVs, which allows us to simplify the distribution of the control, without the need of obtaining a single (monolithic) model which describes the global behavior of the system. When the same map (i.e. geomorphic automata models) is used, each CAV knows intrinsically all possible paths. In other words, CAVs share the same resources and it is necessary to handle the resource dispute.

It is important to observe that, if there is only one CAV in operation, its actions are ruled by the path controller $\mathbf{C}_{\mathrm{pa}_i}$, and there are no restrictions (no disabled events) imposed by any other CAV's coordination controller. Yet, if there are several CAVs in operation, and they do not cross paths, the coordination controller of one CAV does not have any effect on other CAVs, and their operation is ruled solely by their path controllers.

The coordination controller guarantees a collision-free behavior in resource dispute between CAVs, however, to avoid conflicts a CAV may be blocked. In other words, all events of a CAV's path may be disabled, causing it to be blocked. Thus, the concept of *momentary blocking* is proposed in Definition 12. This new concept of blocking is called momentary as it is solved by the reconfiguration of the controllers in runtime.

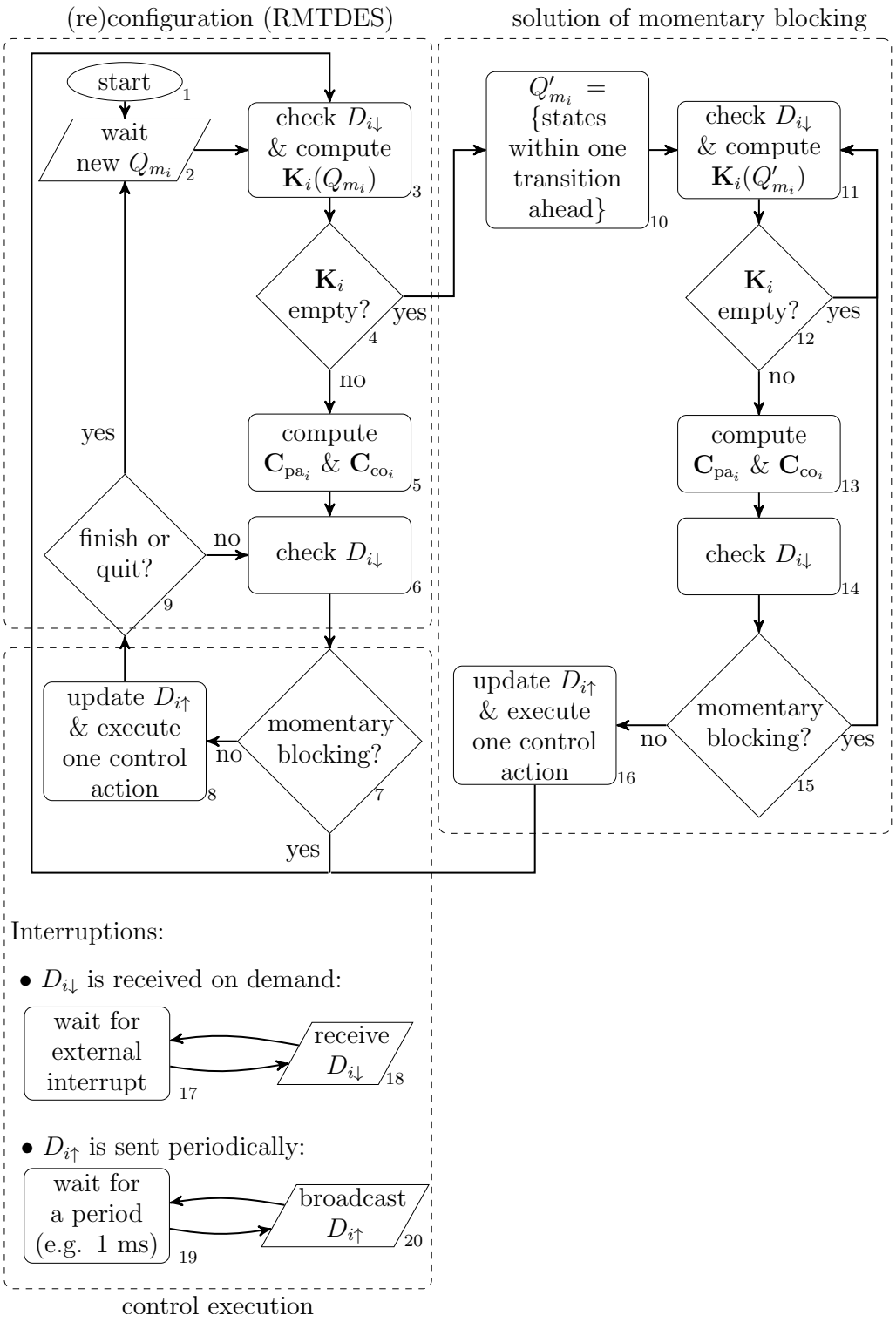
Definition 12 (Momentary Blocking)**.** Occurs when the evolution of states in the current path is momentarily blocked due to the disabled events imposed by other CAVs, requiring the computation of a new path. ◊

The solution to momentary blocking is given in Section 5.2.1.

5.2.1 Reconfigurable Structure

The flowchart in Figure 21 represents the CAV's internal operation, in which the reconfiguration and the momentary blocking solution are given. The embedded reconfiguration procedure makes it possible for each agent to receive a new destination when the path is finished, or when the current path is interrupted. The momentary blocking solution makes use of the reconfiguration to recalculate the path of the CAV.

Figure 21 – Flowchart of a CAV's internal operation.



Source: designed by the author (2022).

Note that all blocks are numbered to better understand the function of the flowchart. Also, note the interruptions at the bottom of the flowchart. The disablings from other

CAVs are received instantly at block 18, by demand, and stored in the memory of cav_i for consultation. The disablings sent by cav_i are updated in the flowchart, however, they are being cyclically sent over a period of time at block 20.

From block 1, the flow starts. At block 2, the CAV receives an input (operator or passenger) of the destination Q_{m_i}.

At block 3, the disablings for cav_i $D_{i\downarrow}$ are checked in the memory. Then, the specification for the path controller ($\mathbf{K}_i$) is computed accordingly to Algorithm 1, in Section 5.2.2. At block 4, a verification of the existence of $\mathbf{K}_i$ is made, if it is empty, it means that the CAV is momentarily blocked. If $\mathbf{K}_i$ is not empty, the path and coordination controllers ($\mathbf{C}_{\mathrm{pa}_i}$ & $\mathbf{C}_{\mathrm{co}_i}$) are computed at block 5. At block 6, the received disablings are checked, and in sequence, the momentary blocking condition is checked at block 7.

Formally, cav_i is momentarily blocked when:

$$\Gamma_{\mathrm{pa}_i}(q_i) \subseteq D_{i\downarrow}, \tag{5.2}$$

where q_i is the current state of the cav_i. This means that all feasible events in the current state of the closed loop behavior are disabled by other CAVs.

If the cav_i is not momentarily blocked, going to block 8, the disablings cyclically sent to other CAVs are updated and the control action (transition of state) is executed. At block 9, it is checked if the path is finished or if the operator/passenger has interrupted the path. If not, the flow goes again to block 6. If the path is finished/interrupted at block 9, then the process starts again by receiving a new destination (marked state) at block 2.

Going back to block 7, if the cav_i is momentarily blocked, a new path is computed flowing to block 3, in which all possible paths from the cav_i's current state to the destination state are computed, excluding the paths which contain events momentarily disabled by other CAVs.

Going back to block 4, in most cases, the specification $\mathbf{K}_i$ is not empty. It will be empty in two specific situations:

- The CAV is in the imminence of reaching its destination, but it is occupied;
- The CAV is surrounded by other CAVs, making it impossible to move in any di-

rection.

In the first case, a relaxation is made regarding the shortest path. It is allowed that the CAV may move one transition ahead in any direction, as long as it is not disabled (block 10). This action is needed to avoid CAVs to remain blocked in the case of two or more CAVs wanting to occupy each other positions as their final destinations. In other words, it is a dispersion in the case of two or more CAVs in the imminence of reaching their destinations, but they are respectively occupied by other CAV. A solution to this case is illustrated in Section 6.1.2.

In the second case, the momentary blocking solution keeps trying to recompute its path until at least one adjacent state is freed (blocks 11 and 12).

The part of the flowchart regarding the dispersion starts by marking all states of $\mathbf{K}_i$ within one transition ahead of the current state of the CAV, at block 10. The set of marked states within one transition ahead is denominated Q'_{m_i}. In the sequence, the disablings $D_{i\downarrow}$ are checked and the specification of the path controller with regard to Q'_{m_i} (i.e., $\mathbf{K}_i(Q'_{m_i})$) is computed (Algorithm 1). In the following, the emptiness of $\mathbf{K}_i$ is checked (block 12). If it is empty, the flow comes back to block 11, checks for changes in $D_{i\downarrow}$, and recomputes $\mathbf{K}_i$, and this cycle continues until an adjacent state is freed. If the path controller is not empty (block 12), the path and coordination controllers are computed (block 13); and the disablings $D_{i\downarrow}$ are checked (block 14).

At this point, the momentary blocking is checked again at block 15 to assure the collision-free control action. If there is no momentary blocking, it means there is a path available, then the flow goes to block 16 to finally update $D_{i\uparrow}$ and execute one control action (transition). After this, the specification and the path controller are recalculated normally with the original (desired) destination Q_{m_i} (block 3). If there is a momentary blocking at block 15, it means there is a change since the execution of block 11 and there is no path available anymore; then, the flow goes back to block 11.

5.2.2 Path Controller Specification

The specification for the path controller ($\mathbf{K}_i$) is computed as follows. First, the enabled event set is calculated:

$$\Sigma'_i = \Sigma_i \setminus D_{i\downarrow}, \tag{5.3}$$

where $D_{i\downarrow}$ is the union of all disabled events received by cav_i from other CAVs, given by:

$$D_{i\downarrow} = R_i\left(\left\{\bigcup_{j=1}^{n} D_{j\uparrow} : j \neq i\right\}\right), \tag{5.4}$$

where $D_{j\uparrow}$ is the event set of disabled events published by cav_j and n is the number of CAVs in the range of communication of cav_i.

The computation of $\mathbf{K}_i$ considers the current state as the initial state. It is shown in Algorithm 1. It receives the root graph RG, disablings for cav_i, current state, and marked states for cav_i as inputs. In the output, it generates the control specification automaton $\mathbf{K}_i$. In lines 1, 2, and 3, the state set, transition function, and event active function of $\mathbf{K}_i$ are defined, respectively, as a relabeling of these functions from RG. In line 4, the alphabet of $\mathbf{K}_i$ is defined as the enabled events for cav_i. In lines 5 to 8, the disabled events are removed from Γ'_i and δ'_i, considering the current state of cav_i. Note that only the transitions with disabled events which are active in the current state of cav_i are removed from $\mathbf{K}_i$. In line 9, an auxiliary automaton is defined; and in line 10, $\mathbf{K}_i$ is computed through the Trim operation of the auxiliary automaton.

Algorithm 1: Computation of $\mathbf{K}_i$.

Input: $RG = (Q,\ \Sigma,\ \delta,\ \Gamma)$, $D_{i\downarrow}$, q_i, Q_{m_i}
Output: $\mathbf{K}_i$

1 $Q'_i := R_i(Q)$;
2 $\delta'_i := R_i(\delta)$;
3 $\Gamma'_i := R_i(\Gamma)$;
4 $\Sigma'_i := R_i(\Sigma) \setminus D_{i\downarrow}$;
5 **for** *each* $d \in D_{i\downarrow} \cap \Gamma'_i(q_i)$ **do**
6 remove d from $\Gamma'_i(q_i)$;
7 $\delta'_i(q_i, d) :=$ not defined;
8 $\mathbf{Y}_i := (Q'_i, \Sigma'_i, \delta'_i, \Gamma'_i, q_i, Q_{m_i})$;
9 $\mathbf{K}_i := Trim(\mathbf{Y}_i)$;

5.2.3 Path Controller

The path controller $\mathbf{C}_{\mathrm{pa}_i} : L(\mathbf{G}_i) \rightarrow 2^{\Sigma_i}$, embedded in each CAV, is responsible for controlling the CAV to execute the computed path. The event set of the path con-

troller is equal to the plant's local event set Σ_i. Its input is the set of events generated by the local plant ($\mathbf{G}_i$), and its output is the set of disabled events for $\mathbf{G}_i$, regarding its local path.

The synthesis of $\mathbf{C}_{\mathrm{pa}_i}$ is embedded in the agent, given in Algorithm 2. This algorithm receives automaton $\mathbf{K}_i$ which is the control specification for the path, considering the disablings imposed by other CAVs. The $dijkstra$ function receives the specification and returns an automaton $\mathbf{C}_{\mathrm{pa}_i}$ with all feasible shortest paths, i.e., $L_{\mathbf{C}_{\mathrm{pa}_i}} = L_{sp}$ (Definition 13). The $Dijkstra$ algorithm is well known in the literature and it is not explained here. It is important to note that any optimization algorithm can be used instead of Dijkstra's.

Algorithm 2: Computation of $\mathbf{C}_{\mathrm{pa}_i}$.

Input: $\mathbf{K}_i$
Output: $\mathbf{C}_{\mathrm{pa}_i}$
1 $\mathbf{C}_{\mathrm{pa}_i} := Dijkstra(\mathbf{K}_i)$;

The optimization algorithm used in the path controller guarantees the computation of all shortest paths available in $\mathbf{K}_i$, regarding the marked state (CAVs destination). In the implementation of the path controller, its control actions execute one of the shortest paths, considering the dynamism of the available paths at the moment.

The path controller guarantees the nonblocking of the own CAV's actions w.r.t. its own path, considering it as an independent monolithic system; i.e., if the CAV is alone in the environment, it will never be blocked.

The shortest feasible paths for each state of $\mathbf{C}_{\mathrm{pa}_i}$ are given by Definition 13. The marked language of $\mathbf{C}_{\mathrm{pa}_i}$ can be defined as $L_m(\mathbf{C}_{\mathrm{pa}_i}) = L_{sp}(q_0)$. It means all equally distant shortest path alternatives are considered in the language, resulting in a maximally permissive language.

Definition 13 (Shortest Path Language)**.** Given the specification automaton $\mathbf{K}_i$, and considering a path starting in a state q and finishing in $q_m \in Q_m$, with $|Q_m| = 1$, then, the shortest path language $L_{sp}(q) = \{s_{sp} \,:\, s_{sp} \in L_m(\mathbf{K}_i, q),\ \|s_{sp}\| \leq \|s\|,\ \forall s \in L_m(\mathbf{K}_i, q)\}$. ◊

Given the possibility of recomputation of $\mathbf{C}_{\mathrm{pa}_i}$ at runtime, there is no need to

verify the global blocking (of all CAVs) at the project phase. In other words, there is no need to synthesize a monolithic supervisor to verify the existence of any blocking, whereas, if at least one CAV is momentarily blocked (given that there is no coordinator), the blocked CAVs themselves will reconfigure, independently, their path controllers as shown in Section 5.2.1.

5.2.4 Coordination Controller

The coordination controller $\mathbf{C}_{\mathrm{co}_i} : L(\mathbf{G}_i) \rightarrow 2^{\Sigma}$, embedded in each CAV, rules the interactions between CAVs. Its function is to disable events of other CAVs to avoid conflicts in the dispute over resources. The coordination controller's event set Σ_{co_1} is the union of the vehicle's local event set and the non-labeled event set, i.e., $\Sigma_{\mathrm{co}_1} = \Sigma_i \cup \Sigma$. The events to be disabled are in the Σ event set, which may represent events of any vehicle after the relabeling R_i. This allows the distributed synthesis of the coordination controller, without the need to compose the CAVs' models. Thereby, disabling an event e from Σ is equivalent to disable the events in the set $\{e_j : j \in \mathbb{N} \text{ and } j \neq i\}$, for the agents which are in the communication's reach area.

The computation of $\mathbf{C}_{\mathrm{co}_i}$ is made through Algorithm 3. The root graph and the path controller automaton are received as input for this algorithm. In lines 1 to 4, the set of events to be disabled in each state of $\mathbf{C}_{\mathrm{co}_i}$ is computed. In lines 5 to 9, the states set, events set, transition function, initial state, and destination state of $\mathbf{C}_{\mathrm{co}_i}$ are defined as a copy from $\mathbf{C}_{\mathrm{pa}_i}$. In lines 10 to 13, the set of active events for each state of $\mathbf{C}_{\mathrm{co}_i}$ is defined as the union of the active events of the respective state in $\mathbf{C}_{\mathrm{pa}_i}$ and self-loops of the allowed events in Σ for other CAVs. In line 14, $\mathbf{C}_{\mathrm{co}_i}$ tuples are specified with the variables computed in the algorithms.

Basically, $\mathbf{C}_{\mathrm{co}_i}$ is an automaton obtained from the copy of $\mathbf{C}_{\mathrm{pa}_i}$, adding to each of its states q_i, self-loops with all Σ events that, in RG, do not lead from any state to the state $R^{-1}(q_i)$. Thus, $\mathbf{C}_{\mathrm{co}_i}$ acts to disable, in the other CAVs, the events that would lead them to the current state of cav_i.

The events disabled by cav_i ($D_{i\uparrow}$) are imposed to other CAVs through the output of this controller, which are broadcast through the communication interface. $D_{i\uparrow}$ is calculated as follows:

$$D_{i\uparrow}(q_i) = \{e \in \Sigma : \delta(q', e) = q \text{ is defined, where } q, q' \in Q \text{ and } q = R_i^{-1}(q_i)\}, \quad (5.5)$$

Algorithm 3: Computation of $\mathbf{C}_{\mathrm{co}_i}$.

Input: RG, $\mathbf{C}_{\mathrm{pa}_i}$
Output: $\mathbf{C}_{\mathrm{co}_i}$, $D_{i\uparrow}$

for *each state* $q' \in Q$ **do**
  for *each state* $q \in Q$ **do**
    for *each event* $e \in \Gamma(q')$ *s.t.* $\delta(q', e) = q$ **do**
      $D_{i\uparrow}(R_i(q)) := D_{i\uparrow}(R_i(q)) \cup \{e\}$;
$Q_{\mathrm{co}_i} := Q_{\mathrm{pa}_i}$;
$\Sigma_{\mathrm{co}_i} := \Sigma_i \cup \Sigma$;
$\delta_{\mathrm{co}_i} := \delta_{\mathrm{pa}_i}$;
$q_{0|\mathrm{co}_i} := q_{0|\mathrm{pa}_i}$;
$Q_{m|\mathrm{co}_i} := Q_{m|\mathrm{pa}_i}$;
for *each state* $q_i \in Q_{\mathrm{co}_i}$ **do**
  for *each event* $d \in \Sigma \setminus D_{i\uparrow}(q_i)$ **do**
    $\delta_{\mathrm{co}_i}(q_i, d) := q_i$;
  $\Gamma_{\mathrm{co}_i}(q_i) := \Gamma_{\mathrm{pa}_i}(q_i) \cup \Sigma \setminus D_{i\uparrow}(q_i)$;
$\mathbf{C}_{\mathrm{co}_i} := (Q_{\mathrm{co}_i}, \Sigma_{\mathrm{co}_i}, \delta_{\mathrm{co}_i}, \Gamma_{\mathrm{co}_i}, q_{0|\mathrm{co}_i}, Q_{m|\mathrm{co}_i})$;

in which q_i is the current state of cav_i. It is important to note that $e,\ q',\ q,\ \delta, Q$ and Σ are from RG. Another observation is that $D_{i\uparrow}$ is mapped for states in Q_i, however $D_{i\uparrow} \subset \Sigma$.

Any other vehicle may receive the command to disable some events, but only CAVs in a potential collision are affected. Thus, the path is private to each CAV. Also, the current position and the actions of a CAV are not informed to other CAVs. This characteristic is important considering that malicious CAVs (or other malicious agents) could take advantage of the information of other CAVs' paths.

5.2.5 Conditions for the Solution

Considering the problem statement and the assumptions in Section 1.4, we present how the statements were fulfilled and the conditions for it.

Statement (S1) is fulfilled by disabling other CAVs events, inhibiting undesired movements, and assuring a collision-free behavior. In practice, this is supported by assumption (A7).

Regarding statement (S2), there is a maximum number of CAVs concerning the size of the map (number of states in the model), in which it is possible to find a solution, which is given by Theorem 1.

Theorem 1 (Number of CAVs)**.** *Consider: (i) a circular accessible automaton* G *which*

represents a map for multiple CAVs; (ii) a system with n CAVs, with $\mathbf{G}_i$ in closed-loop with the controllers $\mathbf{C}_{\mathrm{pa}_i}$ and $\mathbf{C}_{\mathrm{co}_i}$ (calculated as in Algorithm 2 and 3, respectively) with the structure shown in Figure 20 under the logic of the flowchart in Figure 21. Then, statement (S2) will hold under the assumptions (A1) to (A8) iff the maximum number of CAVs in the system is smaller than the number of states in $\mathbf{G}$, i.e., $n_{max} < |Q|$.

Proof. It is trivial that if $n_{max} = |Q|$ all CAVs would be blocked and (S2) would not hold. Thus, it is necessary that $n_{max} < |Q|$ for (S2) to be satisfied. It remains then to prove sufficiency. Note that if $n_{max} < |Q|$ then there will always be at least one free state. Since that $\mathbf{G}$ is circularly accessible, if there exists **one free state**, and one CAV moves to this free state, it would free-up one state which can be occupied by another CAV, and so on. As the CAVs recalculate their routes using free states, all the surrounding CAVs would dispute this free state, which prevents the same vehicle from going back and forth to this free state. Thus, CAVs may move at least one state at a time, as long as there exists at least **one free state**. Furthermore, since automaton $\mathbf{G}$ is circularly accessible, all states are reachable for any CAV in this situation of moving one state at a time. As one CAV reaches its marked state, a different marked state is attributed to it, forcing it to move, and therefore all CAVs eventually reach their marked state. In the case of all marked states being occupied by other CAVs, the execution of the flowchart in Figure 21 forces the CAV to move to a free state, avoiding the blocking of the system. □

In other words, as long as there is a free state, there is a path that leads CAVs to their destination states. However, a multi-CAV system with the maximum number of CAVs does not have an efficient solution. To obtain an efficient solution, it is adequate that the number of CAVs be much smaller than the number of states, i.e., $n \ll |Q|$, which is a realistic scenario.

Statement (S3) is achieved through an optimization algorithm in the calculation of $\mathbf{C}_{\mathrm{pa}_i}$. Statement (S4) is achieved by the sharing of disablings instead of its path. Statement (S5) is achieved as the models of the whole system are not composed; assumption (A4) supports this achievement.

Statements (S6) and (S7) are achieved by the generalization of the events disabling. In other words, the only communication between CAVs is broadcasting the disablings. In other words, it does not matter: from which CAV and from how many

CAVs the disablings are coming; nor for which CAV and for how many CAVS the disablings are sent.

5.3 ABSTRACTING THE SPEED OF CAVS IN IMPLEMENTATION

Discrete path planning is a well-known method in the literature (MAHULEA; KLOETZER; GONZÁLEZ, 2020). Generally discrete paths are represented by oriented graphs, or automata as in this work. One of the main advantages of the representation as an oriented graph is that it allows the use of optimization algorithms (e.g. Dijkstra algorithm) to find the shortest path between two nodes.

In theory, the transition of the CAVs between states is considered as instantaneous, and the size of the robot is considered to be a point in space. However, in practice the transition is not instantaneous and the size of the robot is not a point. This leads to an implementation problem, which could cause collisions between CAVs, if not treated. Thus, the size and speed of the CAV must be considered in order to avoid collisions in CAVs systems.

The aim of discrete path planning is to abstract the physics of the vehicle, such as size, speed and acceleration. So, if the transition between two states A and B is considered as instantaneous at the time of the decision, the $\mathbf{C}_{\mathrm{co}_i}$ will be disabling the events for B state, but physically, the CAV is still in the state A.

If the transition between these states A and B is considered as instantaneous at the time that the CAV is arriving at state B, the $\mathbf{C}_{\mathrm{co}_i}$ will be disabling the events for A state, but physically, the CAV will be very close to state B. In both cases collision could occur as $\mathbf{C}_{\mathrm{co}_i}$ would be disabling the “wrong” events.

One solution is to calculate the future position in function of speed and size in order to determine which CAV has the preference of occupying a state. However, this solution requires a continuous state-space modeling with a considerable computational effort which depends on the speed sensor. In other words, this would be a whole control technique, and, if combined with the proposed architecture would turn the solution into a complex task.

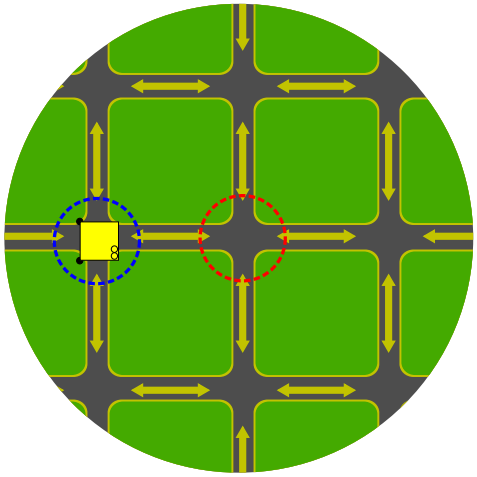
Thus, we have developed a very simple technique to abstract the speed of the CAV in the implementation. Basically, the transition will occur in two steps, a logical transition and a physical transition. The logical transition occurs as in theory: instantaneous. And the physical transition depends on the physical position of the CAV.

To better understand, we can visualize two identical control structures implemented, which represents the “logical” and “physical” models. The states in the logical automata represent the logical position of the CAV, and the state in the “physical” automata represent the physical position of the CAV. This is an artifice to understand, in fact only one control structure is implemented.

The logical transition follows the rules of the control architecture. In the case of a CAV going from state A to B, the transition is performed as soon as it is enabled in the logic.

For the physical transition, a safe radius of the size of the CAV is traced around the physical position of the state. This radius has to be big enough to assure the physical position of the state is a collision-free area at the moment. In Figure 22 the illustration of a CAV moving from left to right is shown. The real state is denoted by the blue color and the logical state is denoted by the red color. Note that the CAV is still in the safe radius, and therefore the real transition has not been performed yet.

Figure 22 – Illustration of real (blue) and logical (red) states.



Source: Silva, Leal and Sebem (2021).

One observation, in a transition from A to B, is that while the CAV is occupying the radius from the A state, in fact, the CAV will be occupying both the physical state A and the logical state B. This way, the speed of the CAV can be safely abstracted from the model.

Hence, in the condition that the real state is equal to the logical state, it means that the CAV have physically arrived at the destination state. Considering a path $A \rightarrow B \rightarrow C$, one interesting observation is that the logical transition from state B to C, for

example, will be only enabled after the physical transition from A to B is finished. In other words, the transition $B \rightarrow C$ is enabled after the real state is equal to the logical state: B. For this purpose, comparing the real and logical states of a CAV is very useful, because if both are equal, it means the last transition has been concluded.

6 SIMULATIONS AND EXPERIMENTS OF THE PROPOSED ARCHITECTURE

In this chapter, we present examples, simulations and experiments which were designed with the aim to validate the control architecture, as well as to stress it to extreme conditions.

In the context of this thesis, we have adapted a simulation environment the *Robotarium*[1] simulator (WILSON et al., 2020), which provides a 2D graphical video of the simulations. For the experimental results, we have built an infrastructure with 5 Lego Mindstorm robots, in the line-following configuration and a road structure with 5×6 intersections.

6.1 EXAMPLES

In this section, we provide two examples to illustrate the proposed methodology in a didactic manner. The examples are presented step by step, and in each step, the situation is referred to the proposed solution. In both examples, we use the same setup, with different conditions.

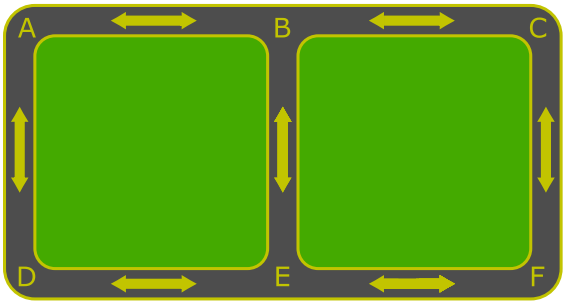
The examples are based on the transit of CAVs in a given map, such as depicted in Fig. 23a. We specifically model a map with “one lane & two ways” roads, as it is the worst case in path dispute between CAVs. Path planning in traditional “two lanes & two ways” roads would be easier to solve.

The system for both examples is formed by two CAVs, named cav_1 e cav_2, which move on the map. The map is shown in Fig. 23a and its model with 6 states (A, B, C, D, E, F) is shown in Fig. 23b. The roads are composed of one lane which allows traveling in both ways, however, only one vehicle must occupy the road at a time. In order to simplify the computation of the shortest path, the distances between adjacent intersections are equal. To improve the solution for arbitrary distances, a weighted automata should be used for map modeling. In these examples, the communication range is considered to cover at least two states ahead in any direction.

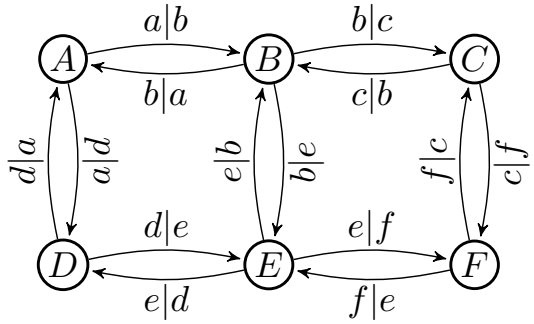
The root graph model of this map is given in Fig. 23b, where each state represents an intersection in space. Thus, if two CAVs cannot occupy the same physical space, consequently they cannot occupy the same state, at the same time. The occu-

[1] <https://www.robotarium.gatech.edu>

Figure 23 – Map for the examples: (a) design and (b) Root Graph model.



(a)



(b)

Source: designed by the author (2022).

pation of the same state by two different CAVs represents a possible collision between them. The transitions' names indicate the states of origin and destination. For example, $a|b$ indicates the transition from state A to state B.

CAVs may move both ways between states, however, only one CAV may occupy the road at a time. Thus, if a vehicle occupies the state A the events $b|a$ e $d|a$ must be disabled for all CAVs in the communication range to avoid two vehicles occupying the same state and consequently avoid a possible collision.

6.1.1 Example 1

Consider that the CAVs are fully automated and have a destination to reach from their starting point. Initially, the cav_1 is in the A_1 position and its destination is to reach the C_1 position. For cav_2, the initial position is C_2 and its destination is to reach the A_2 position.

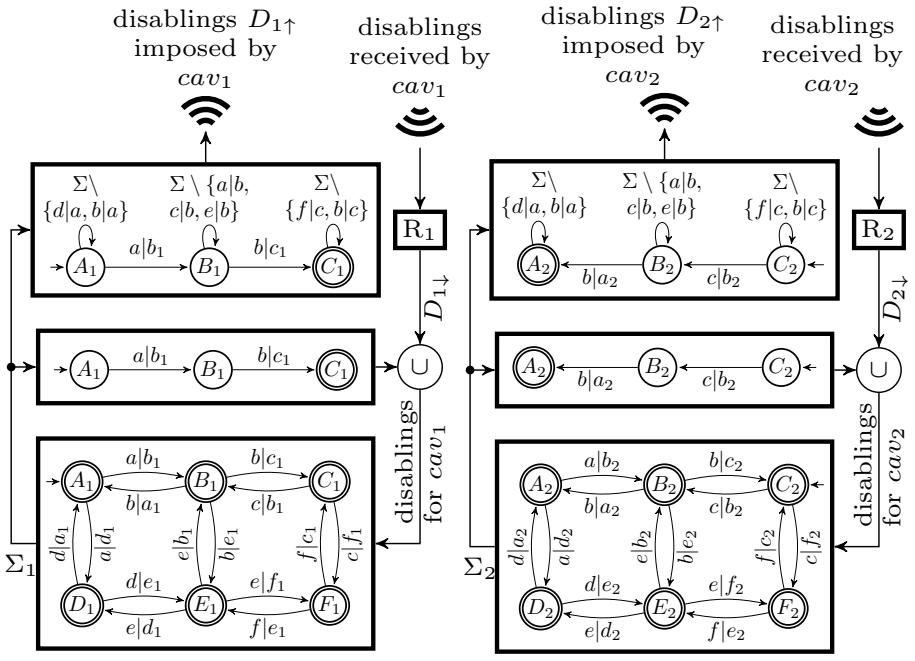
The explanation of the examples is based on the flowchart in Fig. 21. Note that the operation of both CAVs is not synchronous, but we group some initial steps in the flowchart for both CAVs to simplify the explanation. Initially, both CAVs receive their destinations at block 2. At block 3 they check by disablings from other CAVs, however,

none of them has computed their coordination controller, and $D_{1\downarrow} = D_{2\downarrow} = \varnothing$. Then, they compute $\mathbf{K}_1$ and $\mathbf{K}_2$, which in this case is equal to $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. Thus $\mathbf{K}_i$ is not empty for both CAVs, at block 4.

Suppose cav_1 finishes its computation of $\mathbf{C}_{\mathrm{pa}_1}$ and $\mathbf{C}_{\mathrm{co}_1}$ at block 5 first, in which $\mathbf{C}_{\mathrm{pa}_1}$ is such that $\mathcal{L}_m(\mathbf{C}_{\mathrm{pa}_1}) = \{a|b_1b|c_1\}$. At this point, cav_1 begins to broadcast its disablings for other CAVs, $D_{1\uparrow} = \{d|a, b|a\}$, in the cyclic interruption at the bottom of the flowchart. In block 6, cav_1 checks for the received disablings, which is empty while cav_2 has not finished its computation at block 5.

In sequence, cav_2 finishes computing $\mathbf{C}_{\mathrm{pa}_2}$ and $\mathbf{C}_{\mathrm{co}_2}$ at block 5, in which $\mathbf{C}_{\mathrm{pa}_2}$ is such that $\mathcal{L}_m(\mathbf{C}_{\mathrm{pa}_2}) = \{c|b_2b|a_2\}$. The control structure for both CAVs considering this situation is presented in Fig. 24. Note that G_1 and G_2 are local plants, in the inferior blocks in Fig. 24, which represent the map with all possible paths for the CAVs.

Figure 24 – Initial models for Example 1.



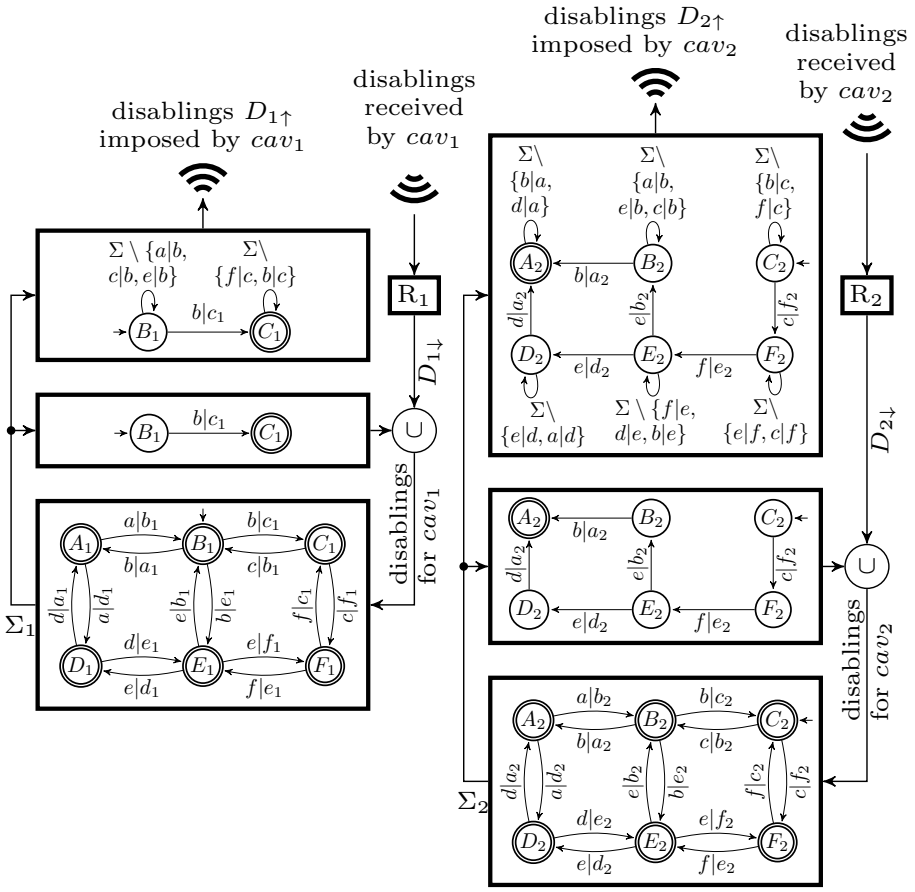
Source: designed by the author (2022).

Continuing the analysis, at block 7, cav_1 checks for momentary blocking and the result is *no*. At block 8, cav_1 updates $D_{1\uparrow} = \{a|b, c|b, e|b\}$, and executes one control action, which is the execution of the event $a|b_1$.

While cav_1 is moving from A_1 to B_1, cav_2 checks for the received disabling which is $D_{2\downarrow} = \{a|b_2, c|b_2, e|b_2\}$, at block 6. And at block 7, cav_2 is momentary blocked (equa-

tion 5.2), as its only active event $c|b_2$ at the current state of $\mathbf{C}_{\mathrm{pa}_2}$ is disabled by cav_1. At block 3, cav_2 checks for changes in $D_{2\downarrow}$, which is still $\{a|b_2, c|b_2, e|b_2\}$; then it computes $\mathbf{K}_2$ considering $c|b_2$ as disabled. Note that only $\delta(C_2, c|b_2) = B_2$ is removed from $\mathbf{K}_2$, $\delta(E_2, e|b_2) = B_2$ and $\delta(A_2, a|b_2) = B_2$ are not removed as they are not in the current state (B_2) of $\mathbf{K}_2$ (see Algorithm 1). At block 4, $\mathbf{K}_2$ is not empty. At block 5 cav_2 recomputes its controller. Considering cav_1 have reached state B_1, the situation of the CAVs is presented in Figure 25.

Figure 25 – Recomputed models after reconfiguration in Example 1.



Source: designed by the author (2022).

As soon as cav_2 moves from state C_2 to state F_2, cav_1 is free to reach its final destination at state C_1. These paths of Figure 25 will hold until CAVs finish their paths, as they do not cross each other in this example.

The final path for cav_2 is longer than its initial path, however, considering both vehicles as part of a global system, on average, this solution is the shortest feasi-

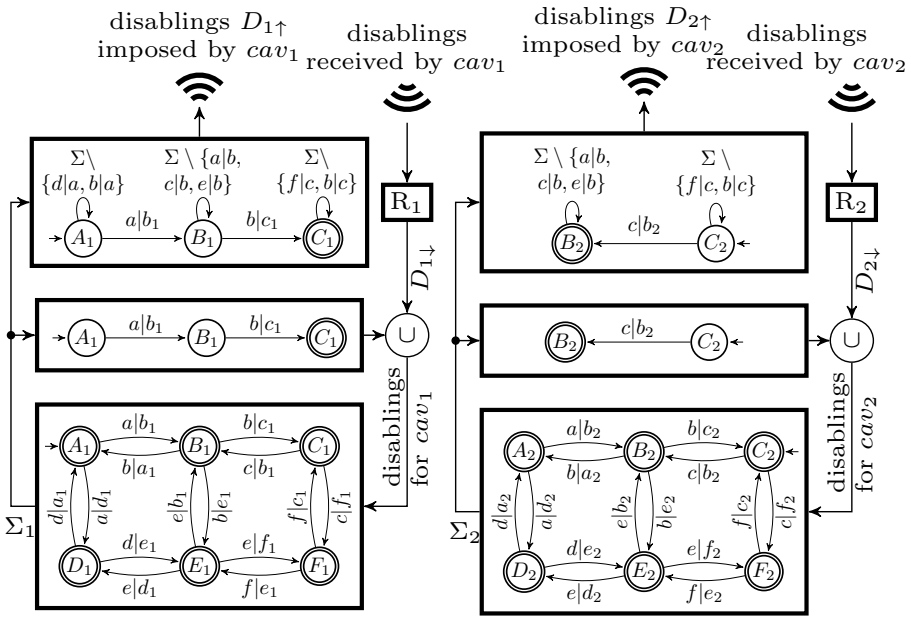
ble path for both CAVs. Note that the final path for cav_2 has two shortest paths in $L_{sp}(E_2) = L_m(\mathbf{C}_{\mathrm{pa}_2}, E_2)$. This is the maximally permissive language for cav_2 at the current situation.

6.1.2 Example 2

The objective of example 2 is to exemplify the solution when the computation of $\mathbf{K}_i$ is empty while solving a momentary blocking. This example is similar to Example 1, however, now the destination of the cav_2 is B_2 position.

The first steps of this example can be considered the same as example 1 until the models for the CAVs are computed at block 5. In this case, the models for the controllers are represented in Figure 26.

Figure 26 – Initial models for Example 2.



Source: designed by the author (2022).

Again, suppose cav_1 finishes computing $\mathbf{C}_{\mathrm{pa}_1}$ and $\mathbf{C}_{\mathrm{co}_1}$ at block 5 first, in which $\mathbf{C}_{\mathrm{pa}_1}$ is such that $\mathcal{L}_m(\mathbf{C}_{\mathrm{pa}_1}) = \{a|b_1b|c_1\}$. Then, cav_1 begins to broadcast its disablings for other CAVs, $D_{1\uparrow} = \{d|a, b|a\}$, in the cyclic interruption. In block 6, cav_1 checks for the received disablings, which is empty while cav_2 have not finished its computation at block 5.

In sequence cav_2 finishes computing $\mathbf{C}_{\mathrm{pa}_2}$ and $\mathbf{C}_{\mathrm{co}_2}$ at block 5, in which $\mathbf{C}_{\mathrm{pa}_2}$ is

such that $\mathcal{L}_m(\mathbf{C}_{\mathrm{pa}_2}) = \{c|b_2\}$ and $D_{2\uparrow} = \{b|c, f|c\}$.

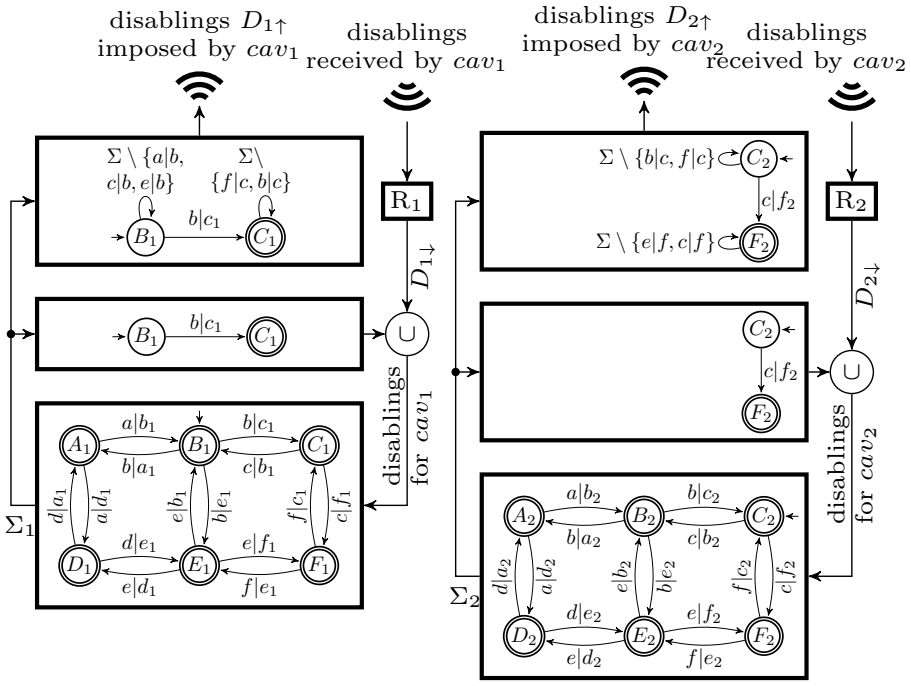
Continuing the analysis, at block 7, cav_1 checks for momentary blocking and the result is *no*. At block 8, cav_1 updates $D_{1\uparrow} = \{a|b, c|b, e|b\}$, and executes one control action, which is the execution of the event $a|b_1$.

While cav_1 is moving from A_1 to B_1, cav_2 checks for the received disabling which is $D_{2\downarrow} = \{a|b_2, c|b_2, e|b_2\}$, at block 6. And at block 7, cav_2 is momentary blocked (5.2), as its only active event $c|b_2$ at the current state of $\mathbf{C}_{\mathrm{pa}_2}$ is disabled by cav_1. At block 3, cav_2 checks for changes in $D_{2\downarrow}$, which is still $\{a|b_2, c|b_2, e|b_2\}$; then it computes $\mathbf{K}_2$ considering $c|b_2$ as disabled. In this case, at block 4, $\mathbf{K}_2$ is empty because cav_1 is occupying the only possible state (B) in the path of cav_2, i.e., the next state in the path which is cav_2 marked state.

At block 10 cav_2 marks all states within one transition ahead, which is $Q'_{m_2} = \{F_2\}$. At block 11, cav_2 checks the received disablings which have not changed; and computes $\mathbf{K}_2(Q'_{m_2})$. Then, at block 12, $\mathbf{K}_2$ is not empty, which leads to compute $\mathbf{C}_{\mathrm{pa}_2}$ and $\mathbf{C}_{\mathrm{co}_2}$ (block 13). The result is such that $\mathcal{L}_m(\mathbf{C}_{\mathrm{pa}_2}) = \{c|f_2\}$.

Considering cav_1 has reached state B_1, the situation of the CAVs is presented in Figure 27.

Figure 27 – Recomputed models after a momentary blocking in Example 2.



Source: designed by the author (2022).

The next step for cav_2 is to check again the received disablings (block 14) and the momentary blocking (block 15). Then, cav_2 executes a transition through event $c|f_2$ (block 16), reaching state F_2. At this stage, the momentary blocking is solved. This procedure allows cav_1 to transit to state C_1 (block 8) to reach its destination, releasing state B_2 for cav_2. Following, cav_2 recomputes $\mathbf{K}_2(Q_{m_2})$ (original destination) at block 3, and its controllers (blocks 4 and 5) to make it possible to reach state B_2 through the path $f|e_2 \rightarrow e|b_2$. Similar to example 1, the final path for cav_2 is longer than its initial path, however, considering both vehicles as part of a global system, on average, this solution is the shortest path for both CAVs.

6.2 SIMULATION RESULTS

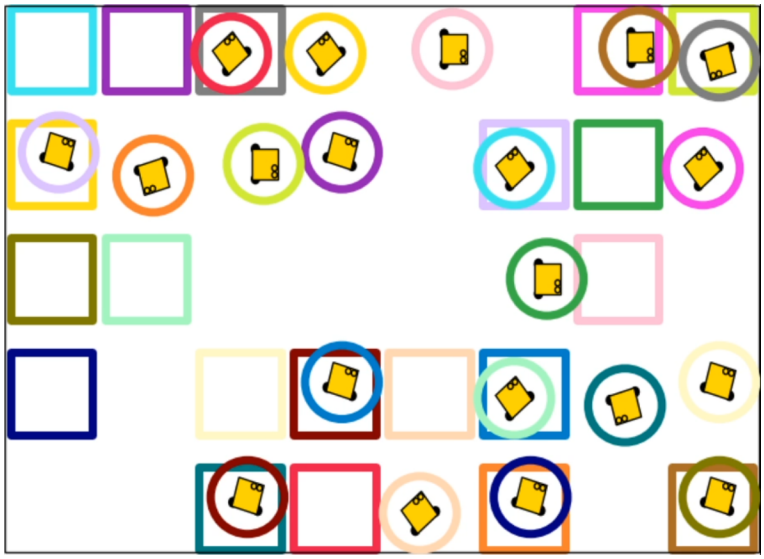
In this section we present the simulation results in a objective manner. All codes developed within this work are available on GitHub[2]. In addition, we provide videos of the simulations on Robotarium for examples 1 and 2[3]; and for simulation 1[4].

6.2.1 Simulation 1

Simulation 1 is designed to evaluate the reconfiguration and performance of the proposed architecture for multiple CAVs. The map for this example has the same characteristics as the map in the previous examples, however, its size is greater with an array of 5×8 states in the automaton model of RG. To evaluate the absence of collisions or blocking, a dense quantity of 20 CAVs is placed on this map, with random initial positions, and random destinations. The disposal of the CAVs in the simulation environment can be seen in Figure 28. Each CAV has a colored circle, and a square with the same color represents their respective destinations. As soon as a CAV reaches its destination state, a new destination is randomly attributed to it. Results show that CAVs have reached their destinations as expected. No collisions, no deadlocks nor livelocks were observed. All momentary blockings were solved. Note that this is a dense setup which makes it difficult for CAVs to directly reach their destinations.

[2] <https://github.com/GASR-UDESC/Control_of_CAVs/tree/CEP_simulations>
[3] <https://youtu.be/NxbA7uSsU28>
[4] <https://youtu.be/rUCDD7MblUg>

Figure 28 – Environment setup for Simulation 1.



Source: designed by the author (2022).

6.2.2 Simulation 2

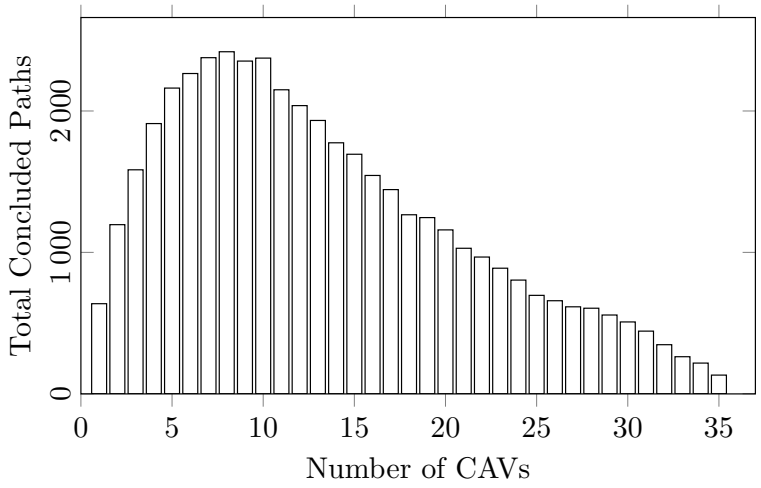
In this simulation, we use a square map modeled with 6×6 states, with a total of 36 states. The objective is to verify the efficiency of the proposed solution w.r.t. the number of CAVs on the map. We simulate the situations with 1, 2, 3, ..., 36 CAVs on the map, i.e. 36 steps, all of them reaching random destinations. Each step was ran for 1.5 hours, totalizing 54h of simulation. It is not recorded on video due to its long duration.

The first analysis consists in the total completed paths, accumulated of all CAVs. In Figure 29, a plot of the total completed paths vs. the number of CAVs is disposed. For example, the first bar of the graph represents the first step of the simulation, in which one CAV ran alone in the environment, with random paths, during 1.5 hours. In other words, this CAV have completed 637 random paths in 1.5 hours. In the second step, two CAVs have shared the environment, and they both have completed 1196 random paths in 1.5 hours. It can be observed that the greatest number of completed paths is with 8 CAVs, which we can say is the most efficient condition considering the system as a whole. It indicates that the ideal occupancy rate of the environment is about 22,2%. This means that 8 CAVs have shared the environment and completed 2419 random paths in 1.5 hours.

The second analysis consists in the amount of completed paths by CAV. In

Figure 30, a plot of the average completed paths by CAV vs. the number of CAVs is disposed. In summary, this plot is the result of dividing the data of the plot in Figure 29 by the number of CAVs. For example, this means that in the second step, each CAV completed 598 random paths. In other words, from one CAV running alone to two CAVs competing for road occupation, the amount of completed paths has dropped from 637 to 598. This makes sense, because when competing, CAVs often have to change their paths. As expected, the average completed paths decays as the number of CAVs grows.

Figure 29 – The total completed paths of all CAVs vs. the number of CAVs.

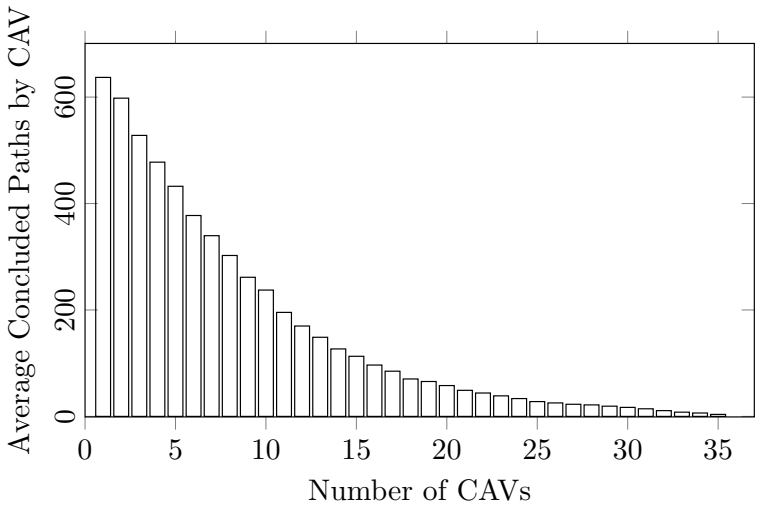


Source: designed by the author (2022).

The third analysis consists in the average distance traveled by path. In Figure 31 the average distance traveled by path, in meters, is plotted with regard to the number of CAVs. In the first step, one CAV alone travels 1.26 m, and this value can be considered as the basis of the distance for random paths for this map configuration. In the second step, with two CAVs competing, in some cases CAVs have to change their paths to avoid collisions, because of this, their traveled distance grows to 1.31 m by path. As expected, the average distance traveled by path grows as the number of CAVs grows.

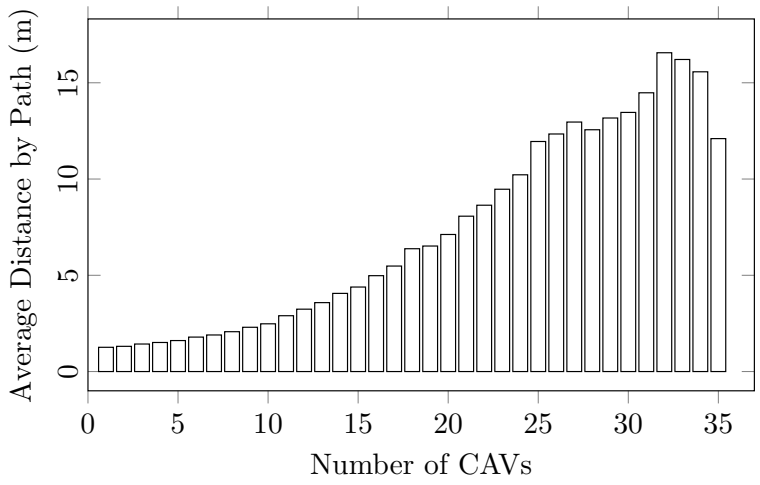
This results make sense with a real application of urban traffic. In other words, with more cars on the road, more difficult it gets to complete a path and longer it gets. However, there is an acceptable amount of cars which makes it possible to share the environment without affecting much of the performance of individual cars, which, in this

Figure 30 – The average completed paths by each CAV vs. the number of CAVs.



Source: designed by the author (2022).

Figure 31 – The average distance traveled by CAV by path vs. the number of CAVs.



Source: designed by the author (2022).

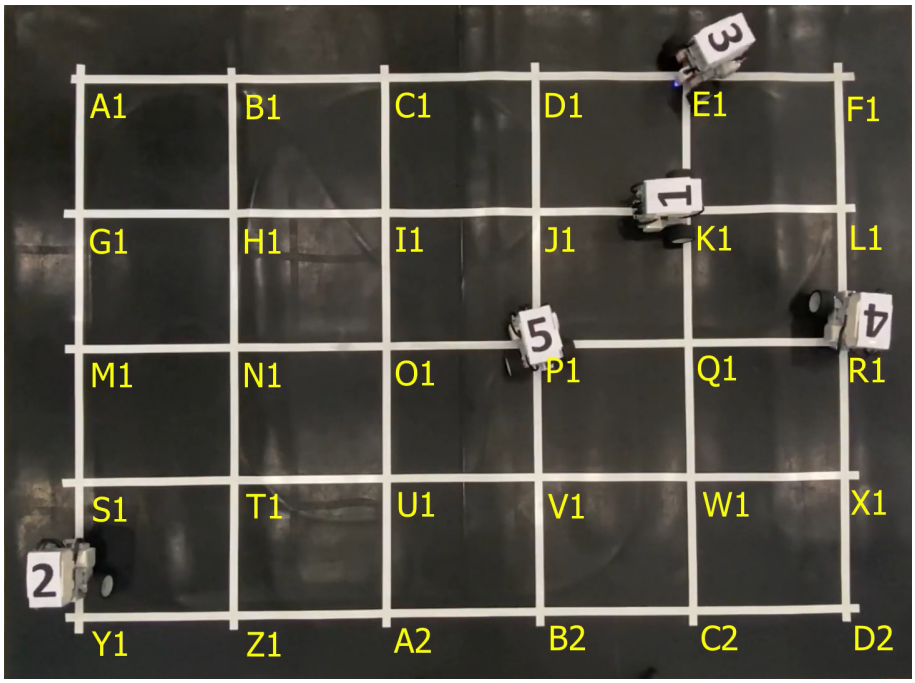
case would be 8 CAVs. Comparing the results from one CAV alone to 8 CAVs in the environment, the path length goes up 64%, and the amount of paths concluded in 1.5 h goes down by 52%, which is acceptable considering a real urban traffic.

6.3 EXPERIMENTAL RESULTS

In this section, we present two experiments with the Lego Mindstorm robots to validate the proposed architecture. The robots are set up in a line-follower configura-

tion to act as CAVs. The road structure is formed by 5×6 intersections, with uniform distances between intersections, as shown in Fig 32. All codes for the experiments are available in GitHub[5].

Figure 32 – Experimental infrastructure with Lego Mindstorm robots.



Source: Teles, Leal and Sebem (2021).

The communication between CAVs is made through wi-fi with the MQQT protocol. In this setup, we have used a computer to simulate the passenger/operator on the task of choosing destinations. It is important to note that the computer does not act as a coordinator for the control, it is just an easy way to send commands (destinations) to many CAVs. Also, in the second experiment random destinations are determined by the computer.

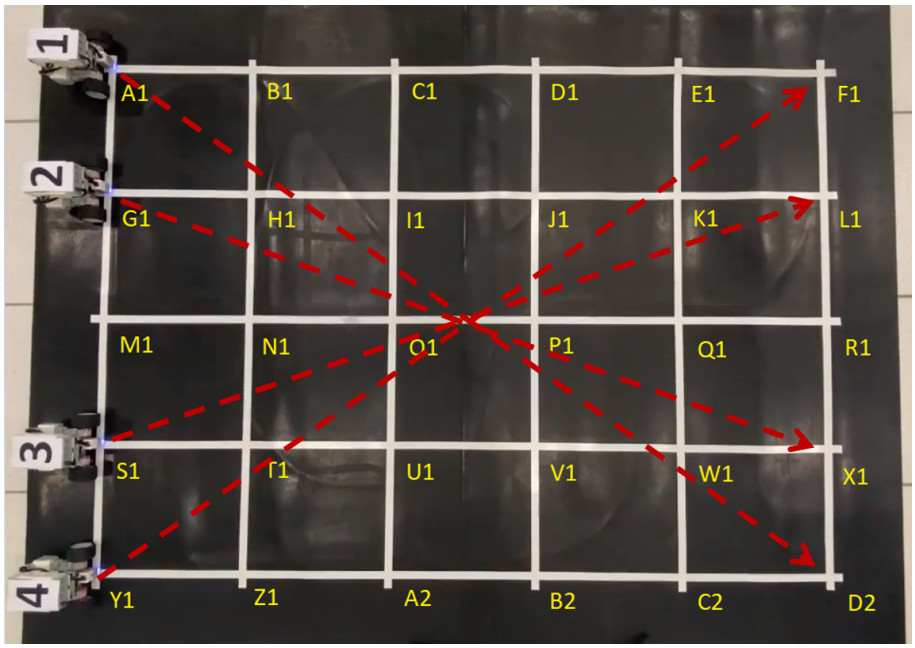
6.3.1 Experiment 1

This experiment consists of 4 CAVs that are positioned respectively at positions A1, G1, S1, and Y1; with destinations respectively at positions D2, X1, L1, F1. In Figure 33, the environment is shown with the CAVs placed at their initial position, and their destinations are denoted by red arrows. This setup forces them to cross paths with each other, which allows us to verify the performance of avoiding collisions with

[5]<https://github.com/GASR-UDESC/Practical_Control_of_CAVs>

the proposed architecture.

Figure 33 – Setup for experiment 1.



Source: Teles, Leal and Sebem (2021).

A video of Experiment 1 is available online[6]. In the video, it can be observed the fulfillment of the statements S1, S2, S6, and S7 (Section 1.4). The other statements are intrinsic to the architecture and cannot be observed in the video.

In Table 11, we provide a comparison of the ideal minimum path distance for each CAV and the executed path. Both CAVs 2 and 3 execute the ideal path. Both CAVs 1 and 4 execute 2 extra transitions in relation to the ideal path, which represents an extra traveling distance of 22,2% for each. The paths of CAVs 1 and 4 have the same length, and it was a coincidence that they both executed 2 extra transitions, resulting in the same extra percentage traveled. The extra distance is traveled because the CAVs have swerved an occupied state which was in the shortest path. On average, each CAV traveled an extra distance of 12,5% in relation to the ideal minimum paths.

6.3.2 Experiment 2

This experiment consists of 5 CAVs that are respectively positioned at A1, F1, Y1, D2, and O1. Each CAV executes three random paths and then stays parked at the

[6] <https://youtu.be/stxbJfdZRmA>

Table 11 – Comparison of the ideal distance and the executed distance in Experiment 1.

CAV	Ideal Distance (m)	Executed Distance (m)	$\triangle$ (m)
1	3.6	4.4	0.8
2	2.8	2.8	0
3	2.8	2.8	0
4	3.6	4.4	0.8
Total	12.8	14.4	1.6

Source: designed by the author (2022).

last destination. We designed this random experiment to be similar to real car traffic.

A video of Experiment 2 is available online[7]. It can be observed that even with random paths the statements S1, S2, S6, and S7 are fulfilled. As it was mentioned before, the other statements are intrinsic to the architecture and cannot be observed in the video.

In Table 12, we provide a comparison of the ideal minimum path distance for each CAV and the executed paths in Experiment 2. It can be seen that only CAV 3, in one path, has executed a path longer than the ideal minimum path. Considering the total traveled distance, on average, each CAV would have traveled an extra distance of 8% in relation to the ideal minimum paths. It can be concluded that Experiment 1 is a specific case because the CAVs are forced to cross paths with each other, and Experiment 2 represents the realistic traffic of CAVs.

[7] <https://youtu.be/zpc1ilvCxAg>

Table 12 – Comparison of the ideal distance and the executed distance in Experiment 2.

CAV	Path	Ideal Distance (m)	Executed Distance (m)	$\triangle$ (m)
1	A1 H1	0.8	0.8	0
1	H1 I1	0.4	0.4	0
1	I1 K1	0.8	0.8	0
2	F1 A1	2	2	0
2	A1 Y1	1.6	1.6	0
2	Y1 U1	1.2	1.2	0
3	Y1 D1	2.8	2.8	0
3	D1 E1	0.4	0.4	0
3	E1 C2	1.6	3.2	1.6
4	D2 A2	1.2	1.2	0
4	A2 V1	0.8	0.8	0
4	V1 F1	2	2	0
5	O1 T1	0.8	0.8	0
5	T1 J1	1.6	1.6	0
5	J1 Z1	2	2	0
	Total	20	21.6	1.6

Source: designed by the author (2022).

7 DISCUSSION

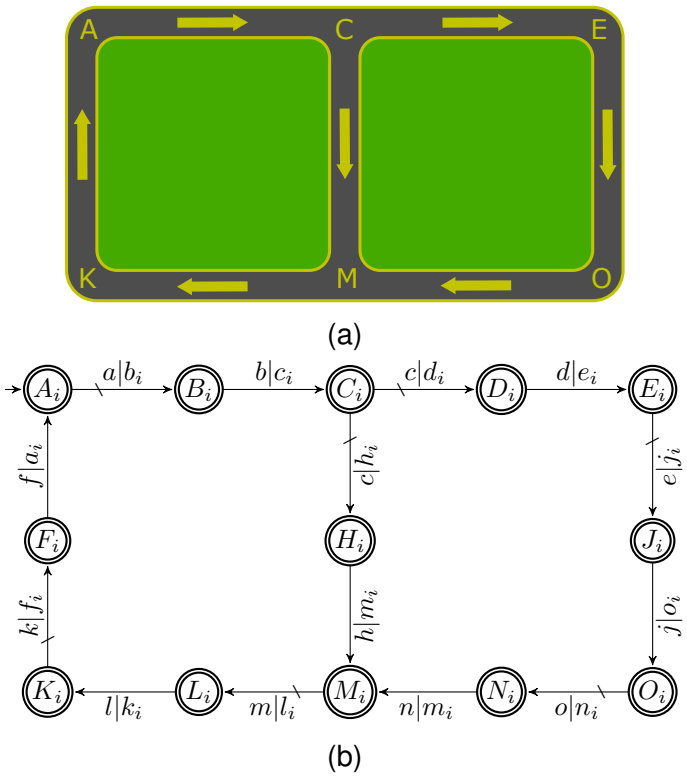
In this chapter, a discussion is made regarding some possibilities which can be explored with the proposed architecture.

7.1 CONSIDERING UNCONTROLLABLE BEHAVIOR

In our studies, we have not visualized the need for modeling uncontrolled behavior of CAVs. Anyway, we present a model with an uncontrolled behavior, and how the proposed control architecture would have to be changed to remain valid. Also, we present what would be a *bad state* in this case, and the equivalent calculation of the supervisor.

There are various manners of abstracting a DES in an automaton model. Considering that events are conceptually instantaneous, one manner of modeling intersections is to separate the events of departing of one intersection and arriving at another intersection. In this case, the arrival at one intersection can be modeled as an uncontrollable event, as shown in Figure 34.

Figure 34 – Map for the examples: (a) design and (b) Root Graph model.



Fonte: Elaborado pelo autor (2022).

Consider a system with two CAVs, if cav_1 sends $D_{1\uparrow} = \{h|m,\ n|m\}$, then cav_2 will have to include the $SupC$ operation in the calculus of $\mathbf{C}_{\mathrm{pa}_2}$, as shown in Algorithm 4.

Algorithm 4: Computation of $\mathbf{C}_{\mathrm{pa}_i}$ with uncontrollable events.

Input: $\mathbf{K}_i$
Output: $\mathbf{C}_{\mathrm{pa}_i}$
1 $\mathbf{S}_{\mathrm{pa}_i} := supC(\mathbf{G}_i, \mathbf{K}_i)$;
2 $\mathbf{C}_{\mathrm{pa}_i} := Dijkstra(\mathbf{S}_{\mathrm{pa}_i})$;

Another interesting idea is to model pedestrian behavior, such as a cross signal, as an uncontrollable event.

7.2 CONSIDERING RESTRICTED MODELS FOR $\mathbf{G}_I$

In a very large-scale map, it is desirable that each CAV uses a model for $\mathbf{G}_i$ which covers the surroundings of the desired path. In other words, there is no need to load an automaton $\mathbf{G}_i$ which models the entire map. For example, if a CAV is moving within a city, there is no need to load an automaton model of the entire state.

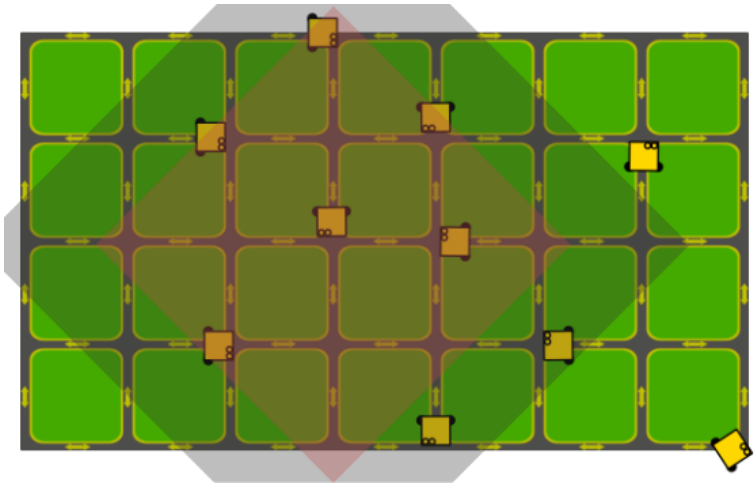
It is important to note that the proposed architecture accepts a restricted model of $\mathbf{G}_i$ and would still work without any loss.

7.3 RESERVING A PATH FOR SPECIAL VEHICLES

In a real world scenario, it is desirable to have special vehicles which have priority in traffic, such as ambulances and firetrucks. In our proposal we have considered the same priority for all CAVs, however, through a simple modification in Algorithm 3, it is possible to reserve a path for special vehicles. Basically, the idea is to disable events for the desirable intersections, for example, the special vehicle could reserve three intersections ahead allowing it to travel faster.

The only restriction of implementing this technique is that the communication range must be greater than the reserving range. In Figure 35, an illustration of the ranges is made for the CAV in the 3 line and fourth column. The communication range is three states ahead, marked by a gray area; and the reserving range is two states ahead, marked by a red area. The range is considered as square to facilitate the understanding.

Figure 35 – Illustration of the communication (gray) and reserving (red) ranges.



Source: Silva, Leal and Sebem (2021).

7.4 COMMUNICATION TECHNOLOGIES FOR CAVS

An important matter to discuss is the reliability of the communication to be implemented for the proposed architecture. As it is based on the exchange of events to be disabled, it is required a very reliable communication system, otherwise, collisions could occur. In this sense, the Ultra-Reliable Low Latency Communication (URLLC) over 5G is a feasible alternative, which provides 99.99% of reliability with less than 1 ms latency (ALI et al., 2021).

Lonc and Cincilla (2016) presents the communications standards which have been developed specifically for cooperative intelligent transportation systems. This topic has been studied in a while and, European (ETSI, 2019) & American (NHTSA, 2017) communications standards already have been developed for V2V communication of CAVs. Singh, Nandi and Nandi (2019) developed an extensive survey on vehicular communication technologies and is an important reference on the topic. Khan et al. (2022) provides a specific survey for communication of CAVs.

8 CONCLUSION

In this thesis, a distributed control architecture for CAVs, with the features of scalability and reconfiguration at runtime was presented. This architecture is capable of solving a variety of problems on CAVs control such as intersection management, road merging, dynamic path planning, and roundabouts. The privacy of the CAVs paths, the nonblocking and collision-free behavior are assured by the control architecture. The major advantage of the proposed control architecture is the combination of three desirable features in a simple methodology: scalable number of CAVs, distributed coordination, and reconfigurable path planning. The implementation of the algorithms developed in this thesis is simple, considering the complexity of the problem. Furthermore algorithms are distributed (embedded) in each CAV, which reduce the complexity and processing capacity, comparing to a algorithm in a centralized processor.

The objectives presented in the Chapter 1 were all achieved with the proposed architecture. The scalability is achieved by exploring the similarity of CAVs in the development of an algorithm that works independently of the number of CAVs. The reconfiguration at runtime, which allows the assignment of new tasks, is achieved by an algorithm which recomputes the CAVs models at runtime. And, the distributed control is achieved by developing a control algorithm which can be embedded in the CAVs, allowing the control actions to be decided locally by each vehicle, with no need for a coordinator. Results show that the proposed architecture is reliable and feasible in various situations. Also, results show that the efficiency of the control is consistent with real situations in traffic of vehicles.

The architecture supports other characteristics of real urban/interurban traffic that were not modeled in the examples, such as exclusive lanes (e.g. bus lanes), roundabouts, road merging, priority for specific vehicles (e.g. ambulances), variable speed of CAVs, signalized intersections, and signalized pedestrian crossings. The architecture is adequate for use in real urban/interurban traffic overcoming both problems of reconfigurable path planning and non-signalized intersections, with nonblocking and collision-free behavior. Furthermore, the architecture does not need an infrastructure, which reduces the cost and time of implementation in large urban environments. Another advantage is the privacy of the path for each CAV, which enhances the safety for

the users.

This method can be easily applied to mixed traffic, with CAVs and HDVs. In the HDVs, the path controller is the human driver, then an electronic device could be installed to serve as coordination controller. This device would show the human driver when to stop or go ahead. In this case, the path would be manually chosen by the human driver. And in the CAVs, nothing would change.

The unfolding of this work includes its application to similar multi-agent systems, such as Unmanned Aerial Vehicle (UAV), as well as the improvement of the modeling towards traffic of CAVs. For example, the map in the simulation and experiments could be modeled to represent real urban maps. In other words, the number of road lanes must be increased, and different configurations of lanes, ways, intersections, or roundabouts must be explored. In this work, it is considered that all CAVs consider disablings one state ahead on its path. This aspect may be explored in the sense that some CAVs may have priority over others (e.g. ambulances). The idea would be to reserve many states ahead on its path, through the disablings of events in a sequence of states. This would guarantee the minimum time for the priority CAV to reach its destination. From another perspective, this method could be easily adapted to support map reconfiguration, for example, given a city, the traffic is more intense in one direction in the morning, and at nightfall, the traffic is more intense in the opposite direction. So the map could be reconfigured accordingly in each CAV, i.e., the street direction could be reversed accordingly to the need of the traffic.

8.1 PUBLICATIONS AND SUPERVISIONS

During the period of development of the thesis, the author have written 8 papers, 4 of them are related to this thesis, and 4 are collaborations within the research group in other topics. In total, 5 papers were published in the period of development of this thesis.

8.1.1 Papers in the Context of this Thesis

The work developed in the context of this thesis comprehends the writing of 4 papers:

1. SEBEM, R.; LEAL, A. B. Proposta de arquitetura para controle distribuído de sis-

temas a eventos discretos multiagentes autônomos. In: **Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI)**. Ouro Preto: [s.n.], 2019.

2. SEBEM, R. et al. Architecture for scalable and distributed control of connected and automated vehicles with reconfigurable path planning. (To be submitted).

3. SEBEM, R.; LEAL, A. B.; BERTOL, D. W. Overview of distributed control for connected and automated vehicles. (In preparation).

4. SILVA, M. F. d.; SEBEM, R.; LEAL, A. B.; BERTOL, D. W. Implementation strategies for distributed control of connected and automated vehicles. (In preparation).

The first paper is the original proposal for the architecture, which was developed to achieve the current proposal.

The second paper is part of the Chapter 5, which is the proposal of this thesis. This paper is being finished and will be submitted to a qualified international journal.

The third paper is based on Chapters 4 and 3, in which the methods of control for CAVs in the literature are reviewed. This paper is in phase of writing of the original draft. We intend to submit this paper to a qualified international journal.

The fourth paper is related to the implementation of the architecture, Section 5.3 and Chapter 6 are related with this paper. This paper is in phase of review by the supervisors.

8.1.2 Collaborations within the Research Group

During the development of this thesis, the author has collaborated with other work in the context of the research group. Below, the list of papers is classified by importance order, and consequently by the time and effort spend by the author on them. The author have spent more than a year, full time, working in the fifth and sixth papers.

5. WATANABE, A. T.; SEBEM, R.; LEAL, A. B.; HOUNSELL, M. da S. Fault prognosis of discrete event systems: An overview. **Annual Reviews in Control**, v. 51, p. 100–110, 2021.

6. MAAS, D.; SEBEM, R.; LEAL, A. B. Multilayer architecture for fault diagnosis of embedded systems. **International Journal of Prognostics and Health Management**, PHM Society, v. 12, n. 2, dec 2021.

7. SCHULZE, L.; BERTOL, D. W.; SEBEM, R. Conventional and explicit MPC applied to robotic systems: a computational cost evaluation. In: **2021 IEEE 29th Mediterranean Conference on Control and Automation (MED)**. [S.l.: s.n.], 2021. p. 861–866.

8. SCHULZE, L.; SEBEM, R.; BERTOL., D. W. Performance of PSO and GWO algorithms applied in text-independent speaker identification. In: FILHO, C. J. A. B. et al. (Ed.). **Anais do 15º Congresso Brasileiro de Inteligência Computacional**. Joinville, SC: SBIC, 2021. p. 1–6.

8.1.3 Supervisions

The proposal in this work was extended in the bachelor thesis of Silva, Leal and Sebem (2021) and Teles, Leal and Sebem (2021). The author have supervised their works in the period of development of this thesis:

SILVA, M. F. d.; LEAL, A. B.; SEBEM, R. **Criação de estruturas de prioridades entre multi agentes autônomos em uma arquitetura de controle distribuído.** Trabalho de Conclusão de Curso (Barchelor's Thesis) — Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Curso de Engenharia Elétrica, 2021.

TELES, M. G.; LEAL, A. B.; SEBEM, R. **Criação de uma infraestrutura para testes de controle distribuído de sistemas multirrobôs**. Trabalho de Conclusão de Curso (Barchelor's Thesis) — Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Curso de Engenharia Elétrica, 2021.

REFERENCES

ALI, R. et al. URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence. **IEEE Access**, v. 9, p. 67064–67095, 2021.

ČAPKOVIč, F. A modular system approach to DES synthesis and control. In: **2009 IEEE Conference on Emerging Technologies Factory Automation**. [S.l.: s.n.], 2009. p. 1–8.

ČAPKOVIč, F. Cooperation of autonomous agents based on supervisory control of DES. In: **Melecon 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference**. [S.l.: s.n.], 2010. p. 178–183.

ČAPKOVIč, F.; SERIN, F. Supervisory control of agents cooperation. In: **2008 4th International IEEE Conference Intelligent Systems**. [S.l.: s.n.], 2008. v. 1, p. 6–8–6–13.

AUER, A.; DINGEL, J.; RUDIE, K. Concurrency control generation for dynamic threads using discrete-event systems. **Science of Computer Programming**, v. 82, p. 22 – 43, 2014. Special Issue on Automated Verification of Critical Systems (AVoCS'11).

BAKIBILLAH, A. S. M.; HASAN, M.; RAHMAN, M. M.; KAMAL, M. A. S. Predictive car-following scheme for improving traffic flows on urban road networks. **Control Theory and Technology**, v. 17, n. 4, p. 325–334, Nov 2019.

BASILE, F.; CHIACCHIO, P.; MARINO, E. D. An auction-based approach to control automated warehouses using smart vehicles. **Control Engineering Practice**, v. 90, p. 285 – 300, 2019.

BORDINI, R. H.; HüBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)**. USA: John Wiley & Sons, Inc., 2007. ISBN 0470029005.

CAI, K.; WONHAM, W. M. Supervisor localization: A top-down approach to distributed control of discrete-event systems. **IEEE Transactions on Automatic Control**, v. 55, n. 3, p. 605–618, March 2010.

CAI, K.; WONHAM, W. M. New results on supervisor localization, with application to multi-agent formations. **IFAC Proceedings Volumes**, v. 45, n. 29, p. 233 – 238, 2012. 11th IFAC Workshop on Discrete Event Systems.

CAI, K.; WONHAM, W. M. Supervisor localization of discrete-event systems based on state tree structures. **IEEE Transactions on Automatic Control**, v. 59, n. 5, p. 1329–1335, May 2014.

CAI, K.; WONHAM, W. M. New results on supervisor localization, with case studies. **Discrete Event Dynamic Systems**, v. 25, n. 1, p. 203–226, Jun 2015.

CASSANDRAS, C.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. 3rd. ed. [S.l.]: Kluwer Academic Publishers, 2021.

CHALAKI, B.; MALIKOPOULOS, A. A. Time-optimal coordination for connected and automated vehicles at adjacent intersections. **IEEE Transactions on Intelligent Transportation Systems**, p. 1–16, 2021.

CHEN, D. et al. Robust H∞ control of cooperative driving system with external disturbances and communication delays in the vicinity of traffic signals. **Physica A: Statistical Mechanics and its Applications**, v. 542, p. 123385, 2020. ISSN 0378-4371.

CHEN, J.; LIANG, H.; LI, J.; LV, Z. Connected automated vehicle platoon control with input saturation and variable time headway strategy. **IEEE Transactions on Intelligent Transportation Systems**, p. 1–12, 2020.

CHEN, J.; LIANG, H.; LI, J.; XU, Z. A novel distributed cooperative approach for mixed platoon consisting of connected and automated vehicles and human-driven vehicles. **Physica A: Statistical Mechanics and its Applications**, v. 573, 2021.

CHEN, T. et al. Connected and automated vehicle distributed control for on-ramp merging scenario: A virtual rotation approach. **Transportation Research Part C: Emerging Technologies**, v. 133, 2021.

CHEN, X.; MOREL, D.; RAKOTO-RAVALONTSALAMA, N. Multi-agent based supervisory control of an experimental manufacturing cell. **IFAC Proceedings Volumes**, v. 37, n. 11, p. 379 – 382, 2004. 10th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems 2004: Theory and Applications, Osaka, Japan, 26-28 July, 2004.

CHEN, X. et al. Non-signalized intersection network management with connected and automated vehicles. **IEEE Access**, v. 8, p. 122065–122077, 2020.

CHO, K.-H. A study on fault-tolerant control and operation of serial production systems. In: **Proceedings KORUS 2000. The 4th Korea-Russia International Symposium On Science and Technology**. [S.l.: s.n.], 2000. v. 2, p. 175–180 vol. 2.

CHO, K.-H.; LIM, J.-T. Multi-agent supervisory control of serial production systems for complementary fault-tolerance**this work was supported by korea research foundation grant. (krf-99-003-e00420). **IFAC Proceedings Volumes**, v. 33, n. 17, p. 793 – 798, 2000. 2nd IFAC Conference on Management and Control of Production and Logistics (MCPL 2000), Grenoble, France, 5-8 July 2000.

CHO, K.-H.; LIM, J.-T. Multiagent supervisory control for antifault propagation in serial production systems. **IEEE Transactions on Industrial Electronics**, v. 48, n. 2, p. 460–466, April 2001.

CHUNG, S. L.; LAFORTUNE, S.; LIN, F. Limited lookahead policies in supervisory control of discrete event systems. **IEEE Transactions on Automatic Control**, v. 37, n. 12, p. 1921–1935, Dec 1992.

CHUNG, S. L.; LAFORTUNE, S.; LIN, F. Recursive computation of limited lookahead supervisory controls for discrete event systems. **Discrete Event Dynamic Systems: Theory and Applications**, v. 3, n. 1, p. 71–100, 1993. Cited By 13.

CLARK, R. A. et al. Autonomous and scalable control for remote inspection with multiple aerial vehicles. **Robotics and Autonomous Systems**, v. 87, p. 258–268, 2017.

CURY, J. E. R. Teoria de controle supervisório de sistemas a eventos discretos. **V Simpósio Brasileiro de Automação Inteligente**, Canela-RS, 2001.

DEBADA, E. G.; GILLET, D. Virtual vehicle-based cooperative maneuver planning for connected automated vehicles at single-lane roundabouts. **IEEE Intelligent Transportation Systems Magazine**, v. 10, n. 4, p. 35–46, 2018.

DING, J.; LI, L.; PENG, H.; ZHANG, Y. A rule-based cooperative merging strategy for connected and automated vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 21, n. 8, p. 3436–3446, 2020.

DU, Z.; HOMCHAUDHURI, B.; PISU, P. Hierarchical distributed coordination strategy of connected and automated vehicles at multiple intersections. **Journal of Intelligent Transportation Systems: Technology, Planning, and Operations**, v. 22, n. 2, p. 144–158, 2018.

DULCE-GALINDO, J. A.; SANTOS, M. A.; RAFFO, G. V.; PENA, P. N. Autonomous navigation of multiple robots using supervisory control theory. In: **2019 18th European Control Conference (ECC)**. [S.l.: s.n.], 2019. p. 3198–3203.

DULCE-GALINDO, J. A.; SANTOS, M. A.; RAFFO, G. V.; PENA, P. N. Distributed supervisory control for multiple robot autonomous navigation performing single-robot tasks. **Mechatronics**, v. 86, p. 102848, 2022.

DURFEE, E. H.; LESSER, V. R. Chapter 10 - negotiating task decomposition and allocation using partial global planning. In: GASSER, L.; HUHNS, M. N. (Ed.). **Distributed Artificial Intelligence**. San Francisco (CA): Morgan Kaufmann, 1989. p. 229 – 243. ISBN 978-1-55860-092-8.

ETSI. **Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service**. [S.l.], 2019. v. 1.4.1. Available at: <https://www.etsi.org/deliver/etsi_EN/302600_302699/30263702/01.04.01_30/en_30263702v010401v.pdf>.

FENG, Y. et al. Design of distributed cyber-physical systems for connected and automated vehicles with implementing methodologies. **IEEE Transactions on Industrial Informatics**, v. 14, n. 9, p. 4200–4211, 2018.

FRANSEN, K. et al. A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems. **Computers & Operations Research**, v. 123, p. 105046, 2020.

FURCI, M.; PAOLI, A.; NALDI, R. A supervisory control strategy for robot-assisted search and rescue in hostile environments. In: **2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)**. [S.l.: s.n.], 2013. p. 1–4.

GORDON, D.; KIRIAKIDIS, K. Adaptive supervisory control of interconnected discrete event systems. In: **Proceedings of the 2000. IEEE International Conference on Control Applications. Conference Proceedings (Cat. No.00CH37162)**. [S.l.: s.n.], 2000. p. 935–940.

GOULET, N.; AYALEW, B. Distributed maneuver planning with connected and automated vehicles for boosting traffic efficiency. **IEEE Transactions on Intelligent Transportation Systems**, 2021.

GUAN, Y. et al. Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization. **IEEE Transactions on Vehicular Technology**, v. 69, n. 11, p. 12597–12608, 2020.

GUANETTI, J.; KIM, Y.; BORRELLI, F. Control of connected and automated vehicles: State of the art and future challenges. **Annual Reviews in Control**, v. 45, p. 18 – 40, 2018.

GUO, H. et al. A distributed adaptive triple-step nonlinear control for a connected automated vehicle platoon with dynamic uncertainty. **IEEE Internet of Things Journal**, v. 7, n. 5, p. 3861–3871, 2020.

GUO, Q.; LI, L.; BAN, X. J. Urban traffic signal control with connected and automated vehicles: A survey. **Transportation Research Part C: Emerging Technologies**, v. 101, p. 313–334, 2019. ISSN 0968-090X.

HADJ-ALOUANE, N. B.; LAFORTUNE, S.; LIN, F. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. **Discrete Event Dynamic Systems: Theory and Applications**, v. 6, n. 4, p. 379–427, 1996.

HüBNER, J. F. **Agent Oriented Programming with Jason @JaCaMo**. 2014. Lecture notes. Access on: may 2019. Available at: <http://jacamo.sourceforge.net/tutorial/gold-miners/aop-altissimo-14.pdf>.

HüBNER, J. F. **Multiagent Systems**. 2017. Lecture notes. Access on: may 2019. Available at: <http://jomi.das.ufsc.br/mas/slides/intro.pdf>.

HILL, R.; LAFORTUNE, S. Scaling the formal synthesis of supervisory control software for multiple robot systems. In: **2017 American Control Conference (ACC)**. [S.l.: s.n.], 2017. p. 3840–3847.

HIRAISHI, K. A formalism for decentralized control of discrete event systems. In: **Proceedings of the 41st SICE Annual Conference. SICE 2002.** [S.l.: s.n.], 2002. v. 1, p. 272–277 vol.1.

HU, Z.; HUANG, J.; YANG, D.; ZHONG, Z. Constraint-tree-driven modeling and distributed robust control for multi-vehicle cooperation at unsignalized intersections. **Transportation Research Part C: Emerging Technologies**, v. 131, p. 103353, 2021. ISSN 0968-090X.

HU, Z.; HUANG, J.; YANG, Z.; ZHONG, Z. Embedding robust constraint-following control in cooperative on-ramp merging. **IEEE Transactions on Vehicular Technology**, v. 70, n. 1, p. 133–145, 2021.

HUBBARD, P.; CAINES, P. E. Initial investigations of hierarchical supervisory control for multi-agent systems. In: **Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)**. [S.l.: s.n.], 1999. v. 3, p. 2218–2223 vol.3.

ITO, Y.; KAMAL, M. A. S.; YOSHIMURA, T.; AZUMA, S. Coordination of connected vehicles on merging roads using pseudo-perturbation-based broadcast control. **IEEE Transactions on Intelligent Transportation Systems**, v. 20, n. 9, p. 3496–3512, 2019.

JIANG, H.; PI, J.; LI, A.; YIN, C. Dynamic local path planning for intelligent vehicles based on sampling area point discrete and quadratic programming. **IEEE Access**, v. 10, p. 70279–70294, 2022.

JING, S. et al. Cooperative game approach to optimal merging sequence and on-ramp merging control of connected and automated vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 20, n. 11, p. 4234–4244, 2019.

KAMAL, M. A. S. et al. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, n. 3, p. 1136–1147, 2015.

KARIMADINI, M.; LIN, H.; LEE, T. H. Decentralized supervisory control: Nondeterministic transitions versus deterministic moves. In: **2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics**. [S.l.: s.n.], 2009. p. 1288–1293.

KATRINIOK, A. Nonconvex consensus ADMM for cooperative lane change maneuvers of connected automated vehicles. **IFAC-PapersOnLine**, v. 53, n. 2, p. 14336–14343, 2020. ISSN 2405-8963. 21st IFAC World Congress.

KATRINIOK, A.; ROSARIUS, B.; MäHöNEN, P. Fully distributed model predictive control of connected automated vehicles in intersections: Theory and vehicle experiments. **IEEE Transactions on Intelligent Transportation Systems**, p. 1–13, 2022.

KHAN, M. A. et al. A journey towards fully autonomous driving-fueled by a smart communication system. **Vehicular Communications**, v. 36, p. 100476, 2022. ISSN 2214-2096.

KING, J.; PRETTY, R. K.; GOSINE, R. G. Coordinated execution of tasks in a multiagent environment. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 33, n. 5, p. 615–619, Sep. 2003.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007.

KNEISSL, M. et al. Combined scheduling and control design for the coordination of automated vehicles at intersections. **IFAC-PapersOnLine**, v. 53, n. 2, p. 15259–15266, 2020. ISSN 2405-8963. 21st IFAC World Congress.

KOVALENKO, I.; TILBURY, D.; BARTON, K. The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. **Control Engineering Practice**, v. 86, p. 105 – 117, 2019.

LI, S. et al. Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities. **IEEE Intelligent Transportation Systems Magazine**, v. 9, n. 3, p. 46–58, 2017.

LI, S. E. et al. Synchronous and asynchronous parallel computation for large-scale optimal control of connected vehicles. **Transportation Research Part C: Emerging Technologies**, v. 121, p. 102842, 2020. ISSN 0968-090X.

LIU, C.; LIN, C.-W.; SHIRAISHI, S.; TOMIZUKA, M. Distributed conflict resolution for connected autonomous vehicles. **IEEE Transactions on Intelligent Vehicles**, v. 3, n. 1, p. 18–29, 2018.

LIU, M.; ZHAO, J.; HOOGENDOORN, S.; WANG, M. A single-layer approach for joint optimization of traffic signals and cooperative vehicle trajectories at isolated intersections. **Transportation Research Part C: Emerging Technologies**, v. 134, p. 103459, 2022. ISSN 0968-090X.

LIU, P.; OZGUNER, U.; ZHANG, Y. Distributed MPC for cooperative highway driving and energy-economy validation via microscopic simulations. **Transportation Research Part C: Emerging Technologies**, v. 77, p. 80–95, 2017. ISSN 0968-090X.

LIU, Y.; CAI, K.; LI, Z. On scalable supervisory control of multi-agent discrete-event systems. **IFAC-PapersOnLine**, v. 51, n. 7, p. 25–30, 2018.

LIU, Y.; CAI, K.; LI, Z. On scalable supervisory control of multi-agent discrete-event systems. **Automatica**, v. 108, p. 108460, 2019.

LIU, Y.; CAI, K.; LI, Z. On scalable supervisory control of multi-agent discrete-event systems. **CoRR**, abs/1704.08858, 2019. Available at: <http://arxiv.org/abs/1704.08858>.

LONC, B.; CINCILLA, P. Cooperative its security framework: Standards and implementations progress in europe. In: **2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)**. [S.l.: s.n.], 2016. p. 1–6.

LOPES, Y. K. et al. Supervisory control theory applied to swarm robotics. **Swarm Intelligence**, v. 10, n. 1, p. 65–97, Mar 2016.

LUO, J.; HE, D.; ZHU, W.; DU, H. Multiobjective platooning of connected and automated vehicles using distributed economic model predictive control. **IEEE Transactions on Intelligent Transportation Systems**, p. 1–15, 2022.

MAAS, D.; SEBEM, R.; LEAL, A. B. Multilayer architecture for fault diagnosis of embedded systems. **International Journal of Prognostics and Health Management**, PHM Society, v. 12, n. 2, dec 2021.

MAHULEA, C.; KLOETZER, M.; GONZÁLEZ, R. **Path Planning of Cooperative Mobile Robots Using Discrete Event Models**. [S.l.]: John Wiley & Sons, Ltd, 2020. ISBN 9781119486305.

MIRHELI, A.; TAJALLI, M.; HAJIBABAI, L.; HAJBABAIE, A. A consensus-based distributed trajectory control in a signal-free intersection. **Transportation Research Part C: Emerging Technologies**, v. 100, p. 161 – 176, 2019.

MOSER, D.; SCHMIED, R.; WASCHL, H.; RE, L. del. Flexible spacing adaptive cruise control using stochastic model predictive control. **IEEE Transactions on Control Systems Technology**, v. 26, n. 1, p. 114–127, 2018.

NHTSA. **Federal Motor Vehicle Safety Standards; V2V Communications**. [S.l.], 2017. v. 1.4.1. Available at: <https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/v2v_pria_12-12-16_clean.pdf>.

OKOLI, C. A guide to conducting a standalone systematic literature review. **Communications of the Association for Information Systems**, v. 37, Nov. 2015.

PARENT, M. Automated vehicles: Autonomous or connected? In: **2013 IEEE 14th International Conference on Mobile Data Management**. [S.l.: s.n.], 2013. v. 1, p. 2–2.

PENA, P. N.; CURY, J. E. R.; LAFORTUNE, S. Verification of nonconflict of supervisors using abstractions. **IEEE Transactions on Automatic Control**, v. 54, n. 12, p. 2803–2815, 2009.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**. Swindon, UK: BCS Learning & Development Ltd., 2008. (EASE'08), p. 68–77.

PETRILLO, A.; SALVI, A.; SANTINI, S.; VALENTE, A. S. Adaptive multi-agents synchronization for collaborative driving of autonomous vehicles with multiple

communication delays. **Transportation Research Part C: Emerging Technologies**, v. 86, p. 372–392, 2018. ISSN 0968-090X.

PHAM, M. T.; SEOW, K. T. Discrete-event coordination design for distributed agents. **IEEE Transactions on Automation Science and Engineering**, v. 9, n. 1, p. 70–82, Jan 2012.

PRAYITNO, A.; NILKHAMHANG, I. Distributed model reference control for cooperative tracking of vehicle platoons subjected to external disturbances and bounded leader input. **International Journal of Control, Automation and Systems**, 2022.

QUEIROZ, M. H. de. **Controle Supervisório Modular de Sistemas de Grande Porte**. Dissertação de Mestrado (Master's Thesis) — Universidade Federal do Estado de Santa Catarina, 2000.

QUEIROZ, M. H. de. **Controle Supervisório Modular e Multitarefa de Sistemas Compostos**. Tese de Doutorado (PhD Thesis) — Universidade Federal de Santa Catarina, Florianópolis, Maio 2004.

QUEIROZ, M. H. de; CURY, J. E. R. Modular supervisory control of large scale discrete event systems. **Proceedings of the 5th International Workshop on Discrete Event Systems: Analysis and Control**, Ghent, Belgium: Kluwer Academic Publishers, p. 103–110, 2000.

QUEIROZ, M. H. de; CURY, J. E. R. Modular multitasking supervisory control of composite discrete-event systems. **IFAC Proceedings Volumes**, v. 38, n. 1, p. 91–96, 2005. 16th IFAC World Congress.

QUEIROZ, M. H. de; CURY, J. E. R.; WONHAM, W. M. Multitasking supervisory control of discrete-event systems. **Discrete Event Dynamic Systems**, v. 15, n. 4, p. 375–395, Dec 2005.

RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event processes. **SIAM J. Control Optim.**, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 25, n. 1, p. 206–230, Jan. 1987.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event system. In: **Proceedings of the IEEE**. [S.l.: s.n.], 1989. v. 77, n. 1, p. 81–98.

RIOS-TORRES, J.; MALIKOPOULOS, A. A. Automated and cooperative vehicle merging at highway on-ramps. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 4, p. 780–789, 2017.

RIOS-TORRES, J.; MALIKOPOULOS, A. A. A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 5, p. 1066–1077, 2017.

ROHLOFF, K.; LAFORTUNE, S. The verification and control of interacting similar discrete-event systems. **SIAM Journal on Control and Optimization**, Society for Industrial and Applied Mathematics, v. 45, p. 634–667, 2006.

ROMANOVSKI, I.; CAINES, P. E. On multi-agent product systems: Graph MA products and partially observed MA products. In: **42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)**. [S.l.: s.n.], 2003. v. 3, p. 2680–2685 Vol.3.

ROMANOVSKI, I.; CAINES, P. E. On the supervisory control of multiagent product systems. **IEEE Transactions on Automatic Control**, v. 51, n. 5, p. 794–799, May 2006.

ROMANOVSKI, I.; CAINES, P. E. On the supervisory control of multi-agent product systems: Controllability properties. **Systems & Control Letters**, v. 56, n. 2, p. 113 – 121, 2007.

ROSZKOWSKA, E.; REVELIOTIS, S. A distributed protocol for motion coordination in free-range vehicular systems. **Automatica**, v. 49, n. 6, p. 1639–1653, 2013.

SCHULZE, L.; BERTOL, D. W.; SEBEM, R. Conventional and explicit MPC applied to robotic systems: a computational cost evaluation. In: **2021 IEEE 29th Mediterranean Conference on Control and Automation (MED)**. [S.l.: s.n.], 2021. p. 861–866.

SCHULZE, L.; SEBEM, R.; BERTOL., D. W. Performance of PSO and GWO algorithms applied in text-independent speaker identification. In: FILHO, C. J. A. B. et al. (Ed.). **Anais do 15º Congresso Brasileiro de Inteligência Computacional**. Joinville, SC: SBIC, 2021. p. 1–6.

SEBEM, R.; LEAL, A. B. Proposta de arquitetura para controle distribuído de sistemas a eventos discretos multiagentes autônomos. In: **Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI)**. Ouro Preto: [s.n.], 2019.

SEBEM, R.; LEAL, A. B.; BERTOL, D. W. Overview of distributed control for connected and automated vehicles. (In preparation).

SEBEM, R. et al. Architecture for scalable and distributed control of connected and automated vehicles with reconfigurable path planning. (To be submitted).

SEOW, K.-T.; MA, C.; YOKOO, M. Multiagent planning as control synthesis. In: **Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.** [S.l.: s.n.], 2004. p. 972–979.

SHI, H. et al. A deep reinforcement learning-based distributed connected automated vehicle control under communication failure. **Computer-Aided Civil and Infrastructure Engineering**, p. 1–19, 2022.

SILVA, M. F. d.; LEAL, A. B.; SEBEM, R. **Criação de estruturas de prioridades entre multi agentes autônomos em uma arquitetura de controle distribuído.** Trabalho de Conclusão de Curso (Barchelor's Thesis) — Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Curso de Engenharia Elétrica, 2021.

SILVA, M. F. d.; SEBEM, R.; LEAL, A. B.; BERTOL, D. W. Implementation strategies for distributed control of connected and automated vehicles. (In preparation).

SINGH, M.; HUHNS, M. **Service-Oriented Computing: Semantics, Processes, Agents**. [S.l.]: Wiley, 2006. ISBN 9780470091494.

SINGH, P. K.; NANDI, S. K.; NANDI, S. A tutorial survey on vehicular communication state of the art, and future research directions. **Vehicular Communications**, v. 18, p. 100164, 2019. ISSN 2214-2096.

STEINMETZ, E. et al. Collision-aware communication for intersection management of automated vehicles. **IEEE Access**, v. 6, p. 77359–77371, 2018.

SU, R.; LIN, L. Synthesis of control protocols for multi-agent systems with similar actions. In: **52nd IEEE Conference on Decision and Control**. [S.l.: s.n.], 2013. p. 6986–6991.

TAJALLI, M.; MEHRABIPOUR, M.; HAJBABAIE, A. Network-level coordinated speed optimization and traffic light control for connected and automated vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 22, n. 11, p. 6748–6759, 2021.

TAKAI, S.; USHIO, T. Supervisor synthesis for a class of concurrent discrete event systems. In: **42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)**. [S.l.: s.n.], 2003. v. 3, p. 2686–2691 Vol.3.

TAKAI, S.; USHIO, T. Supervisory control of a class of concurrent discrete event systems under partial observation. **Discrete Event Dynamic Systems**, v. 15, n. 1, p. 7–32, Mar 2005.

TATSUMOTO, Y.; SHIRAISHI, M.; CAI, K.; LIN, Z. Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. **Control Engineering Practice**, v. 81, p. 97 – 104, 2018.

TELES, M. G.; LEAL, A. B.; SEBEM, R. **Criação de uma infraestrutura para testes de controle distribuído de sistemas multirrobôs**. Trabalho de Conclusão de Curso (Barchelor's Thesis) — Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Curso de Engenharia Elétrica, 2021.

VEMULAPALLI, M.; DASGUPTA, S.; KUHL, J. G. Fault tolerant, scalable multi-agent control under medium access constraints. **IFAC Proceedings Volumes**, v. 41, n. 2, p. 6608–6613, 2008. 17th IFAC World Congress.

WANG, F.; CHEN, Y. A novel hierarchical flocking control framework for connected and automated vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 22, n. 8, p. 4801 – 4812, 2021. ISSN 15249050.

WANG, J.; ZHAO, X.; YIN, G. Multi-objective optimal cooperative driving for connected and automated vehicles at non-signalised intersection. **IET Intelligent Transport Systems**, v. 13, n. 1, p. 79–89, 2019.

WANG, M. Infrastructure assisted adaptive driving to stabilise heterogeneous vehicle strings. **Transportation Research Part C: Emerging Technologies**, v. 91, p. 276–295, 2018. ISSN 0968-090X.

WANG, X.; LEE, P.; RAY, A.; PHOPA, S. A behavior-based collaborative multi-agent system. In: **SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)**. [S.l.: s.n.], 2003. v. 5, p. 4242–4248 vol.5.

WATANABE, A. T.; SEBEM, R.; LEAL, A. B.; HOUNSELL, M. da S. Fault prognosis of discrete event systems: An overview. **Annual Reviews in Control**, v. 51, p. 100–110, 2021.

WEI, Y. et al. Dynamic programming-based multi-vehicle longitudinal trajectory optimization with simplified car following models. **Transportation Research Part B: Methodological**, v. 106, p. 102–129, 2017. ISSN 0191-2615.

WEISS, G. **Multiagent Systems**. [S.l.]: The MIT Press, 2013. ISBN 0262018896, 9780262018890.

WILSON, S. et al. The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. **IEEE Control Systems Magazine**, v. 40, n. 1, p. 26–44, 2020.

WU, J. et al. Distributed multilane merging for connected autonomous vehicle platooning. **Science China Information Sciences**, v. 64, n. 11, 2021.

WU, R. et al. A distributed trajectory control strategy for the connected automated vehicle in an isolated roundabout. **IET Intelligent Transport Systems**, v. 16, n. 2, p. 232–251, 2022.

WUTHISHUWONG, C.; TRAECHTLER, A. Consensus-based local information coordination for the networked control of the autonomous intersection management. **Complex & Intelligent Systems**, v. 3, n. 1, p. 17–32, 2017.

XIAO, W.; CASSANDRAS, C. G. Decentralized optimal merging control for connected and automated vehicles. In: **2019 American Control Conference (ACC)**. [S.l.: s.n.], 2019. p. 3315–3320.

XU, B. et al. Distributed conflict-free cooperation for multiple connected vehicles at unsignalized intersections. **Transportation Research Part C: Emerging Technologies**, v. 93, p. 322–334, 2018. ISSN 0968-090X.

YANG, Z.; FENG, Y.; LIU, H. X. A cooperative driving framework for urban arterials in mixed traffic conditions. **Transportation Research Part C: Emerging Technologies**, v. 124, 2021. ISSN 0968090X. Cooperative adaptive cruise control;Cooperative driving;Enabling technologies;Mixed-integer linear programming;Signalized intersection;State transition diagrams;Trajectory Planning;Volume fluctuations;.

YU, C. et al. Corridor level cooperative trajectory optimization with connected and automated vehicles. **Transportation Research Part C: Emerging Technologies**, v. 105, p. 405 – 421, 2019.

ZHANG, J. et al. Analysis and design on intervehicle distance control of autonomous vehicle platoons. **ISA Transactions**, v. 100, p. 446–453, 2020. ISSN 0019-0578.

ZHANG, R.; CAI, K. On supervisor localization based distributed control of discrete-event systems under partial observation. In: **2016 American Control Conference (ACC)**. [S.l.: s.n.], 2016. p. 764–769.

ZHANG, R.; CAI, K. Localization-based distributed control for large discrete-event systems under partial observation. In: **2018 Chinese Control And Decision Conference (CCDC)**. [S.l.: s.n.], 2018. p. 1492–1497.

ZHANG, Y.; CASSANDRAS, C. G. Decentralized optimal control of connected automated vehicles at signal-free intersections including comfort-constrained turns and safety guarantees. **Automatica**, v. 109, p. 108563, 2019.

ZHENG, Y. et al. Cooperative lane changing strategies to improve traffic operation and safety nearby freeway off-ramps in a connected and automated vehicles environment. **IEEE Transactions on Intelligent Transportation Systems**, v. 21, n. 11, p. 4605–4614, 2020.

ZHU, Y.; ZHU, F. Barrier-function-based distributed adaptive control of nonlinear CAVs with parametric uncertainty and full-state constraint. **Transportation Research Part C: Emerging Technologies**, v. 104, p. 249–264, 2019. ISSN 0968-090X.

ZHUANG, W.; XU, L.; YIN, G. Robust cooperative control of multiple autonomous vehicles for platoon formation considering parameter uncertainties. **Automotive Innovation**, v. 3, n. 1, p. 88–100, Mar 2020.