Safe and reconfigurable manufacturing: safety aware multi-agent control for Plug & Produce system

The International Journal of Advanced Manufacturing Technology (2024) 134:529–544
https://doi.org/10.1007/s00170-024-14112-7

ORIGINAL ARTICLE

Safe and reconfigurable manufacturing: safety aware multi-agent control for Plug & Produce system

Bassam Massouh[1] · Fredrik Danielsson[1] · Bengt Lennartson[2] · Sudha Ramasamy[1] · Mahmood Khabbazi[1]

Received: 22 February 2024 / Accepted: 6 July 2024 / Published online: 23 July 2024

Abstract
Plug & Produce aims to revolutionize manufacturing by enabling seamless machine integration into production processes without extensive programming. This concept, leveraging multi-agent systems (MAS), offers increased flexibility and faster production ramp-up times after reconfiguration. As automated manufacturing moves towards greater human integration, this paper addresses safe operation within the Plug & Produce concept. The main safety challenge arises from autonomous decision-making, as agents in the MAS lack awareness of the risk consequences of their behavior. Additionally, the difficulty of perceiving the system's exact behavior leads to the implementation of overly restrictive safety measures. This limits the system's flexibility and ability to make decisions for efficient production. This paper proposes a method utilizing multi-agent control to conduct automatic safety analysis and reason task allocations to avoid risks. The method's benefits are the generation of control actions that comply with safety requirements during operation, eliminating the need for overly restrictive safety measures and allowing more effective equipment utilization. The method's benefit is illustrated through a manufacturing scenario with two different configurations: one using a hazardous machine and the other using a less hazardous one. Formal verification using the model checker NuSMV demonstrated that safety requirements were satisfied in both configurations, without the need for manual modifications of the safety control system after reconfiguration. The results for this specific manufacturing scenario showed that there are more reachable states (20 states) in the safer second configuration, compared to the first configuration (16 states). This means that the presented control strategy dynamically adjusts the system's behavior to confirm safety. Hence, this method maintains safety without fixed safety rules that limit the operations.

Keywords Reconfigurable manufacturing · Plug & Produce · Safety · Risk assessment · Multi-agent system

1 Introduction

Modern production environments are required to cope with changing market demand, mass customization of products, and shorter product lifecycles. Reconfigurable Manufacturing Systems (RMS), characterized by their ease of adaptation to changes in product design and production requirements, surpass other manufacturing systems in throughput, cost-effectiveness, and the ability to accommodate product variety [1]. The concept of Plug & Produce is within the topic area of RMS, and it is centered around the automatic addition and removal of manufacturing devices with minimal intervention [2]. Furthermore, Plug & Produce systems adapt to a variety of new or modified products through in-house competencies that prioritize manufacturing processes rather than machine function programming [3]. The reconfiguration ability of the Plug & Produce system requires autonomy among the system's components, which is realized with multi-agent control [4]. In multi-agent control, each piece of equipment is controlled by an agent. These autonomous agents negotiate and decide the allocation of tasks automatically without manual intervention, achieving flexibility in job-shop task allocation through autonomy [5]. The multi-agent controller approach proved valid to support the concept of Plug & Produce [6, 7]. Notably, the IDEA project validated with industrial experiments the effectiveness of multi-agent control in achieving seamless shop-floor reconfiguration [8]. A more recent

✉ Bassam Massouh
bassam.massouh@hv.se

[1] Department of Industrial Automation, University West, Trollhättan, Sweden

[2] Department of Systems and Control Technology, Chalmers University of Technology, Gothenburg, Sweden

advancement in multi-agent control of Plug & Produce is the Configurable Multi-Agent System (C-MAS), which implies that all agents are configured rather than programmed, as in many other similar concepts. C-MAS shortens engineering time and reduces the level of needed competencies [9]. Indeed, autonomous decision-making improves efficiency by automating task distribution and execution and providing optimization possibilities during runtime. However, the autonomous feature complicates the implementation of safety measures. The complexity is twofold: (I) after a system reconfiguration, the system must be examined to identify possible new risks, and new safety measures must be implemented if needed. This requirement adds to the time needed for the system to reach full operational capacity which opposes the advantages of the Plug & Produce concept of flexibility [10]. (II) The risk assessment must be performed in a way that perceives every possible task distribution and plan execution, and the consequent safety measures must cover all these possibilities. The difficulty of perceiving the exact behavior of the system leads to the implementation of overly restrictive safety measures. This in turn limits the system's flexibility and ability to make decisions for efficient production. There is a gap in the scientific literature and the current practices regarding addressing these safety-related issues. Traditional industrial safety solutions in manufacturing often deal with predefined scenarios and perceived flexibility, which are insufficient for the unforeseen changes characteristic of Plug & Produce systems. In Plug & Produce systems, manufacturing equipment and produced parts are not always known during the design stage, making traditional safety approaches inadequate. Furthermore, the review of related works provided in this paper shows that there is a need for proactive safety methods, within the smart controller of Plug & Produce to dynamically adapt to new configurations without compromising system flexibility. This research is motivated by the need to address the gap in ensuring the safe operation of RMS, specifically Plug & Produce systems, while preserving their inherent flexibility. The objective is to propose a method that enables the realization of the Plug & Produce concept in modern manufacturing environments. As automated manufacturing moves towards more human integration, this research aligns with the goals of Industry 5.0, which focuses on designing human-centric manufacturing systems. This paper addresses the issue of safety assurance after reconfiguration and the safety challenges coming from the autonomous nature of the Plug & Produce system. It presents a control method within the Plug & Produce multi-agent controller, realized through an algorithm that forms an integral component of the strategy utilized by individual agents. The paper has implications for human-centric and reconfigurable manufacturing as the proposed method is an enabler for safe and flexible Plug & Produce systems and it is essential for their practical implementation and realization. The proposed control strategy ensures safety after reconfiguration without the need for manual modification of the control logic, and it copes with the autonomous decision-making by planning safe operations without limiting flexibility or reliance on overly restrictive measures. The contribution of this work is twofold: (I) a control method that enables the Plug & Produce multi-agent controller to automatically generate the runtime safety requirements and autonomously distribute the shop-floor tasks in a way that satisfies the safety requirements. The proposed method makes safety more proactive rather than reactive, and it complements the safety-related part of the control system by incorporating within the multi-agent controller the capability to predict risk events. Accordingly, the method enables the control system to autonomously schedule operations in a way that avoids safety stops. (II) The method prevents risk situations by specifically avoiding combinations of equipment that could potentially lead to risk. However, it does not impose a complete restriction on equipment operation. Instead, the method allows the equipment to participate in other processes that are considered safe and do not pose a risk of harm. This flexibility allows for more effective utilization of equipment while maintaining safety requirements. There are limitations to the scenarios where the methodology proposed in this paper can be applied. This paper focuses on the operational safety of RMS, specifically Plug & Produce systems at the system control level. It does not address the safe design of individual components, physical safety barriers, or emergency stops within a manufacturing system. Additionally, ergonomics and hazards from unintended resource use are beyond this paper's scope, as these topics are covered by established safety assurance methods and engineering practices. The proposed method does not involve altering or bypassing the safety PLC, which remains an essential component of the safety infrastructure. Instead, our method complements the safety PLC by providing an automatic safety assessment method that is integrated into the operational planning, ensuring that the system can adapt dynamically to new configurations while maintaining stringent safety. The proposed method is specifically for reconfigurable and discrete manufacturing cells and may not apply to process manufacturing environments. Industries like chemical processing or food and beverage manufacturing, which rely on continuous processes, have different safety requirements that this methodology may not adequately address. This paper is organized into five sections; Sect. 1 introduces the topic area, the research problem and the contribution of this paper. Section 2 presents the related work and the theoretical framework of the proposed approach. Section 3 describes the proposed method for safety reasoning in MAS control Plug & Produce. Section 4 describes a manufacturing scenario, the results, and the discussion, and Sect. 5 describes the conclusion and future work.

2 Related work

Despite the wealth of literature on the design and management of RMS, the topic of safety assurance and risk management remains limited. Recent studies have recognized the importance of human integration and are now addressing safety assurance in RMS. One line of research advocates for incorporating safety considerations at the layout design stage of RMS [11]. This approach is complemented by efforts aimed at optimizing ergonomics to enhance the safety and efficiency of operations [12]. Tools have been developed to aid safety managers in conducting safety analyses and in the ergonomic design of system layouts [13]. The design of the safety-related part of the control system has also been addressed, with an approach to functional safety that revolves around the dynamic and rapid reconfiguration of the safety system to adapt to changes within the manufacturing setup [14]. These design approaches emphasize adjusting the safety measures rather than planning the distribution of tasks to avoid potential risks. There is a need for proactive risk management techniques that can as well retain system flexibility without imposing restrictive safety measures [15]. The second line of research focuses on the incorporation of advanced sensors and monitoring to improve safety. Recently, the main application of advanced sensor technology has been improving the health and safety of workers [16]. Integrating sensing and prediction into robot controllers allows robots to dynamically adapt their actions based on human interactions and predefined task execution plans [17]. Also, advanced sensors enable robots to determine the best control strategies for various levels of human–robot interaction. They can switch to an appropriate collaborative mode based on sensor data, such as stopping the robot if a human is detected too close or replanning the robot's path to avoid collisions [18]. While these technologies are crucial in robotics, their application is currently limited to robotic systems and has not been extended to other machines and processes in manufacturing settings. Additionally, hazards in Plug & Produce systems are not confined to a single process or device; they emerge when several components are configured together in the system [19]. This necessitates methods for planning on the system controller level. Furthermore, integrating sensor data with existing manufacturing systems can be complex and costly, requiring significant upfront investment in technology [20]. In this work, the objective is to address the risks within the multi-agent control of the RMS proactively. Multi-agent control is advantageous in RMS as agents autonomously accommodate new scenarios through collaborative decision-making among human workers, machines, and robotic systems [21]. It enables optimization of manufacturing processes and addresses the challenges posed by dynamic production environments and customer demands [22, 23]. In the topic of safety, scheduling in multi-agent control is employed for collision avoidance. Key concepts include path planning, where algorithms determine the routes, agents should take to reach their destinations without collisions [24]. Temporal aspects are crucial to avoid collisions, especially in dynamic environments, by scheduling the timing of movements to ensure that two agents do not occupy the same space at the same time [25]. This has been applied in autonomous vehicles to ensure that self-driving cars do not collide with each other or with human-driven vehicles [26] and in robotic swarms to coordinate large numbers of robots and avoid collisions [27]. Despite the advancements in MAS and dynamic scheduling, the application of these technologies has had a limited focus in the domain of safety of manufacturing, primarily concentrating on collision avoidance in autonomous vehicle contexts, rather than comprehensive manufacturing systems. Additionally, the literature review reveals a gap in proactive safety management strategies within RMS. Most existing research emphasizes reactive or immediate solutions rather than ongoing, dynamic risk management strategies. Notably, existing agent architectures in manufacturing propose a part agent that is responsible for scheduling and resource allocation, utilizing model-based reasoning for production goals and cooperative scheduling [28, 29]. In this work, a model-based approach is chosen for safety reasoning in multi-agent control of Plug & Produce and the theoretical framework is further described in the next subsection.

2.1 Theoretical framework

The model-based approach in MAS leverages the use of computational models to represent the system, its components, and the interactions among agents. Model-based reasoning is the agents using computational models to reason and collaborate on decision-making. This approach typically relies on creating a formalized representation of the agents' goals and the available resources to achieve these goals [30]. Within this framework, agents utilize their internal models to reason about the environment, predict the consequences of their actions, and make informed decisions to achieve their goals [31, 32]. On the other hand, the model-based safety approaches are centered on a formalism model of the control system and the safety-related information, followed by a safety analysis of the formal system model [33]. It offers automation opportunities for safety analysis in complex systems [34]. These methods enable automation of the traditional safety analysis [35–37] and allow quicker design processes that integrate safety considerations [38]. In addition, model-based approaches closely integrate safety and the design models which enables the reusability of the safety-related information. This is particularly important in a changing environment such as a

Plug & Produce as safety models can be traced and reused after a change in the system [39]. Furthermore, model-based techniques enhance interoperability and reusability of safety knowledge contributing to the automatic generation of safety requirements, which can be deployed manually with system support [40] or automatically to a safety PLC [41]. Also, model-based safety approaches enable verification of safety before production. Verification is a critical step in ensuring the efficacy of safety measures as the safety of manufacturing processes can be verified before the start of production on the factory floor. For verification of safety, model-based approaches leverage formal methods in traditional and autonomous control systems [30, 42, 43]. By adding safety models to the system model, agents can use these models to reason about safe behavior and adapt to the new manufacturing requirements. This enables adaptive safety measures that can dynamically adjust to changing conditions. This is commonly used in the control of autonomous vehicles, as it supports adaptive safety measures, enabling vehicles to dynamically adjust their behavior in response to changing traffic conditions [44–46].

The literature reviewed in this study provides a comprehensive overview of existing approaches to model-based reasoning for multi-agent control in reconfigurable manufacturing and its role in facilitating adaptability through collaborative decision-making. Previous research in the safety of RMS has predominantly focused on reactive functional safety measures aimed at minimizing risks associated with known scenarios or solutions that are limited to robot collision avoidance and individual processes. In contrast, the work presented in this paper extends beyond existing frameworks and introduces a novel solution to address safety in reconfigurable manufacturing on a system level, prioritizing proactive safety measures, through planning, to avert risk scenarios altogether.

3 Safety reasoning in MAS control Plug & Produce

This section describes the proposed method for safety reasoning in MAS control Plug & Produce. It includes three subsections; the first subsection describes the proposed control architecture, the second subsection describes the method for agent reasoning for risk scenarios, and the third subsection describes the proposed algorithm to implement the agent reasoning.

3.1 The MAS model for control of a Plug & Produce system

In the MAS model, there are two types of agents that are parts and resources. These agents have interfaces that are used to match and negotiate with each other. A part agent represents a product, and it is configured with goals that describe how the part is manufactured from the start until a finalized product is achieved. A resource agent represents a manufacturing asset. For instance, a robot, a machine, or a human operator is a resource in the Plug & Produce system. A resource agent is configured with skills that represent the capabilities of the corresponding physical manufacturing asset. Both part's goals and resource's skills utilize process plans which are data entities that include a structured text code. These codes include demands for skills on abstract interfaces. Meaning that a certain skill is demanded to be executed without specifying the resource that owns the skill. This non-restrictive design of process plans enables a flexibility aspect, in which a part autonomously determines the execution of an abstract plan based on its strategy. Another aspect of flexibility enabled by this design is that skills can also be abstract. This is typical for a skill that requires other skills to be able to complete its task. An example is a robotic gripper tool that has the skill "*load*." This skill will include a process plan that demands another skill "*moveTool*" that can be offered by a robot.

In this model, the part agent has a strategy that constitutes the methods to reach its goals. This strategy includes choosing a process plan that solves the goal with the lowest cost and allocating tasks/skills to resources. Also, the goals have preconditions that describe the criteria that must be fulfilled before running the goal. The preconditions are generated automatically based on the manually created sequence of goals. If several goals have fulfilled preconditions, the goals are allowed to run in parallel if enough resources are available for a parallel behavior. Indeed, the resource agents communicate to their corresponding physical entities through the instructions in the structured text code in the process plans. For the human operator, its resource agent negotiates with other agents to include the operator within the production plans, and it controls the human-system interaction by delivering instructions and receiving input from the operator.

This paper proposes a formalism to incorporate and model safety-related information within MAS. This is a generic model that can be applied to versatile applications within Plug & Produce. In this proposal, agents use the safety models to perform automatic risk assessments of their goals' process plans. Based on the risk assessment, the agents decide a behavior that avert risks in the production of parts. Hazard-related information is integral to the risk assessment process. Hazard identification, according to the safety standard ISO 12100, is to list all hazards within the determined machine limitation. This includes investigating the intended use of the machine and identifying any source of harm within the associated task. In a Plug & Produce environment, this corresponds to hazard identification of resources. Hazard identification of a single resource includes determining the set of skills that the resource can perform

and identifying the hazards associated with each of the skills. When a process plan of a skill is composed in a way that only includes the agent logic that instructs the physical resource, the hazards can be easily identified and fetched from the hazard information identified in the design phase. However, when the process plan includes other skills to be achieved by other resources, then it is required to fetch the hazards related to these remote skills. Modeling the hazards associated with a skill allows for including this safety-related information within the system's logical configuration and allows for the reusability of this information when the skill is demanded for the execution of a process plan.

Within the context of the proposed formalism, an agent interface is denoted as *if* and is defined as the tuple, $if = \langle S_{if}, V_{if} \rangle$, where S_{if} is a set of skills of the interface and V_{if} is a set of variables. A skill $s \in S_{if}$ is defined as the tuple:$s = \langle \pi_s, \tau_s, H_s, o_s \rangle$, where π_s is the process plan of skill$s$, τ_s is the type of skill, H_s is a set of hazards, and o_s is operational space. A single hazard denoted $h \in H_s$ is defined with the tuple,$h = \langle k_h, \tau_h \rangle$, where k_h is risk level and τ_h is the type of target skill by the hazard$h$. The operating space o_s is the space in which a resource performs the skill s according to its configuration, and this attribute is used to detect the unsafe situations in a certain space. The term is adopted from the standard ISO 10218 and is used to represent the occupied space by a resource while it performs the skill$s$. The operating space is defined by its shape, and it has a coordinate system that is relative to the resource coordinate system. The tuples defined for a MAS control Plug & Produce are summarized in a UML diagram as shown in Fig. 1, which defines the complete model of MAS control Plug & Produce and incorporates the safety models within the system control architecture.

3.2 The agent's reasoning for risk scenarios

Agent's reasoning on risk scenarios uses the information in the proposed safety models. To explain this process, an example is visualized in Fig. 2 and described as follows. Assume process plan π_{load} for the skill "*load*" on a gripper tool. The process plan includes three skills, "*pick*," "*moveTool*," and "*place*." It is sufficient to choose the skill "*moveTool*" to further describe the part strategy.

The skill "*moveTool*," which is achieved by a robot, generates a high-risk hazard $h_{moveTool}$. In addition, $h_{moveTool}$ is harmful to the operator. Based on this, the hazard of the skill "*moveTool*" is configured as $h_{moveTool} = \langle high_{h_{moveTool}}, opSkill_{h_{moveTool}} \rangle$. To discover the risk scenario, the proposed part's strategy instructs to check the risk level of the hazard, and in this case, it finds it is a high-risk hazard. Then, the strategy checks if there is any other skill in the system that overlaps with the operational space of skill "*moveTool*." Assuming the system is designed in a way that the operator's skill "*load*" has operational space that overlaps with the skill "*moveTool*" of the robot. In this case, the part strategy instructs to check if the skill "*moveTool*" harms the operator while performing the skill "*load*." The part finds that the skill "*load*" that is owned by the operator resource and has the type ''*opSkill*'' is targeted by the skill "*lmoveTool* "that is owned by the robot. Based on this information, the part books both skills "*load*" and "*moveTool*." This means that the part not only uses the robot to achieve its plan but also prevents the operator from being demanded, by another plan execution, to perform the skill load. Note that the operator is only prevented from doing a load and remains unrestricted in executing other skills.

3.3 The part agent method for goal scheduling according to the safety analysis

The purpose of a part is to achieve its goals, and the part's strategy instructs the planning of goals. The part strategy includes methods of finding matching interfaces for the abstract plans [47]. Also, the part strategy includes methods for planning the execution of concurrent goals [9]. This section describes further development to the part strategy and includes a method for executing the goals safely with the lowest production cost. The proposed method runs on each part, and when several parts are added to the system, to avoid conflict over resources, one part at a time can plan.

A part plans its goals to avoid the violation of the safety requirements which are derived from automatic risk assessment of the parts plans. In line with the guidelines described in the standard ISO 12100, the risk assessment method consists of three steps. First, detect runtime hazards in the process plan; second, evaluate the risk level of the detected hazards; and third, perform risk prevention using control actions. The runtime hazard detection finds the hazard associated with the skill, considering that the skill's process plan also may include other skills on other resources. This can only be done in runtime when all process plans are made executable. To achieve this, Algorithm 1 is proposed to be included in the part strategy for planning the safe execution of goals.

To achieve the part's strategy, the algorithm generates runtime variables that include, for a given interface *if* , an interface schedule denoted as sc_{if}. The interface schedule has states included in a set St and it registers at each state $st \in St$, the skills S_{if} that are used in the parts' plans.

The algorithm can be summarized by the following steps:

1. Get the next goal from the ordered set of goals G_p.
2. Get all executable process plans for the selected goal.
3. Select the plan with the least cost among the alternatives.
4. For each skill s in the selected plan, assess the risks associated with the skill using the risk assessment function.
5. Book the skill s and the skills in S^{τ} that are targeted by the hazards of the skill.

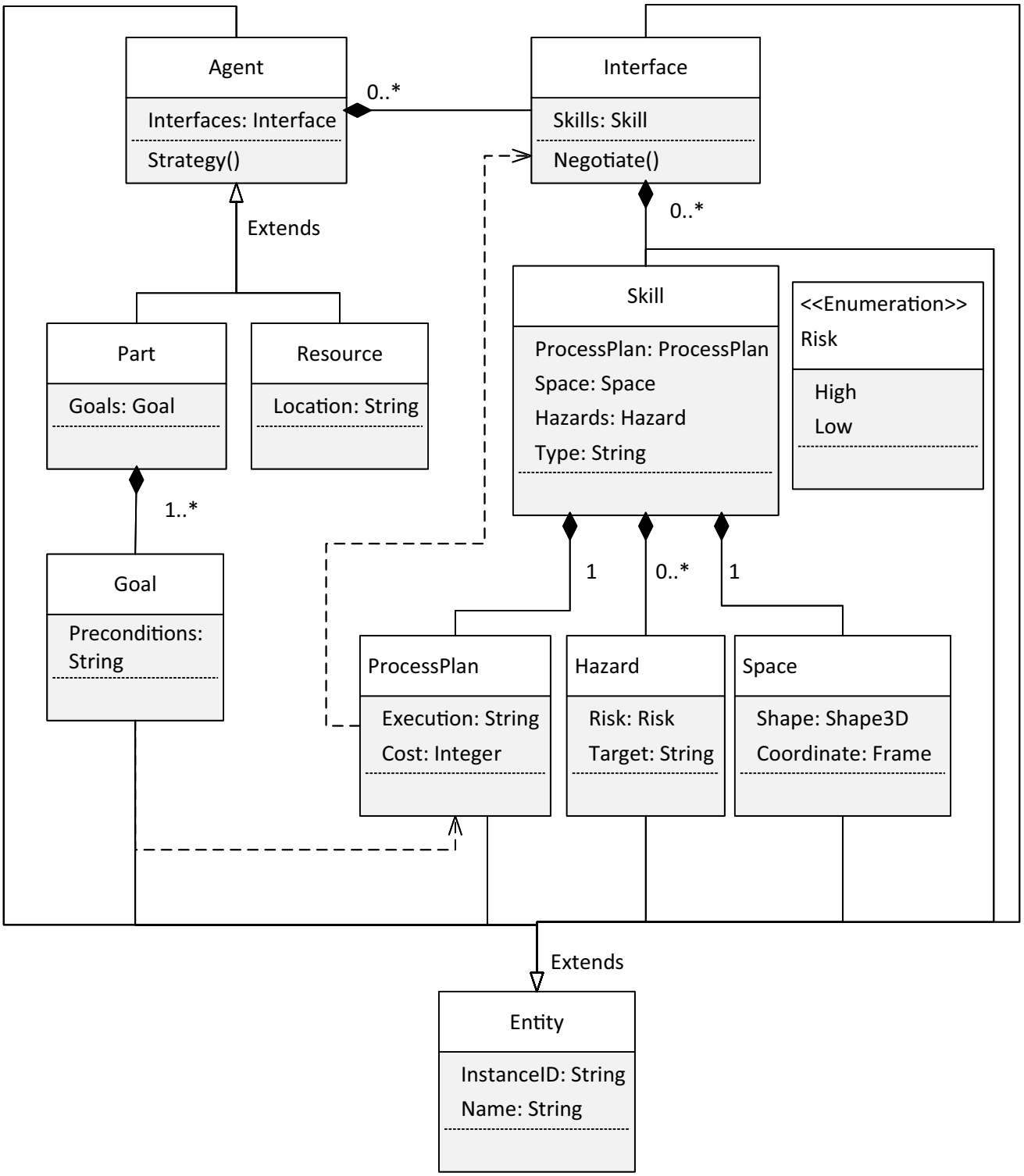

Fig. 1 The complete model of MAS control Plug & Produce, incorporating the safety models within the system control architecture

The outermost loop iterates through the goals G_p and performs steps 2 to 5 for each goal in order. The function *getAllExecutableProcessplans*() returns all the possible combinations of resources within alternative executions of part's p goal g_p, and they are stored in the set Π_g^{alt}. Then, the function *selectPlanWithLeastCost*() returns the least cost executable process plan. The inner loop iterates through the skills of the selected process plan and includes risk assessment and booking of the skills.

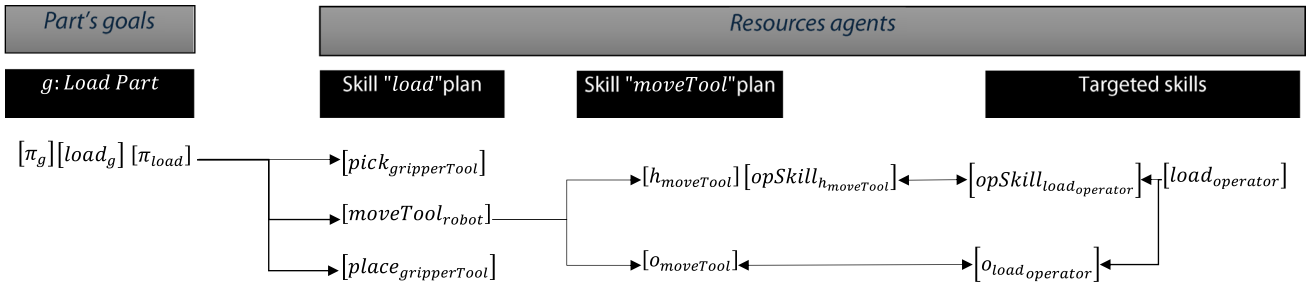

Fig. 2 The discovery of a risk scenario that includes a robot and an operator

Algorithm 1 imposes the safety requirements in the planning of a part agent goals.

```
safePlanningPartGoals(Part: p)

    while g = getNextGoalInLine(G_p)
        Π_g^alt = getAllExecutableProcessplans(g)
        π_g = selectPlanWithLeastCost(Π_g^alt)
        st = 0
        while s = getNextSkillInLine(π_g)
            [IF^τ, S^τ] = riskAssessment(s)
            st = book(s, [< IF^τ, S^τ >], st + 1)
        end while
    end while

riskAssessnet(skill: s)

    S^τ = [], IF^τ=[]
    H_s = getSkillHazards(s)
    for each h ∈ H_s
        if k_h = high
            [⟨IF^o, S^o⟩] = getAllOverlappingSkills(o_s)
                for each < if^o, s^o > ∈ [< IF^o, S^o >]
                if τ_{s^o} = τ_h
                    S^τ = [S^τ, s^o]
                    IF^τ = [IF^τ, if^o]
                end if
            end for
        end if
    end for
    return [⟨IF^τ, S^τ⟩]

book(skill: s, interfaces: IF, skills: S, state: st)

    if_s = getInterfaceForSkill(s)
    sc = getSchedule(if_s)
    for each < if, s > ∈ < IF, S >
        sc = [sc, getSchedule(if)]
    end for
        st = findAvailableState(s, S, st, sc)
        addToSchedule(s, S, st, sc)
    return st
```

The function *riskAssessment*(*skill* : *s*) takes a skill, *s*, as an input and returns a set of pairs of interfaces and skills, IF^{τ} and S^{τ}, that are targeted by the input skill. The function uses the *getSkillHazards*() function to get the hazards that are locally on skill *s* and hazards that are on the remote skills used by the skill's process plan π_s. The skill's hazards are stored in the set H_s. Next, for each $h \in H_s$, the risk k_h is checked, and if the risk is "high," all skills that have operational space that overlaps with the operational space o_s of skill *s* are collected. The function *getAllOverlappingSkills*() returns the overlapping skills and their interfaces. The returned list of skills and interfaces is stored in the set $[\langle IF^o, S^o \rangle]$. For each overlapping skill s^o, which shares the same hazard type $\tau_{s^o} = \tau_h$, the function adds the skill and its interface to the S^{τ} and IF^{τ} lists. Finally, the function *riskAssessment*(*skill* : *s*) returns the IF^{τ} and S^{τ} list as output.

The function *book*(*skill* : *s*, *interfaces* : *IF*, *skills* : *S*, *state* : *st*) takes a skill *s*, a list of interfaces *IF*, a list of skills *S*, and a state *st*, as input parameters. The function first gets the interface associated with the input skill *s*, using the *getInterfaceForSkill*() function. It then retrieves the current schedule *sc*, associated with that interface using the *getSchedule*() function. For each interface and skill in the *IF* and *S* lists, the schedule of each interface is added to *sc*, which makes *sc* a total schedule for all the involved interfaces. The function *findAvailableState*(s, *S*, *st*, *sc*) returns the first state in which the skill *s* and all the skills of set *S* are available in the schedule *sc*. Then, the returned state *st* is passed to the function *addToSt*(s, *S*, *st*, *Sc*), which schedules the skill *s* and the skills in the set *S* in *sc*, at the available state *st*. Finally, the function *addToSt*(s, *S*, *st*, *Sc*) returns the updated state *st*, as output.

The booking function books the targeted skills only and not the whole interface that it belongs to. This prevents the targeted skill from being scheduled by other parts at state *st* and at the same time allows the other skills on the same interface to be scheduled by other parts. There is here an expected benefit by preventing a resource from being allocated to an unsafe task, while the resource remains available for allocation to other safe tasks. This ensures that the resource is efficiently utilized.

The logic of the algorithm has been formally verified by modeling the part behavior using Extended Finite State Machines (EFSM) [48]. The model is checked if it satisfies the safety requirements that are formalized from the model-based analysis performed by the function *riskAssesment*(). The NuSMV solver [49] is used to check if the parts strategy and the proposed method satisfy the safety requirements, which are described in more detail in the next section.

4 Formal verification of a manufacturing scenario

In this section, to validate the efficacy of the suggested method, a Plug & Produce manufacturing scenario is formalized. Model checking is employed for formal verification of the strategy [35]. This section is divided into five subsections. The first subsection describes the manufacturing scenario, the second subsection describes the resource models, the third subsection describes the part models, the fourth subsection resents the safety requirements in this scenario, and the fifth subsection presents the verification result and discusses the satisfaction of the safety requirements.

4.1 Description of the manufacturing scenario

The manufacturing scenario includes a part type *p* which has three goals: goal g_1 to be prepared, goal g_2 to be loaded into a machine, and goal g_3 to be machined. For simplicity, each goal has one process plan, and each process plan has one skill. Also, the scenario includes four resources, two machines, a robot, and an operator. Each resource has one interface, and each interface has one skill. The robot interface has the "*load*" skill that loads the part into a machine, and the machines' interfaces have the "*machine*" skill that processes the part, and the operator agent interface has the "*prepare*" skill. For simplification of description and as each interface has one skill, the whole resource is considered booked if its skill is booked for a part. The abstract layout of the manufacturing scenario is shown in Fig. 3.

The operational spaces of each of the different resources' skills are represented with different dotted rectangles and all operational spaces overlap. To count for unplanned operator actions, the operational spaces are monitored by safety sensors, and in case the operator, in an unplanned manner, enters another resource's space, the safety control logic enforces a safety stop.

The scenario includes two configurations, each with a different machine. In the first configuration, the robot skill and the first machine skill are hazardous to the operator and therefore have high risk levels. In the second configuration, only the robot skill has a hazard that targets the operator and the second machine, which has a low-risk level, replaces the first machine.

4.2 Resource models

Based on the description of the scenario, the resources are defined by the following tuples.

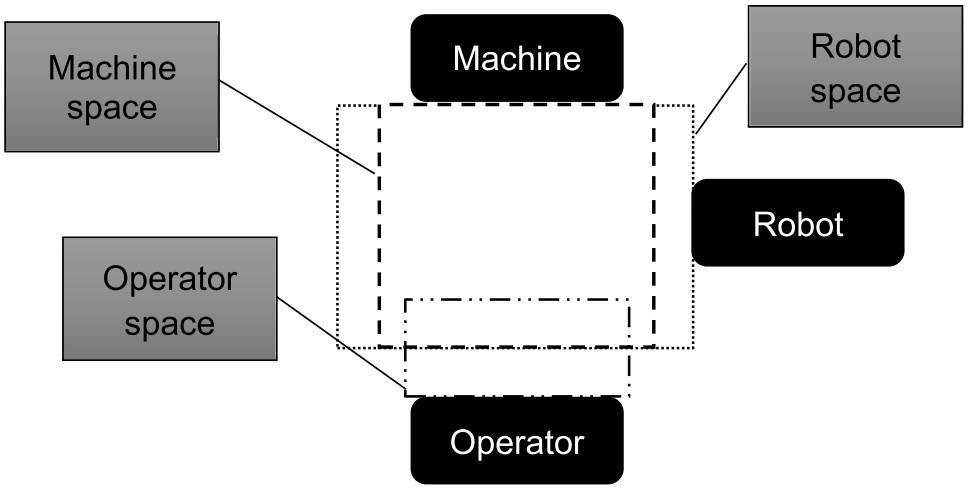

Fig. 3 The abstract layout of the manufacturing scenario

The robot resource has the interface if_R with the skill *load* and is defined as $if_R = \langle \{load_R\}, V_R \rangle$, where $load_R = \langle \pi_{load}, robotSkill, h_{load}, o_{load} \rangle$ and $h_{load} = \langle high, operatorSkill \rangle$. The skill *load* has operational space o_{load}, and it has a hazard h_{load} which is a high-risk hazard, i.e., $k_{h_{load}} = high$, and it targets the operator skill meaning that $\tau_{h_{load}} = operatorSkill$.

Similarly, the first machine resource has interface if_{M1} with the skill *machine* and is defined as $if_{M1} = \langle \{machine_{M1}\}, V_{M1} \rangle$, where $machine_{M1} = \langle \pi_{machine_{M1}}, machineSkill, h_{machine_{M1}}, o_{machine_{M1}} \rangle$ and $h_{machineSkill_{M1}} = \langle high, operatorSkill \rangle$.

Likewise, the second machine resource has interface if_{M2} with the skill *machine* and is defined as $if_{M2} = \langle \{machine_{M2}\}, V_{M2} \rangle$, where $machine_{M2} = \langle \pi_{machine_{M2}}, machineSkill, h_{machineSkill_{M2}}, o_{machine_{M2}} \rangle$ and $h_{machineSkill_{M2}} = \langle low, operatorSkill \rangle$.

The skill *machineSkill* of the second machine resource has a low-risk hazard to the operator. Therefore, it has a different hazard model than the first machine and the risk level of the hazard $k_h = low$. Finally, the operator resource agent has interface if_{op} that has the operator skill *prepare*, defined as $if_{Op} = \langle \{prepare_{Op}\}, V_{op} \rangle$, where $prepare_{Op} = \langle \pi_{prepare}, operatorSkill, \emptyset, o_{prepare} \rangle$. The skill *prepare* has no hazard as this skill does not generate any harm to the operator or the other resources.

4.3 Part model

Consider a set P of parts, where each part $p \in P$ is assumed to be modeled by an EFSM [48]. An EFSM is an ordinary Finite State Machine (FSM), also called an automaton, where a set of variables V is also included. The total discrete state space of an EFSM is the combination of the locations (the states in an ordinary FSM) and the values of the involved variables. The variables are used to determine conditions (guards) that must hold before a transition from one location to another may occur. When a transition occurs, the values of some variables can also be updated, also called actions, and the updated variables are determined by adding a prime on the variables resulting in the updated variable set $V\prime$. As an illustration, assume that a variable $v = 1$ is a guard and after the transition the updated value is $v\prime = 0$. This is then expressed by the formula $v = 1 \wedge v\prime = 0$ as a transition condition. All such possible transition conditions including the variables in V and $V\prime$ are included in set denoted F_V, and a complete EFSM is defined by the tuple $ES = \langle \Sigma, Q, \rightarrow, Q^o, Q^w \rangle$. The EFSM for a part p is defined by the tuple $ES_p = \langle \Sigma, Q, \rightarrow, Q^o, Q^w \rangle$, where Σ is a set of events including the completed skill event, Q is a finite set of locations, $\rightarrow \subseteq Q \times \Sigma \times F_V \times Q$ is conditional transition relation including the set F_V of all possible expressions on variables and their transition updates, $Q^o \subseteq Q$ is a set of initial locations, and Q^w is a set of marked locations. There are three part model variables controlled by Algorithm 1, which are involved in transition guards. The variables are $V_p \in \{R, Op, M\}$ and $V_p\prime \in \{R\prime, Op\prime, M\}$, where R, Op, and M are the variables representing robot, operator, and a machine, in a specific configuration. The domain of each variable is $\{A, B\}$ where A means that the resource is available and B that it is booked. Figure 4 shows the EFSM of a part p in the first configuration in which the first machine is used. In this model, q_0 is the initial location and the conditional transition from q_0 to q_1 makes the variable $Op = B$ as the operator skill "*prepare*" is used to achieve the plan of the first goal g_1. The operator is made unavailable (booked) for goals g_2 and g_3 to not be able to interact with the hazardous skills of the robot and the machine. The model shows four conditional transitions and the final one is to the marked location q_4 at which the event "*mc*" represents the completion of the skill "*machineSkill*." This skill is performed by the first machine resource that is included in the plan of the third goal g_3.

Figure 5 shows the EFSM, of a part p in the second configuration, in which the second machine is used. In this EFSM, it is shown that the operator skill."*prepare* "is available at location q_3 and this is because the "*machineSkill*,"

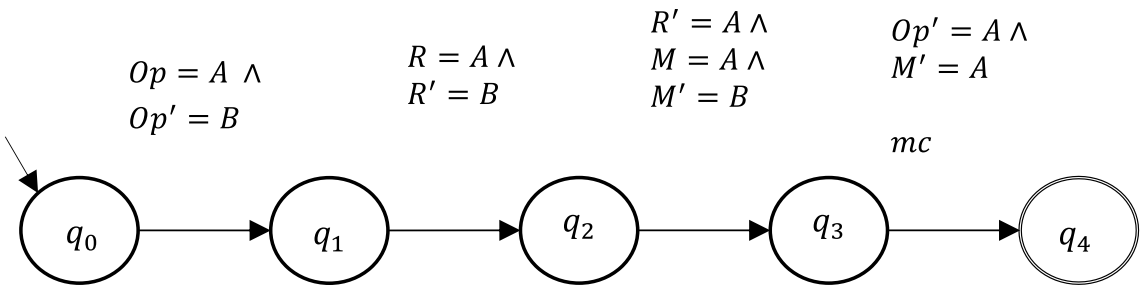

Fig. 4 EFSM for a part p in Configuration 1

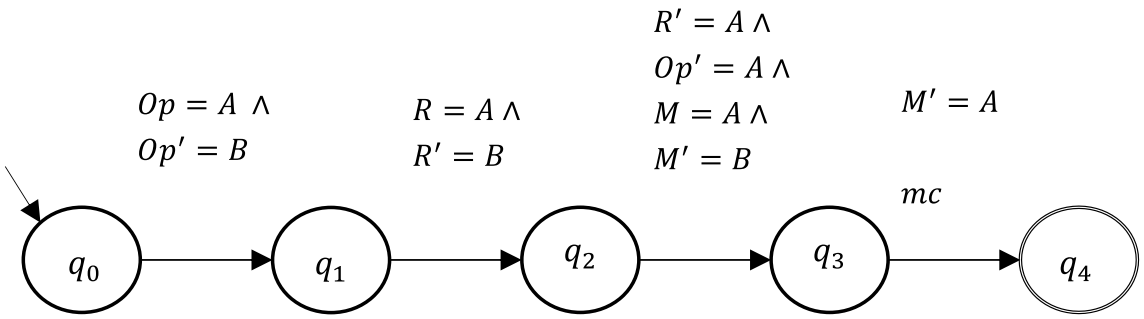

Fig. 5 EFSM for a part p in Configuration 2

in contrary with the first configuration, is not hazardous to the operator and the operator can interact with the machine. The event "*mc*" represent that the second machine has completed" *machineSkill*" and the part life cycle is completed.

4.4 Safety requirements: first and second configuration

Consider a part p_i and any other part p_j. Now assume that the system includes n parts, meaning that $|P| = n$. The resulting EFSM from the parallel composition of all parts $\mathrm{ES}_{p_1} \parallel \cdots \parallel \mathrm{ES}_{p_n}$ models the instantiation of n part agents concurrently. Algorithm 1 generates a part behavior described by the ESFM in the previous subsection. To prove that the total behavior of all parts is safe, the safety requirements are derived and then the parallel composition of all parts $\mathrm{ES}_{p_1} \parallel \cdots \parallel \mathrm{ES}_{p_n}$ is checked such that it satisfies the safety requirements. The safety requirement that concerns the reachability of the forbidden location (q_{ik}, q_{jm}) is described as $q_{ik} \in Q_i$ and $q_{jm} \in Q_j$ and $\mathrm{G}\,\neg(q_{ik} \wedge q_{jm})$, where G is the temporal logical modal operator globally, and this property must be satisfied in every globally reachable state. The forbidden state describes that part p_i is not allowed to be in the location q_k at the same time as part p_j is in location q_m.

In the presented scenario, the operator and the robot are working concurrently in the locations (q_{i2}, q_{j1}) and (q_{i1}, q_{j2}). Also, the operator and the machine are working concurrently in the locations (q_{i3}, q_{j1}) and (q_{i1}, q_{j3}) and in the first machine configuration this is not allowed. Thus, the safety requirements in the first machine configuration are $G\neg(((q_{i2} \wedge q_{j1}) \vee (q_{i3} \wedge q_{j1})) \wedge ((q_{i1} \wedge q_{j2}) \vee (q_{i1} \wedge q_{j3})))$. $G\neg()$ means that the statement within parenthesis is not true at any time. In the second configuration where the second machine is used, the safety requirement is $G\neg((q_{i2} \wedge q_{j1}) \wedge (q_{i1} \wedge q_{j2}))$. This only forbids the operator and the robot to work concurrently, while the machine and the operator now can perform their skills in the same area.

4.5 Test results and discussion

The EFSM models were built using the formal model-checking software NuSMV [49], and the results were obtained from the simulation of the part agents' strategy including resource scheduling and safety analysis. The models used for the simulation are presented in the Appendix. It is noticed that with the first configuration that involves the first unsafe machine, the composition of the parts' schedules achieves risk reduction by only scheduling the safe states as shown in the reachability graph in Fig. 6, including 16 reachable states, while the total global state space has 200 states. The unsafe locations are never reached, which are (q_{i2}, q_{j1}) and (q_{i1}, q_{j2}) which include the operator and the robot working concurrently or the locations (q_{i3}, q_{j1}) and (q_{i1}, q_{j3}) in which the operator and the machine are working concurrently. In the first

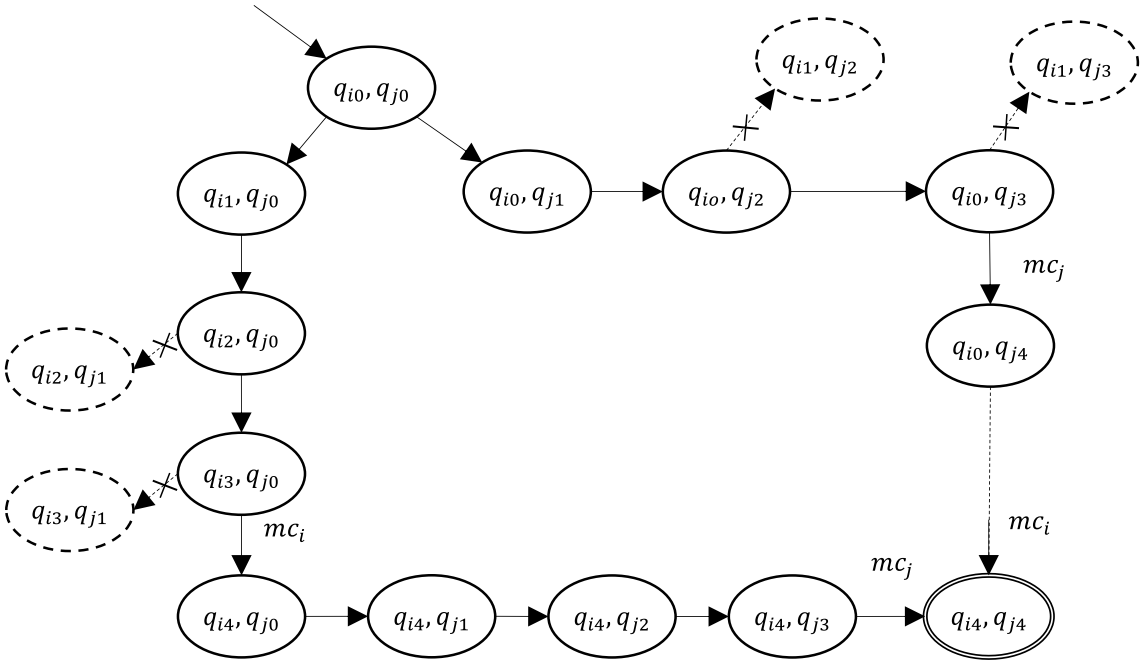

Fig. 6 Reachability graph for two parts' composition applying Configuration 1. The dashed states are unsafe states. mc is the event of "machineSkill" completed and q_{i4},q_{j4} is the final location where the production of parts is completed

configuration, the reachability graph shows that parts are produced sequentially as expected and that is because the operator skill *prepare* which is the first skill in the part plan cannot be initiated if any other machine is working.

In the second configuration that includes the second safer machine, it is noticed that the composition of parts schedules achieves risk reduction by only scheduling the safe states as shown in the reachability graph in Fig. 7, but now with 20 reachable states within the same number of global states as in the first configuration with 200 states. This is understandable as more restrictions are implemented by the controller in the first configuration due to the presence of more risks. In the second configuration, as it is safe for the operator to work concurrently with the machine, a part plan may be scheduled to be parallel to another part. Also, the reachability graph shows that the final event for a part p_i, mc_i (machine completed for part *i*), can happen before the start of the first skill *prepare* in a part p_j plan, as in the first configuration, but also later when skill *load* is scheduled for part p_j.

The safety requirements for each of the configurations were checked, and the results showed that the model of the parts' joint schedules satisfies the requirements. This means that safety requirements are always met based on each part's self-scheduling of its goals, without the need for manual risk assessment and modification of the safety-related part of the controller.

5 Conclusion

The Plug & Produce concept, facilitated by MAS, offers seamless machine integration and rapid production ramp-up times after reconfiguration. This paper proposes a novel method for employing the MAS control of Plug & Produce systems to perform model-based safety analysis and generate control actions that ensure compliance with safety requirements during production. This method removes the need to manually modify the safety-related part of the control logic after reconfiguration. The proposed method includes automatic risk assessment performed by agents in line with the safety regulations, and it enables foreseeing risk events and scheduling the operation pre-emptively to avert risk scenarios rather than reactively responding to them. This significantly reduces the reliance on physical barriers and emergency stop mechanisms. The formal verification of the method demonstrated that safety requirements are consistently satisfied in two different configurations, validating the method's efficacy. Quantitative results from the formal validation further underscore the effectiveness of the method. The manufacturing scenario tested involved two configurations with different levels of risk. In the first configuration, where a hazardous machine was used, the reachability graph showed 16 safe states out of a possible 200 global states, effectively avoiding unsafe states where concurrent operations could lead to hazards. In

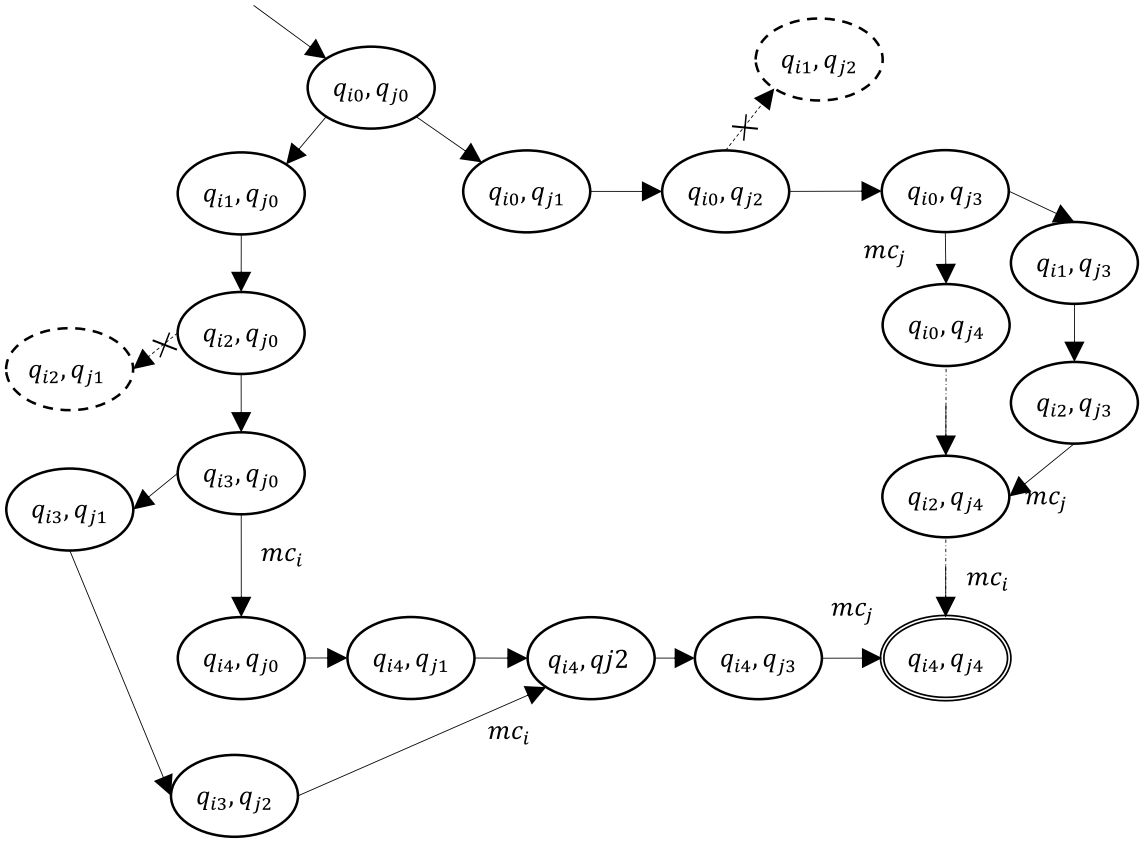

Fig. 7 Reachability graph for the composition of two parts applying Configuration 2. The dashed states are unsafe states. mc is the event of "machineSkill" completed and q_{i4},q_{j4} is the final location where the production of parts is completed

contrast, the second configuration, featuring a safer machine, resulted in 20 safe states within the same global state space. This increase in reachable safe states demonstrates the method's ability to maintain safety while allowing for greater flexibility and parallel operations. In comparing our approach with existing methods, several distinctions stand out. Traditional safety solutions often rely on predefined scenarios and fixed safety measures, which limit flexibility and responsiveness to unforeseen changes characteristic of Plug & Produce systems. Our method contrasts by offering proactive safety through dynamic planning, significantly enhancing system flexibility without compromising safety. Advanced sensor-based safety methods, while improving human–robot interaction safety, often involve high costs and complex integration processes. Our approach leverages MAS reasoning capabilities to cover the entire manufacturing system more comprehensively and cost-effectively. Furthermore, while MAS has been primarily focused on collision avoidance in autonomous vehicle contexts, our method extends these technologies to provide a comprehensive safety solution for RMS. The implications of this work are profound for human-centric and reconfigurable manufacturing environments. By ensuring safe operations after reconfiguration without manual intervention, our method supports the practical implementation and realization of flexible Plug & Produce systems. It allows for more effective utilization of equipment, maintaining safety without overly restrictive measures, thereby fostering a safer, more adaptable production environment. Future work will consider achieving global lowest-cost schedules for all parts. This must include a dynamic cost that is negotiated between parts based on their schedules. Additionally, it is important to develop a method for rescheduling part's plan when failures occur which includes considering adjusting schedules accordingly. The continuous optimization of safety assurance methods within Plug & Produce contributes to the feasibility and implementation of this solution for manufacturing.

Appendix

The following section shows the NuSMV model of the parts in the first and second configurations described in Sect. 4.3. The safety requirements for each configuration are formulated as linear temporal logic specifications to check if it is satisfied.

Part model and specification for the first configuration

```
MODULE part(op, ma, ro)                -- defining a part model with three variables, op for the operator, ma for the
machine and ro for the robot.
VAR
        location: {q0, q1, q2, q3, q4};   -- the part has four locations as described in Section 4.3
ASSIGN                                 -- defining the initial location, transitions to the next locations, and the
variables values.
        init(location)   := q0;
        next(location) := case
                location = q0  & op = available  :q1;
                location = q1  & ro = available   :q2;
                location = q2  & ma = available  :q3;
                location = q3                    :q4;
                TRUE                             :location;
        next(op) := case
                location = q0   & op = available : booked;
                location = q3                    : available;
                TRUE                             : op;
                esac;
        next(ma) := case
                location = q2   & ma = available : booked;
                location = q3                    : available;
                TRUE                             : ma;
                esac;
        next(ro) := case
                location = q1   & ro = available   : booked;
                location = q2   & ma = available : available;
                TRUE                              : ro;
                esac;

MODULE main
VAR
        op      : {available,booked};    -- variable op can take two values, available or booked.
        ma      : {available,booked};    -- variable ma can take two values, available or booked.
        ro      : {available,booked};    -- variable ro can take two values, available or booked.
        p1      : process part(op,ma,ro); -- initiating part p1
        p2      : process part(op,ma,ro); -- initiating part p2

ASSIGN                                 -- assigning initial values for variables.
        init(op) := available;
        init(ma) := available;
        init(ro) := available;

-- first configuration safety specifications. The result is true and the specifications are satisfied.
LTLSPEC G !(p1.location=q2 & p2.location=q1) | (p1.location=q3 & p2.location=q1)
LTLSPEC G !(p1.location=q1 & p2.location=q2) | (p1.location=q1 & p2.location=q3)
```

Part model and specification for the second configuration

```
MODULE part(op,ma,ro)                -- defining a part model with three variable, op for the operator, ma for the
machine and ro for the robot.
VAR
        location: {q0,q1,q2,q3,q4};  -- the part has four locations as described in Section 4.3
ASSIGN                               -- defining the initial location, transitions to next locations, and the variables
values.
        init(location) := q0;
        next(location) := case
                location = q0  & op =available   :q1;
                location = q1  & ro =available   :q2;
                location = q2  & ma =available   :q3;
                location = q3                    :q4;
                TRUE                             :location;
                esac;
        next(op) := case
                location = q0  & op =available   : booked;
                location = q2  & ma =available   : available;
                TRUE                             : op;
                esac;

        next(ma) := case
                location = q2   & ma =available  : booked;
                location = q3                    : available;
                TRUE                             : ma;
                esac;

        next(ro) := case
                location = q1   & op =available  : booked;
                location = q2   & ma =available  : available;
                TRUE                             : ro;
                esac;

MODULE main
VAR
        op      : {available,booked};    -- variable op can take two values, available or booked.
        ma      : {available,booked};    -- variable ma can take two values, available or booked.
        ro      : {available,booked};    -- variable ro can take two values, available or booked.
        p1      : process part(op,ma,ro); -- initiating part p1
        p2      : process part(op,ma,ro); -- initiating part p2

ASSIGN                               -- assigning initial values for variables.
        init(op) := available;
        init(ma) := available;
        init(ro) := available;

-- second configuration safety specification. The result is true and the specifications are satisfied.
LTLSPEC G !((p1.location=q2 & p2.location=q1) & (p1.location=q1 & p2.location=q2))
```

Author contribution The study conception and design were performed by Bassam Massouh and Fredrik Danielsson. Material preparation and data collection were performed by Bassam Massouh and Fredrik Danielsson and Sudha Ramasamy. Analysis was performed by Bassam Massouh, Fredrik Danielsson, and Bengt Lennartson. The first draft of the manuscript was written by Bassam Massouh, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by University West. The authors declare that no funds, grants, or other support was received during the preparation of this manuscript.

Declarations

Competing interests The authors declare no competing interests.

References

1. Khan AS, Homri L, Dantan JY, Siadat A (2022) An analysis of the theoretical and implementation aspects of process planning in a reconfigurable manufacturing system. Int J Adv Manuf Technol 119:5615–5646. https://doi.org/10.1007/s00170-021-08522-0
2. Arai T, Aiyama Y, Maeda Y et al (2000) Agile assembly system by "plug and produce." CIRP Ann 49:1–4. https://doi.org/10.1016/S0007-8506(07)62883-2
3. Nilsson A, Danielsson F, Svensson B (2023) From CAD to Plug & Produce: a generic structure for the integration of standard industrial robots into agents. Int J Adv Manuf Technol 128:5249–5260. https://doi.org/10.1007/s00170-023-12280-6
4. Arai T, Aiyama Y, Sugi M, Ota J (2001) Holonic assembly system with Plug and Produce. Comput Ind 46:289–299. https://doi.org/10.1016/S0166-3615(01)00111-7
5. Arai T, Izawa H, Maeda Y et al (2003) Real-time task decomposition and allocation for a multi-agent robotic assembly cell. In: Proceedings of the IEEE international symposium on assembly and task planning, 2003. IEEE, Besancon, France, pp 42–47. https://doi.org/10.1109/ISATP.2003.1217185
6. Rocha A, Di Orio G, Barata J et al (2014) An agent based framework to support plug and produce. In: 2014 12th IEEE international conference on industrial informatics (INDIN). IEEE, Porto Alegre, Brazil, pp 504–510. https://doi.org/10.1109/INDIN.2014.6945565
7. Leitao P, Barbosa J, Pereira A et al (2016) Specification of the PERFoRM architecture for the seamless production system reconfiguration. In: IECON 2016 - 42nd annual conference of the IEEE industrial electronics society. IEEE, Florence, Italy, pp 5729–5734. https://doi.org/10.1109/IECON.2016.7793007
8. Onori M, Lohse N, Barata J, Hanisch C (2012) The IDEAS project: plug & produce at shop-floor level. Assem Autom 32:124–134. https://doi.org/10.1108/01445151211212280
9. Nilsson A, Danielsson F, Svensson B (2023) Customization and flexible manufacturing capacity using a graphical method applied on a configurable multi-agent system. Robo Comput-Integr Manuf 79:102450. https://doi.org/10.1016/j.rcim.2022.102450
10. Koo CH, Vorderer M, Junker S et al (2018) Challenges and requirements for the safety compliant operation of reconfigurable manufacturing systems. Procedia CIRP 72:1100–1105. https://doi.org/10.1016/j.procir.2018.03.038
11. Besbes M, Mahjoub YI, Bonte T et al (2021) Solving facility layout problem with safety consideration of reconfigurable manufacturing and assembly systems. Procedia CIRP 104:1942–1947. https://doi.org/10.1016/j.procir.2021.11.328
12. Bortolini M, Botti L, Galizia FG, Regattieri A (2020) Bi-objective design and management of reconfigurable manufacturing systems to optimize technical and ergonomic performances. Appl Sci 11:263. https://doi.org/10.3390/app11010263
13. Sallez Y, Berger T, Bonte T (2020) The concept of "safety bubble" for reconfigurable assembly systems. Manuf Lett 24:77–81. https://doi.org/10.1016/j.mfglet.2020.03.015
14. Etz D, Denzler P, Fruhwirth T, Kastner W (2022) Functional safety use cases in the context of reconfigurable manufacturing systems. In: 2022 IEEE 27th international conference on Emerging Technologies and Factory Automation (ETFA). IEEE, Stuttgart, Germany, pp 1–8. https://doi.org/10.1109/ETFA52439.2022.9921448
15. Vital-Soto A, Olivares-Aguila J (2023) Manufacturing systems for unexpected events: an exploratory review for operational and disruption risks. IEEE Access 11:96297–96316. https://doi.org/10.1109/ACCESS.2023.3311362
16. Menolotto M, Komaris D-S, Tedesco S et al (2020) Motion capture technology in industrial applications: a systematic review. Sensors 20:5687. https://doi.org/10.3390/s20195687
17. Lavit Nicora M, Ambrosetti R, Wiens GJ, Fassi I (2021) Human-robot collaboration in smart manufacturing: robot reactive behavior intelligence. Journal of Manufacturing Science and Engineering, Transactions of the ASME 143:. https://doi.org/10.1115/1.4048950
18. Hashemi-Petroodi SE, Thevenin S, Kovalev S, Dolgui A (2020) Operations management issues in design and control of hybrid human-robot collaborative manufacturing systems: a survey. Annu Rev Control 49:264–276. https://doi.org/10.1016/j.arcontrol.2020.04.009
19. Massouh B, Ramasamy S, Svensson B, Danielsson F (2022) A framework for hazard identification of a collaborative plug & produce system. In: Sanfilippo F, Granmo OC, Yayilgan SY, Bajwa IS (eds) Intelligent Technologies and Applications. INTAP 2021. Communications in computer and information science, vol 1616. Springer, Cham. https://doi.org/10.1007/978-3-031-10525-8_12
20. Benmessabih T, Slama R, Havard V, Baudry D (2024) Online human motion analysis in industrial context: a review. Eng Appl Artif Intell 131:107850. https://doi.org/10.1016/J.ENGAPPAI.2024.107850
21. Pulikottil T, Estrada-Jimenez LA, Ur Rehman H et al (2023) Agent-based manufacturing — review and expert evaluation. Int J Adv Manuf Technol 127:2151–2180. https://doi.org/10.1007/s00170-023-11517-8
22. Barenji AV, Barenji RV, Roudi D, Hashemipour M (2017) A dynamic multi-agent-based scheduling approach for SMEs. Int J Adv Manuf Technol 89:3123–3137. https://doi.org/10.1007/s00170-016-9299-4
23. Tan Q, Tong Y, Wu S, Li D (2019) Modeling, planning, and scheduling of shop-floor assembly process with dynamic cyber-physical

interactions: a case study for CPS-based smart industrial robot production. Int J Adv Manuf Technol 105:3979–3989. https://doi.org/10.1007/s00170-019-03940-7
24. Alkazzi J-M, Okumura K (2024) A comprehensive review on leveraging machine learning for multi-agent path finding. IEEE Access 12:57390–57409. https://doi.org/10.1109/ACCESS.2024.3392305
25. Choudhury S, Gupta JK, Kochenderfer MJ et al (2022) Dynamic multi-robot task allocation under uncertainty and temporal constraints. Auton Robot 46:231–247. https://doi.org/10.1007/s10514-021-10022-9
26. Zhou W, Chen D, Yan J et al (2022) Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. Autono Intell Syst 2:5. https://doi.org/10.1007/s43684-022-00023-5
27. Shibuya T, Endo T, Matsuno F (2023) Experimental investigation of distributed navigation and collision avoidance for a robotic swarm. Artif Life and Robot 28:50–61. https://doi.org/10.1007/s10015-022-00843-x
28. Vrba P, Tichý P, Mařík V et al (2011) Rockwell automation's holonic and multiagent control systems compendium. IEEE Transactions on Systems, Man, and Cybernetics. Part C (Appl Rev) 41:14–30. https://doi.org/10.1109/TSMCC.2010.2055852
29. Kovalenko I, Balta EC, Tilbury DM, Barton K (2023) Cooperative product agents to improve manufacturing system flexibility: a model-based decision framework. IEEE Trans Autom Sci Eng 20:440–457. https://doi.org/10.1109/TASE.2022.3156384
30. Ma C, Provost J (2019) Introducing plant features to model-based testing of programmable controllers in automation systems. Control Eng Pract 90:301–310. https://doi.org/10.1016/j.conengprac.2019.07.006
31. Haddad Y, Salonitis K, Emmanouilidis C (2021) Design of redistributed manufacturing networks: a model-based decision-making framework. Int J Comput Integr Manuf 34:1011–1030. https://doi.org/10.1080/0951192X.2021.1946860
32. Huang M, Lin X, Feng Z et al (2023) A multi-agent decision approach for optimal energy allocation in microgrid system. Electr Power Syst Res 221:109399. https://doi.org/10.1016/j.epsr.2023.109399
33. Joshi A, Heimdahl MPE, Miller SP, Whalen MW (2006) Model-based safety analysis. NASA technical report, NASA/CR-2006–213953 (NASA Langley Research Center, Hampton, 2006)
34. Gradel S, Aigner B, Stumpf E (2022) Model-based safety assessment for conceptual aircraft systems design. CEAS Aeronaut J 13:281–294. https://doi.org/10.1007/s13272-021-00562-2
35. Dickerson CE, Roslan R, Ji S (2018) A formal transformation method for automated fault tree generation from a UML activity model. IEEE Trans Reliab 67:1219–1236. https://doi.org/10.1109/TR.2018.2849013
36. Guiochet J (2016) Hazard analysis of human-robot interactions with HAZOP-UML. Saf Sci 84:225–237. https://doi.org/10.1016/j.ssci.2015.12.017
37. Koo CH, Schröck S, Vorderer M et al (2020) A model-based and software-assisted safety assessment concept for reconfigurable PnP-systems. Procedia CIRP 93:359–364. https://doi.org/10.1016/j.procir.2020.03.076
38. Saenz J, Behrens R, Schulenburg E et al (2020) Methods for considering safety in design of robotics applications featuring human-robot collaboration. Int J Adv Manuf Technol 107:2313–2331. https://doi.org/10.1007/s00170-020-05076-5
39. Massouh B, Danielsson F, Ramasamy S et al (2024) Online hazard detection in reconfigurable plug & produce systems. In: Lecture notes in mechanical engineering. Porto, Portugal, pp 889–897. https://doi.org/10.1007/978-3-031-38241-3_97
40. Koch T (2019) Approach for an automated safety configuration for robot applications. Procedia CIRP 84:896–901. https://doi.org/10.1016/j.procir.2019.04.280
41. Askarpour M, Lestingi L, Longoni S et al (2021) Formally-based model-driven development of collaborative robotic applications. J Intell Rob Syst 102:59. https://doi.org/10.1007/s10846-021-01386-2
42. Krishnan R, Bhada SV (2022) Integrated system design and safety framework for model-based safety assessment. IEEE Access 10:79311–79334. https://doi.org/10.1109/ACCESS.2022.3193495
43. Horel JB, Ledent P, Marsso L et al (2023) Verifying collision risk estimation using autonomous driving scenarios derived from a formal model. J Intell Robot Systems: Theory Appl 107:59. https://doi.org/10.1007/s10846-023-01808-3
44. Li N, Yao Y, Kolmanovsky I et al (2022) Game-theoretic modeling of multi-vehicle interactions at uncontrolled intersections. IEEE Trans Intell Transp Syst 23:1428–1442. https://doi.org/10.1109/TITS.2020.3026160
45. Chen S, Dong J, Ha P et al (2021) Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. Comput-Aided Civ Infrastruct Eng 36:838–857. https://doi.org/10.1111/mice.12702
46. Lv X, Li W, Wang J (2022) Safety-field-based path planning algorithm of lane changing for autonomous vehicles. Int J Control Autom Syst 20:564–576. https://doi.org/10.1007/s12555-020-0942-3
47. Bennulf M, Danielsson F, Svensson B, Lennartson B (2021) Goal-oriented process plans in a multiagent system for Plug & Produce. IEEE Trans Industr Inf 17:2411–2421. https://doi.org/10.1109/TII.2020.2994032
48. Mohajerani S, Malik R, Fabian M (2016) A framework for compositional nonblocking verification of extended finite-state machines. Discret Event Dyn Syst 26:33–84. https://doi.org/10.1007/s10626-015-0217-y
49. Cimatti A, Clarke E, Giunchiglia E et al (2002) NuSMV 2: an opensource tool for symbolic model checking. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). pp 359–364. https://doi.org/10.1007/3-540-45657-0_29