Enhancing Awareness of Industrial Robots in Collaborative Manufacturing

Alessandro Umbrico *, Amedeo Cesta and Andrea Orlandini
Institute of Cognitive Sciences and Technlogies, National Research Council, Rome, Italy
E-mail: alessandro.umbrico@istc.cnr.it

Abstract. The diffusion of Human-Robot Collaborative cells is prevented by several barriers. Classical control approaches seem not yet fully suitable for facing the variability conveyed by the presence of human operators beside robots. The capabilities of representing heterogeneous knowledge representation and performing abstract reasoning are crucial to enhance the flexibility of control solutions. To this aim, the ontology SOHO (Sharework Ontology for Human Robot Collaboration) has been specifically designed for representing Human-Robot Collaboration scenarios, following a context-based approach. This work brings several contributions. This paper proposes an extension of SOHO to better characterize behavioral constraints of collaborative tasks. Furthermore, this work shows a knowledge extraction procedure designed to automatize the synthesis of Artificial Intelligence plan-based controllers for realizing a flexible coordination of human and robot behaviors in collaborative tasks. The generality of the ontological model and the developed representation capabilities as well as the validity of the synthesized planning domains are evaluated on a number of realistic industrial scenarios where collaborative robots are actually deployed.

Keywords: Ontology, Knowledge Representation and Reasoning, Human-Robot Collaboration, Automated Planning and Scheduling, Artificial Intelligence

1. Introduction

Nowadays, robots are successfully deployed in a large spectrum of real-world applications. Nevertheless, robots require increased level of autonomy and additional features to operate in "open environments" guaranteeing reliable and safe interactions. These constitute major scientific challenges and many research activities are ongoing to address them. Especially in manufacturing, the design of control systems capable of robustly dealing with the evolving needs and unpredictable changes of production requirements, is still an open research issue. In particular, an open problem for researchers is the construction of systems that can quickly adapt (and possibly anticipate) changes in the production requirements [1]. Higher levels of flexibility and adaptability of industrial robots are crucial to face the challenges of Industry 4.0 [2, 3]. Industry 4.0 [4] is indeed pushing manufacturing systems towards customer-oriented and personalized production, while trying to guarantee the advantages of mass production systems in terms of both productivity and costs [5]. Future manufacturing systems should in other words evolve towards flexible production supporting an "easy" and possibly automatic adaptation of production policies to changing needs, requirements and objectives of the factory [6].

Research in Human-Robot Collaboration (HRC) in particular pursues these challenging objectives by investigating the tight and symbiotic collaboration between human workers and autonomous (or semi-autonomous) robots. Novel production paradigms that see humans and robots working side-by-side as interchangeable resources, thus

*Corresponding author. E-mail: alessandro.umbrico@istc.cnr.it.

combining the precision and tirelessness of the former with the versatility and problem solving skills of the latter are crucial to boost flexibility of industrial processes [7, 8].

Classical control approaches usually rely on static models of robot capabilities and skills, and a static or hard-coded description of production requirements and objectives. Such technology is not fully able to support the level of autonomy necessary to dynamically adapt robot behaviors to the changing conditions and needs of future production environments. This is especially true in HRC where robots and humans share the working space and physically interact together to achieve common objectives. Human "actors" are neither predictable nor controllable, thus controllers should reliably deal with a significant source of *uncertainty* in order to safely interact with human workers and effectively support production [9]. It is crucial to propose control approaches implementing advanced *cognitive capabilities* so as to allow robots to achieve a higher level of *awareness* about itself, its "peer companions" and the production context with related dynamics e.g., production procedures, task requirements, needed skills and capabilities of actors taking part to production processes. In this regard, Artificial Intelligence (AI) studies technologies that are suitable to implement such cognitive capabilities. The tight integration of AI and Robotics [10, 11] would indeed allows (collaborative) robots to: (i) *perceive* the environment, correctly *interpret* occurring events and situations to properly build and maintain *knowledge* about the production context; (ii) *reason* about their own *capabilities/skills* and dynamically contextualize possible actions according to the *known* state of a production scenario and; (iii) *autonomously decide* how to act and interact with the environment and other "actors" (i.e., human operators but also other robots if necessary) in order to carry out (assigned) tasks and dynamically support production needs; (iv) *continuously adapt* robot behaviors over time according to "learned experience" and evolving production needs.

In this context, our long-term research objective is to enrich robot controllers with an AI-based "perceive-reason-act" paradigm implementing advanced cognitive features. The envisaged cognitive control approach would allow a collaborative robot to be *aware* about the actual state of the environment and autonomously decide needed tasks and how to execute them in order to realize the most suitable behavior supporting its human companion and the whole production. For example, the way an even consolidated production procedure is executed may depend on several factors. The device the worker needs to carry out some tasks could be damaged or not available. The resources like e.g., bolts or other pieces needed to perform some tasks could be not available or not sufficient. The worker joining the production today could be not skilled to perform certain task or may not know well all the procedure because of low experience. In all these cases (and other), a robot with an enhance awareness about the production would autonomously adapt its behavior and the execution of collaborative processes to the different situations/conditions detected. Suppose for example that the screwdriver a worker uses for some tasks is not available, the robot, *knowing* this *fact* and *knowing* that it can perform that tasks using its own screwdriver (i.e., the robot *knows* its capabilities), would autonomously synthesize a contextualized collaborative plan by assigning the execution of the tasks that require the screwdriver to itself and not the human worker. This level of autonomy and flexibility of the implemented collaborative process could not be achieved by classic control approaches without interrupting/stopping the production and manually intervene on the hard-coded/static control procedure of the robot.

To pursue this level of awareness and cognitive control, we investigate the integration of AI-based Knowledge Representation & Reasoning with Automated Planning and Execution. The combination of these AI technologies with Robotics has shown promising results in heterogeneous scenarios ranging from service and assistive robotics [12–15] to Reconfigurable Manufacturing Systems [16–20], improving flexibility of robot behaviors. Semantic technologies are crucial to realize *cognitive capabilities* and provide robot controllers with the *semantics* necessary to *represent* heterogeneous information coming from different sources (e.g., deployed sensing devices or domain expert knowledge about production processes) and *reason* about the resulting *knowledge* in order to (autonomously) *understand* the state of a production environment and make *contextualized decisions* accordingly. This paper advances a recent work extending an ontological model for Human-Robot Collaboration in manufacturing [21] and investigating the integration between Knowledge Representation & Reasoning and Automated Planning to enhance *awareness*, *adaptability* and *flexibility* of collaborative robots. On the one hand, the paper provides a refined and complete characterization of the developed context-based ontological model introduced in [21]. On the other hand, the paper proposes *knowledge reasoning mechanisms* that contextualize *production knowledge* and synthesize plan-based control models executing flexible robot behaviors while taking into account *human factor* and *uncertainty* [22, 23].

In particular the work emphasizes the use of *ontology design patterns* [24] as a mean to facilitate the description of HRC production processes by taking into account consolidated schema of interaction between humans and robots [25, 26]. Similarly to software design patterns, ontology design patterns are used to narrow knowledge design choices and define sufficiently general and reusable concepts characterizing behavior constraints of typical interactions between a human and a robot when performing collaborative tasks [25, 26]. Such patterns are thus suitable to characterize reusable behavioral constraints within collaborative manufacturing. Developed knowledge reasoning and knowledge extraction procedures automatically generate and validate planning models. Here ontological patterns are translated into sets of causal and temporal constraints that comply with the desired "shape" of the collaboration between the human and the robot. Such procedures are crucial to achieve the desired level of *awareness* and thus enable the automatic synthesis and online adaptation of (plan-based) control models and consequently the implemented policies (i.e., robot behaviors). AI-based planning and semantic technologies realizes *cognitive skills* suitable to enhance *autonomy* and *context awareness* of industrial (collaborative) robots and robustly deal with evolving production needs (e.g., changing production requirements, changing capabilities of a robotic platform or changing skills of human operators, etc.). The validity and generality of the proposed approach are evaluated on a number of realistic HRC scenarios, pilots of the EU H2020 research project Sharework [1]. These scenarios concern different types of production environment involving different production entities, tools, objects and procedures. The assessment shows that the developed ontology-based control approach effectively supports the definition of valid and complete *production knowledge* as well as the automatic synthesis of valid task planning models, concretely used to coordinate human and robot actions.

The paper is structured as follows. Section 2 discusses related works concerning the integration of knowledge reasoning and automated planning with robotics. It specifically highlights how other researchers are investigating the integration of the mentioned technologies to enhance the flexibility, adaptability and (social) context awareness of robot behaviors. Section 3 discusses the Sharework Ontology for Human-Robot Collaboration (SOHO) initially introduced in [21]. SOHO defines the representational space of the proposed approach and this section provides a complete and refined definition of its concepts and properties. Section 4 briefly introduces the timeline-based planning formalism and then describes the developed knowledge extraction procedure. This procedure is the central point linking knowledge defined according to SOHO and the task planning model used to synthesize collaborative plans. It supports the automatic and continuous updated and adaptation of the planning control model to the contextualized knowledge of a (collaborative) production scenario. Section 5 evaluates the proposed approach on a number of realistic scenarios taken from the pilot cases of the EU H2020 research project Sharework. On the one hand, the objective of this section is to show the generality of the ontological model and its capabilities of capturing all relevant aspects of different collaborative scenarios. On the other hand, the objective is to show the feasibility and correctness of the knowledge extraction procedure, generating valid timeline-based planning models in reasonable time. Finally, Section 6 summarizes the contribution of the paper pointing out possible directions for future developments.

2. Knowledge Representation and Reasoning in Robotics

Robotics and Artificial Intelligence (AI) are two research areas that historically addressed the challenge (among others) of building embedded intelligent systems capable of acting in a real-world environment from two different but synergetic perspectives [10]. Recent technological advancements in Robotics and AI are pushing towards the design and deployment of *autonomous robots* in increasingly complex/unstructured environments and "common-life" situations. However, if on the one hand technological advancements concerning the increased reliability and efficiency of sensing devices, manipulation and navigation skills of robots as well as solving and predictive capabilities of AI technologies, open new opportunities for the effective deployment of Robotics and AI solutions. On the other hand the increased complexity of application scenarios raises new and increasingly complex research challenges like e.g., moving from structured environments to semi-structured or unstructured environments. A tight integration

[1] https://sharework-project.eu

of Robotics and AI is crucial to enhance the autonomy and control capabilities of robots and allow them to safely and reliably *act* in the real-world [11, 27]. In particular, robots acting in the real-world should take into account a number of "non-functional" qualities that are crucial to realize behaviors that are safe and acceptable with respect to humans. Indeed, doing the *right thing* is not always sufficient when robots act in "social contexts". It is seamlessly important to do the *thing right* and thus reason about *how* a robot should behave when interacting or collaborating with humans [13, 28–30]. Robot controllers should therefore evolve towards an advanced "Perception, Reason, Act" paradigm implementing the cognitive capabilities needed to synthesize and execute flexible behaviors that are valid from both a technical and social point of view.

To implement the envisaged cognitive control paradigm, we found particularly promising the integration of ontology-based reasoning with AI-based planning and robot control. The integration of semantic technologies with robot controllers indeed has been widely studied in the literature [31]. Ontologies have been recognized as true enablers of adaptable and flexible systems compared to classic approaches [32, 33] and, consequently, have been exploited in the attempt to design more autonomous, flexible, adaptive and proactive artificial agents. Robot-integrated ontology-based reasoning has in particular shown effective results in the enhancement of robot flexibility and *awareness* [12, 16, 18]. This section discusses some relevant works in the literature concerning the integration of ontology-based knowledge reasoning, planning, and robotics. In particular, it aims at showing the enhanced flexibility and awareness obtained in different domains by robots, thanks to the integration of the mentioned technologies.

2.1. *Ontology in Robotics and Human-Robot Interaction Scenarios*

KnowRob [34, 35] is a well-known framework supporting advanced Perception, Reasoning and Control. The framework provides robots with a logical representation of a number of entities ranging from robotic parts and objects (with their composition and functionalities) to tasks, actions, and behaviors. This framework in particular focuses on manipulation tasks and allows robots to perceive objects of the environment, reason about their features and functionalities (e.g., formal description of affordances of objects [36]), and decide how to use them by synthesizing a suitable sequence of (STRIPS/PDDL) planning actions [37, 38] that accomplish some complex task [18, 39]. Although general KnowRob is mainly suitable to deal with scenarios where a single robot manipulates objects and interact with an environment executing sequences of actions in autonomy. However, the dyadic nature of HRC scenarios and the coexistence with a human actor requires reasoning on simultaneous execution of actions and the synergetic combination of robotic and human actors. Furthermore, reasoning about different human skills, preferences and performance as well as production requirements is necessary to contextualize production requirements and synthesize effective collaborations.

An ontological model characterizing object manipulation tasks of robots has been also considered within the PMK framework [17]. Similar to KnowRob 2 [34], PMK supports a "standardized" representation of the environment defining a "common language" to exchange information between a human and a robot. It also models sensory capabilities to perceive objects in the environment, linking perception outcomes to the ontological models of related objects. Reasoning capabilities allow a robot to evaluate observed/perceived situations (i.e., configurations of the environment in terms of available objects and their features) and enhance the synthesis of decisions on manipulation tasks. An interesting framework concerning the ontological description and integration of robotic skills is SkiROS [40]. Less general than KnowRob, this framework is designed on ROS with the objective of proposing an ontological model of robot skills. On top of this knowledge, action-based planning supports a dynamic combination of skills to realize complex behaviors. This work proposes the use of ontology as a meta-language allowing users to describe robotic skills (knowledge engineering) and integrate available skills into a uniform cognitive control framework based on ROS. Similar to KnowRob [35], SkiROS focuses on the description/control of a single robot acting in the environment. Concerning HRC scenarios, this framework does not support an explicit representation of the skills of multiple agents sharing the same work environment and production objectives. It does not support reasoning on human and robot behaviors taking into account concurrency, time, production constraints, and controllability issues.

The ORO framework [41] develops a knowledge reasoning framework endowing robots with common sense reasoning capabilities to autonomously operate in semantically-rich human environments. With respect to KnowRob the ORO framework addresses the control problem from a cognitive perspective and realizes a general cognitive ar-

chitecture deployed on different robotic platforms and assessed on different cognitive scenarios [41]. This architecture has been specifically developed to support advanced cognitive skills (e.g., *theory of mind* capabilities) and thus support increasingly flexible and adaptive human-robot interactions [12]. Considering an HRC perspective, ORO pursues a turn-based interaction approach where the human and the robot are supposed to perform an action at a time with the robot "reacting" to the observed behavior and inferred state of the human. This interaction mechanism is not fully effective in a production scenario where efficiency and safety play a central role. Indeed, in such a context it is necessary to take into account (possible) simultaneous behaviors of the human and a robot and accordingly decide the best actions to execute in order to achieve shared production goals. Furthermore, the behavior of a human directly (or indirectly) affects the decisions of the robot, constituting to a significant source of uncertainty (the actual behavior of a human worker is not controllable). Robot controllers should deal with this source of uncertainty in order to reliably carry out collaborative tasks.

Concerning human-robot social interactions, knowledge representation and reasoning have been used to realize socially compliant robot behaviors. Non-functional requirements like those regarding social norms are crucial to realize acceptable behaviors [29] and effective human-robot interactions. For example, the work [13] uses knowledge reasoning to represent social norms and allow a social robot to implement socially acceptable behaviors for social tasks. More specifically, the work proposes a formal description of the functional affordances of objects to reason about their possible use and thus infer those that are suitable to accomplish the requested social task (i.e., serving coffee to guests using the right object). This is just one example about a possible use of knowledge reasoning (and its integration with planning) to synthesize flexible and socially-aware robot behaviors. The work [30] proposes the use of knowledge reasoning to adapt human-robot interactions to the cultural knowledge of different contexts and people. This is another example of how ontology-based reasoning can enhance *context awareness* of robots by implementing some cognitive capabilities. In this case, such capabilities are used to reason about non-functional qualitative aspects of human-robot interactions and synthesize socially-compliant and acceptable behaviors. Another example of work investigating the use of ontology in social robotics is [15]. Similar to other works e.g., KnowRob [35], it proposes an ontological model characterizing users, the interacting environment, capabilities (not necessarily correlated to the robot only) and tasks. On top of this model the work instantiates a cognitive architecture realizing a perceive, reason, act control loop. The resulting "cognitive agent" incrementally decides which social task to perform according to the perceived state of the interaction context. An added value of the work [15] is the explicit representation of interacting users through *user profiles* that support the realization of *personalized* and user-oriented services.

Ontology-based representation and reasoning has been used also in medical scenarios. The work [42] proposes the use of ontology in orthopedic surgery. It uses the ontological model OROSU to uniformly represent domain knowledge to different types of user e.g., surgeons, nurses, or technicians, during surgery. In particular, it relies on the KnowRob framework [35] to support the description of surgical procedures concerning Hip Surgery. An ontological model characterizing object manipulation tasks of robots has been also considered within the PMK framework [17]. It supports a "standardized" representation of the environment defining a "common language" to exchange information between a human and a robot. Taking into account other frameworks e.g., KnowRob [35], PMK well characterizes causal information and constraints about manipulation tasks and defines knowledge that is useful at both task and motion planning level. This framework realizes a knowledge-based approach to Integrated Task and Motion Planning (TAMP) problems and it enables reasoning capabilities that allow a robot to synthesize enhanced manipulation tasks, from perceived situations (i.e., configurations of the environment in terms of available objects and their features). The work [43] exploits a robot knowledge framework (OMRKF) consisting of a series of ontology layers, including a robot-centered and human-centered ontology. The system in particular relies on an object layer, a context layer and an activity layer realizing abstraction capabilities of gathered sensory data. It however lacks of foundational background which limits the reliability of inferred knowledge as can be see with the concept of "living room" which is classified as a space region and not as the "role" of the region (the problem becomes clear by observing that a region of space is fixed while the living room can be located in different parts of the building at different times). Furthermore, it does not distinguish between activity and functionality resulting in a rigid characterization of robot capabilities and behaviors. For example "avoid obstacle" is not a behavior but a function which can be implemented by different behaviors like, e.g., moving away or turning around.

Ontology-based reasoning has been used also to formally represent normative standards and evaluate compliance with them. An example is the work [44] where normative standards characterizing indoor environmental qualities have been encoded into an ontological model. A social robot has been endowed with cognitive capabilities to ground detected quality conditions of an environment and automatically evaluate the compliance of a perceived environment to the normative standards. The work [45] proposes a semantic approach to the Internet of Robotic Things (IoRT). A production rule language called SmartRules is developed to support context-aware management and development of IoRT applications. Specifically, an ontology model abstracts sensing and actuator functionalities of "connected devices" that are part of an environment. Production rules then define general semantics to contextualize gathered information and recognize events/situations concerning the environment. The approach has been evaluated in an Ambient Assisted Living scenario to monitor and assist elderly persons during their daily-home living activities.

2.2. *Ontology in Manufacturing*

In manufacturing, ontologies have mainly focused on the manufacturing system as a whole or rather on specific production aspects like e.g., [46–48]. A formal model describing procedures, capabilities of working entities and their possible interactions to support (agile) production objectives is challenging. The description of so-called Cyber-Physical Systems (CPSs) like for example a Human-Robot Collaborative scenario, requires to model complex adaptive behaviors of involved agents from both a "local perspective" (i.e., the point of view of a specific agent) and "global perspective" (i.e., the point of view of the production and related constraints and objectives) [49].

Ontologies have been applied wit the aim of increasing flexibility in modeling and planning of, e.g., mechatronic devices [50], resources in collaborative environments and the whole enterprise [51], collaborative robots [52] and navigation robots [53]. [50] uses a fixed ontology to collects static and dynamic information relative to a robot. The basic actions of a robot are hard-coded but the ontological system, which includes a PDDL ontology where knowledge about actions is stored, adds some flexibility like the possibility to learn articulated actions and to act with partial information, e.g., information about the location of the object to move. In this application, beside the lack of functionality/activity distinction, the robot has very limited knowledge of the environment. [51] describes an ontological model aimed to represent the resource capabilities for the development of products and processes. The model is part of a larger ontology for collaborative and integrated development of products, processes and resources. The work [18] uses a well structured ontological model to characterize product assembly tasks. Specifically, the work extends the KnowRob framework [35] by integrating inference rules necessary to reason about incomplete assembles of different products. Based on the outcome of the implemented perception and reasoning processes they automatically plan the next action to be executed and incrementally assemble the desired products. The work [54] uses ontology to represent kit building parts for assembly operations. Kit building parts are presented by means of XML descriptions whose schema (XSDL) is mapped to an ontological model providing a uniform logic-based representational space. The ontological model and the automatic generation of OWL descriptions from XML schema are then used within an agility framework to evaluate the agility performance of robotic systems.

Concerning collaborative manufacturing, the work [55] proposes an ontological model called OCRA which takes into account uncertainty and safety constraints. Interestingly, it characterizes reliable human-robot collaborations providing robots with a well structured formalization suitable to reason about the "execution perspective" of their plans. In this regard, OCRA characterizes collaborative scenarios from a perspective that is different but compatible with the ontological model considered within the current work [21]. The work [19] proposes an ontological approach to the representation of collaborative working environments supporting agile production. It realizes an ontology-based multi-agent system integrated with a Business Rule Management System to define a language to coordinate human and robotic agents. The defined language is thus used inside a multi-agent framework to represent and share information about a collaborative environment (e.g., information about the cobot, the worker, the different products, and assigned tasks). The proposed ontological model does not rely on a structured theoretical background in which the knowledge about tasks and agents capabilities is hard-coded. It proposes a limited and schematic description of a collaborative environment. For example workers and cobots are represented with a simple schema describing just their location within the environment, no additional information about their capabilities, composition, or behavioral features is provided. Furthermore, it defines domain-specific concepts that describe tasks without providing a general and structured characterization of related properties and compositional dependencies, relevant for their execution.

Works within the ROSETTA project [56] have also investigated the use of ontology-based representation and reasoning to simplify the programming of industrial robots and facilitate the interaction with humans in manufacturing. The work [57] uses and extends the SIARAS ontology to characterize knowledge about robot skills focusing in particular on manipulation skills and related devices that may compose a manufacturing environment (e.g., gripper, fixture, the robot itself). The presented knowledge integration framework offers services that uniformly describe knowledge about robot manipulation scenarios (sensing and manipulation devices, robot skills, risks) and facilitate the programming of robots for the execution of such tasks. It aggregates several ontological models that are relevant also from a human-robot collaboration perspective e.g., the *injury.owl* which characterizes the expected risk level of injury when a robot cooperates with a human or shares the same environment. However, the resulting ontological model does not rely on a well-structured theoretical background and is characterized by a narrow scope strictly focused on robot-centered manipulation tasks. The supported description of tasks is for example mainly syntactic, the model does not support a structured description of production procedures with related requirements and constraints. The model is thus not suitable to capture the complexity of more general scenarios entailing for example a peer interaction of agents that jointly perform tasks and achieve shared production objectives.

3. An Ontology for Human-Robot Collaboration

Several researchers have investigated the integration of ontology with Robotics and other AI technologies to enhance the *qualities* of robot behaviors in terms of *awareness*, *flexibility*, *reliability*, *safety* as well as *trustworthiness* and *acceptability*. Such qualities are fundamental for the effective deployment of autonomous robots in real-world scenarios. This is especially true in collaborative manufacturing systems or more in general cyber-physical manufacturing systems requiring a tight and continuous interaction between robots (or artificial autonomous agents) with humans. In this regard, an ontological model capable of capturing capabilities of different types of resource, actors with different interacting features, production and behavioral requirements is still missing. Such an ontology and related semantics would be crucial to allow future manufacturing systems to: (i) increase their level of *awareness* about the state and needs of a production environment; (ii) autonomously interpret production events and properly coordinate “active resources" like e.g., collaborative robots and workers and; (iii) dynamically adapt “implemented processes" to different skills and *working/health conditions* of collaborating workers and changing production needs.

The work [21] made a first step towards the definition of such an ontological model specifically designed for Human-Robot Collaboration (HRC). SOHO (*Sharework Ontology for Human Robot Collaboration*) is a *domain ontology* [58] aiming at characterizing HRC scenarios from different but synergetic levels of abstraction (contexts). The objective is to define a well-structured model of production environments, human, machine and robot structures, capabilities and functional operations. The ontological model pursues a flexible interpretation of these concepts in order to dynamically contextualize and interpret production states/situations according to the specific needs of processes and features of the environment. For example, the *acting qualities* of “productive actors" e.g., a human worker or a collaborative robot, can be automatically inferred by *contextualizing* their operational capabilities and the *functional operations* that can be performed in the manufacturing domain (i.e., *functions* as introduced by [59]). Operational capabilities (or simply capabilities) of robots and workers intrinsically depend respectively on their structures (e.g., actuators and end-effectors that are part of the robotic device - *embodiment*) and their skills or abilities (e.g., a worker can perform some welding operations only if she is skilled in that task). These capabilities combined with the specific needs and requirements of a production environment would then enable the execution of different (instances of) functions [59]. This section provides a complete and refined description of SOHO [2], defined using Protégé [3] and the OWL language [60]. Concerning the work [21], this section better describes the correlation between SOHO and the foundational level in order to clarify the organization of the knowledge and the resulting interpretation of collaborative scenarios.

[2]The ontology is publicly available at the following GitHub repository - https://github.com/pstlab/SOHO.git.
[3]https://protege.stanford.edu/

3.1. Foundations and Contexts

Foundational ontologies aim at describing reality from a high-level perspective in order to define concepts that are general enough to be valid across many domains. Their use is generally recommended and represents a good design choice in order to base new (more specific) ontological models on a well-structured semantics. These ontological models constitute a stable theoretical background of more specific ontologies and thus foster a clear structuring and disambiguation of domain concepts [61, 62]. Several foundational ontologies have been introduced in the literature like e.g., BFO [63], DOLCE [64] or SUMO [65]. Among these, SOHO relies on DOLCE [64] in order to support a flexible interpretation of temporally evolving entities and, also, to rely on a recognized standard representation framework (ISO 213838-3). DOLCE therefore represents a flexible model, well suited to support the interpretation of domain entities whose state depends on the context and may change/evolve over time.

In addition to DOLCE, SOHO is built on top of two other ontologies: (i) the CORA ontology [66] and; (ii) the SSN ontology [67]. CORA is an IEEE standard ontology for robotics and automation aiming at promoting a common language in the robotics and automation domain. It characterizes knowledge about robots and robot parts, robot positions and configurations and groups of robots. This standard relies on SUMO [65] as theoretical foundation and integrates the framework ALFUS [68] to define possible autonomy levels and related operative modes of a robot. SSN is a W3C standard ontology for IoT devices and sensor networks. It defines basic concepts and properties characterizing the capabilities of sensing devices, their deployment into a physical environment and the outcome of sensing processes. SSN relies on DOLCE and defines a sufficiently general model to represent physical properties of an environment and physical entities that can be observed or monitored over time.

Both CORA and SSN define concepts and properties relevant to HRC but they are not sufficient to describe production procedures and possible collaborations needed between human and robot agents. The scope of SSN is limited to the characterization of a physical environment in terms of properties that can be observed and sensing devices that carry out “sensing processes". This ontology is quite “self-contained" and can be easily integrated with CORA to represent also robot interfaces and sensing parts. CORA instead has a broader scope. It focuses on robot parts, robot configurations and levels of autonomy. However, CORA does not support the contextualization and interpretation of behaviors of robots and other autonomous agents (e.g., human operators) with respect to the global production objectives and processes. Considering HRC scenarios, CORA does not provide constructs characterizing properties and skills of human workers and their “agentive behaviors". It therefore would not support an effective description of collaborative processes specifying possible interactions between robots and humans and how they could jointly achieve common (production) goals.

Before entering into the details of SOHO next subsection describes the most relevant concepts and properties inherited from DOLCE. A clear description of the theoretical background and underlying assumptions is crucial to introduce new concepts defined by SOHO and explain how collaborative scenarios are interpreted.

3.1.1. Qualities, Norms and Events

Human-Robot Collaboration scenarios are a combination of *technical*, *physical* and *social contexts* since the acting entities should physically interact while complying with a number of *rules* that guarantee a correct and safe execution of production processes. Indeed, a HRC scenario is composed by a number of *physical entities* each characterized by different *features* and *qualities* that cooperate to achieve common (production) objectives. We therefore interpret HRC scenarios as *social contexts* where the behavior of each acting entity affects the behavior of others and coordination is necessary to correctly and safely carry out production processes. As such, any HRC scenario is subject to social structures known as norms [69] either implicit or explicit *rules*, that constrain the behavior of involved actors.

To model concepts and properties that suitably capture such dynamics SOHO relies on the DOLCE+DnS Ultralite ontology (DUL) [4] which is a lightweight version of DOLCE suitable to model either physical or social contexts. DUL uses simplified constructs to represent temporal and spatial relations and supports a modular, pattern-based, structures (*content ontology design pattern*).

[4] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite

The concept DUL:Object models any physical, social or mental object or a substance of the domain. SOHO in particular considers the sub-concept DUL:Agent and DUL:PhysicalObject to characterize respectively acting entities of the domain (i.e., collaborative robots and human workers) and “passive" physical elements that are part of a production environment (e.g., tools, robot parts, sensing devices, production resources etc.). The concept DUL:Agent characterizes any agentive object either physical (e.g., a robot) or social (e.g., an institution) that behave according to some logic/algorithm. The concept of DUL:Agent is equivalent to the concept DUL:PhysicalAgent that is a particular type of DUL:PhysicalObject which is in turn any object associated with a space region. The concept DUL:PhysicalObject is thus useful to characterize the structure of a production environment and the types of object that could compose it. Each DUL:PhysicalObject is described by a set of *attributes* that characterize its specific features. DUL supports a flexible representation of such attributes and the way they can be measured and expressed through the distinction of DUL:Quality and DUL:Region.

According to the documentation, a DUL:Quality is a any aspect of an entity (e.g., a DUL:PhysicalObject) that cannot exists without that entity. However, a quality is not part of an entity. Rather it represents a particular attribute/aspect that is relevant to be expressed in the considered domain. The physical location, the shape or the color of the surface of a DUL:PhysicalObject are examples of possible DUL:Quality. For each DUL:Quality there can be one or more DUL:Region expressing the *value* of the associated quality. Namely, a DUL:Region is any *dimensional space* which can be used as a value for a quality. Examples of (general) regions available within DUL are DUL:TimeInterval and DUL:SpaceRegion that are used to represent respectively *time* and object location, with respect to a particular dimensional space. Next sub-sections will show with further details how SOHO uses these classes (i.e., DUL:PhysicalObject, DUL:Quality and DUL:Region) to characterize different types of physical entities that can be part of a HRC.

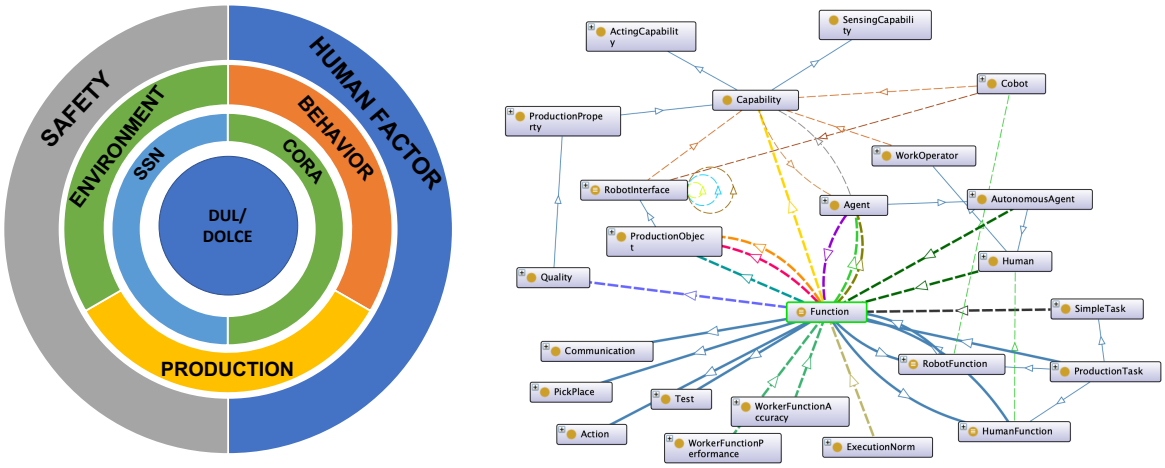
Another relevant concept used by SOHO is DUL:Description and its sub-concepts DUL:Method, DUL:Goal and DUL:Norm. A DUL:Description is defined as a DUL:SocialObject representing a conceptualization. According to the documentation, it can be thought also as a “descriptive context" that create a view of a “relational context" out of a set of data or observations. SOHO uses this concept to conceptualize production goals, procedures and related constraints. In this regard, a DUL:Method is a DUL:Description that defines concepts to guide carrying out actions aimed at a solution with respect to a problem. It is worth noticing that a DUL:Method is different from a DUL:Plan since plans could be carried out in order to follow a method while a method can be followed by executing different plans. This concept is therefore well suited to characterize *production procedures* that can be instantiated into different collaborative plans entailing the execution of human and robot actions. A DUL:Goal is the description of a DUL:Situation that is desired by an DUL:Agent and usually associated to a DUL:Plan describing how to actually achieve it. SOHO extends this concept to model *social goals* and thus describe (production-related) situations that would be jointly achieved by more than one agent (i.e., a human and a robot in the specific case of a HRC production scenario).

As mentioned at the beginning of the section, SOHO interprets a HRC scenario as a social context where two or more agents should cooperate to achieve a common objective (SocialGoal). Involved agents should therefore comply with a number of *rules* in order to satisfy requirements concerning for example safety, procedure consistency, production quality etc. To describe such rules SOHO relies on the concept DUL:Norm which generally represents *social norms*. In particular, SOHO distinguishes between two types of norms: (i) norms determining how human and robot agents should behave while carrying out joint tasks (i.e., collaborative tasks) and; (ii) norms determining how tasks should be executed to satisfy production requirements. Next sub-sections describe with further details how the concept DUL:Norm is specialized within SOHO.

Finally, the concept DUL:Event represents any physical, social or mental process that can occur in a domain. SOHO uses the concept DUL:Event to characterize situations that can occur and be observed within the environment, and that can affect production. Namely, this concept is used to define exogenous events that should be taken into account during the life-cycle of a collaborative production cell and that may require the adaptation of implemented production procedures and collaborative plans. The concept DUL:Action represents a particular type of DUL:Event with at least one participating DUL:Agent. SOHO specifically uses this concept to represent actions physically executed either by a human, by a robot or jointly by both of them. An action is thus seen as a temporal occurrence (i.e., instance or *implementation*) of a DUL:Description about a task/function of a production procedure.

3.1.2. *Integrating Synergetic Perspectives*

Ontologies should be adequate to their domains, and domains come along on different granularity levels [61]. An ontology should therefore account for all perspectives and levels of abstraction that are relevant for a domain. SOHO follows a context-based approach and organizes knowledge in a number of synergetic contexts, each describing the domain from a particular perspective. Namely, the contexts support modularity and *multi-perspective* representation of domain knowledge. Figure 1 shows the general structure of SOHO: (i) the *Environment Context*; (ii) the *Behavior Context* and; (iii) the *Production Context*. The *safety* and *human factor* contexts do not represent actual ontological contexts. Rather they define two "meta-perspectives" that must be uniformly considered at different levels of abstraction by all ontological contexts.



(a) (b)

Figure 1. Overview of SOHO: (a) general structure and defined contexts; (b) excerpt of concepts and properties

Environment Context. The environment context defines physical elements and general properties of an environment that can be observed. This context strongly relies on SSN which is crucial to characterize the sensing capabilities of available devices and the physical properties of domain entities they can observe. First, SOHO defines a concept to model objects that are part of a production environment by extending the concept `DUL:PhysicalObject`.

$$\texttt{ProductionObject} \sqsubseteq \texttt{DUL:PhysicalObject} \sqcap \exists\, \texttt{DUL:hasQuality.ObjectQuality} \tag{1}$$

where `ObjectQuality` is a subclass of `ProductionProperty` which in turn is a subclass of `DUL:Quality`. The concept of `ObjectQuality` thus describes the attributes of any `DUL:PhysicalObject` that specifically compose a production environment (i.e., a `ProductionObject`).

Some properties of the objects that constitute a production environment can change over time and it could be necessary to observe/monitor them in order to correctly carry out tasks and adapt production processes if necessary. Examples are the position in space of an object necessary to perform a task, the position and occupancy state of an area where to place an object or the state of a bolt etc. This kind of data position in space, orientation, weight, etc., and the kind of data that can be gathered through their observation. SOHO relies on SSN to model sensing devices and the information they can gather through (implemented) sensing processes. SSN defines the concept `SSN:FeatureOfInterest` which generally describe aspects of an environment (e.g., properties of objects of a production environment) that are interesting to be observed through some sensing device. Observable properties may change according to the specific features (i.e., `DUL:Quality`) of the objects that compose the environment but also according to the sensing capabilities of available devices as well as their deployment. The "observability" of one or

more quality of an object actually depends on the sensing capabilities of available devices and their deployment in the production environment. To characterize the *observable properties* of a production environment SOHO therefore extends SSN by leveraging the concept `DUL:Role`. A `DUL:Role` is a concept used to *classify* an object of the domain and is thus useful to support a flexible classification of `ProductionObject` that can be actually observed in a production environment. SOHO defines the concept `ObservableFeature` as a `DUL:Role` that objects can play according to the available sensing devices.

$$\begin{aligned}\texttt{ObservableFeature} \sqsubseteq\ & \texttt{DUL:Role} \sqcap \texttt{SSN:FeatureOfInterest} \sqcap \\ & \exists\, \texttt{DUL:isRoleOf.ProductionObject} \sqcap \\ & \exists\, \texttt{hasObservableProperty.ProductionProperty} \sqcap \\ & \exists\, \texttt{isObservableThrough.SSN:Sensor}\end{aligned} \tag{2}$$

The design choices of defining `ObservableFeature` and the property `hasObservableProperty` emphasizes the decoupling between the intrinsic properties of an object and the ones that can actually be observed. For example, human tracking processes can be realized only if the considered working environment is endowed with sensing devices capable of observing physical properties of a human worker like e.g., physical position, posture, motion speed and direction, etc. This distinction allows knowledge reasoning processes to dynamically infer the actual *perception capabilities* of a production scenario by taking into account available sensing devices.

$$\begin{aligned}&\texttt{ProductionObject(o)} \wedge \\ &\texttt{DUL:hasQuality(o, p)} \wedge \\ &\texttt{SSN:Sensor(s)} \wedge \\ &\texttt{SSN:observes(s, p)} \\ &\qquad\qquad \rightarrow\ \texttt{ObservableFeature(x)} \wedge \\ &\qquad\qquad\quad\ \texttt{DUL:isRoleOf(x, o)} \wedge \\ &\qquad\qquad\quad\ \texttt{hasObservableProperty(x, p)} \wedge \\ &\qquad\qquad\quad\ \texttt{isObservableThrough(x, s)}\end{aligned} \tag{3}$$

To support this reasoning, SOHO enforces the equivalence between the concept `ProductionProperty` and `SSN:Property`. Since a `ProductionProperty` is subclass of `DUL:Quality`, SOHO interprets `SSN:Property` as qualities of the objects composing the environment.

Behavior Context. The behavior context extends the environment context to characterize the behaviors of the acting entities of a production environment. The central concept of this concept is `DUL:Agent` as the physical entities that actually *act* in the environment and carry out production processes. Given the focus on Human-Robot Collaboration, SOHO distinguishes two particular types of `DUL:Agent` that are: (i) `Cobot` and; (ii) `Worker`. SOHO interprets both agents as two physical *autonomous agents* that are associated with an *embodiment* which is in turn associated with a number of physical and behavioral qualities. The two concepts differ in terms of the specific types of *physical objects* that compose their embodiment, associated qualities that can be observed/monitored and related capabilities. More specifically, a `Cobot` is defined as follows:

$$\begin{aligned}\texttt{Cobot} \sqsubseteq\ & \texttt{DUL:Agent} \sqcap \\ & \exists\, \texttt{DUL:hasPart.RobotInterface} \sqcap \\ & \exists\, \texttt{DUL:hasQuality.(Autonomy} \sqcup \texttt{Capability} \sqcup \texttt{RobotProperty)}\end{aligned} \tag{4}$$

The role `DUL:hasPart` associates a robot with a set of `RobotInterface` representing its *embodiment* i.e., the set of `PhysicalObject` composing the physical device. Being physical objects, each `RobotInterface` is associated with a number of `DUL:Quality` representing relevant attributes that can be measured and/or observed (`RobotProperty`). Let us consider for example a wheeled base *wb*1 as a possible individual of `RobotInterface` being part of a collaborative robotic agent *cob* (individual of `Cobot`). The individual *wb* would then be associated for to qualities `SpatialLocation` and `RobotSpeed` describing respectively the known geometric position and motion speed of the wheeled base *wb* (and consequently of the cobot *cob*).

A particular type of DUL:Quality is Capability. SOHO defines the concept Capability to characterize which types of operations a DUL:Agent can support through its “functional parts". Namely, such qualities characterize general operations like e.g., GraspingObject, UseTool or MoveObject that an acting entity can intrinsically perform according to its physical/technical composition. A Cobot inherits the set of Capability supported by its parts. Let us consider for example a robot gripper *grip* as a possible individual of RobotInterface being part of *cob*. The description of *grip* would associate the individual with the capability GraspObject. Consequently, the robot *cob* itself would be associated with the same capability given its internal composition. In other words, a robot would be capable of grasping objects if and only if the agent is endowed with a physical interface capable of grasping objects (i.e., a *gripper*). Another particular type of DUL:Quality is Autonomy which represents a specific *behavioral quality*. SOHO defines behavioral qualities of DUL:Agent introducing the concept AgentBehavior as subclass of DUL:Quality. This concept is then specialized into Autonomy and WorkerLevel to characterize knowledge about the expected behavior of acting agent, respectively robots and workers. In the case of the robot for example the quality Autonomy is expressed in a number of AutonomyLevel (subclass of DUL:Region) structured according to the ALFUS framework [68]. Different levels of autonomy of a robot in general would determine different operation modalities and different safety constraints that should be considered when performing production tasks. It could be the case, for example, that some tasks can be performed only if a robot can operate in FullyAutonomy. Also, the production procedure implemented when a robot works in Teleoperation could be different from the procedure implemented when a robot works in SemiAutonomy of FullyAutonomy. A proper characterization of this information is useful to “parametrize" production procedures and support the conditions under which different procedures and/or differet types of tasks can be actually executed in a HRC scenario.

Similar to Cobot a human agent Worker is defined as follows:

$$\begin{aligned} \texttt{Worker} \sqsubseteq\ & \texttt{DUL:Agent} \sqcap \\ & \exists\, \texttt{DUL:hasPart.HumanBodyPart} \sqcap \\ & \exists\, \texttt{DUL:hasQuality.(WorkerLevel} \sqcup \texttt{Capability} \sqcup \texttt{HumanBodyProperty)} \end{aligned} \tag{5}$$

The general assumptions and structure described for Cobot holds also in the case of Worker. SOHO distinguishes these two concepts with respect to the specific types of DUL:Quality associated through their embodiment. Concerning behavioral qualities, SOHO introduces WorkerLevel to describe the expertise level of a worker. This information is useful to adapt production procedures and the type of support provided by a collaborative robot according to the known level of knowledge of a worker about a procedure. Concerning physical qualities, SOHO introduces the concepts HumanHealthProperty and HumanPhysicalProperty. The former supports the representation of health parameters like e.g., MentalStatus and PhysiologicalStatus that are useful to consider possible known impairments of workers and/or to monitor health parameters (e.g., heart rate, cognitive stress, etc) that may influence their performance, safety and health state.

It is worth noticing that SOHO pursues a flexible interpretation of behavioral capabilities and behavioral qualities of acting entities. Behavioral knowledge is indeed determined according to the specific features and internal composition (i.e., the *embodiment*) of the specific agents that join a production environment. In this way, the set of operations that can be actually implemented in a particular scenario (and “who" can implement them) can be dynamically inferred according to known agents’ capabilities. To support this reasoning it is necessary to characterize general production operations of manufacturing environments and correlate them to capabilities necessary for their correct execution. To this aim, SOHO integrates the *Taxonomy of Functions* defined in [59] which characterizes low-level operations that can be performed in general manufacturing environments. The taxonomy characterizes such low-level operations as Function and classifies them according to the *effects* they have on the DUL:Quality of *target* objects. SOHO extends this interpretation of functions by taking into account also the capabilities they

require for the execution and proposes the following definition:

$$\begin{aligned}\texttt{Function} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\, \texttt{canBePerformedBy.DUL:Agent} \sqcap \\ & \exists\, \texttt{hasEffectOn.DUL:Quality} \sqcap \\ & \exists\, \texttt{requires.Capability} \sqcap \\ & \exists\, \texttt{hasTarget.ProductionObject}\end{aligned} \tag{6}$$

A `Function` is classified as a particular type of `ProductionTask` and is characterized according to the needed `Capability` and the affected `DUL:Quality` of the target `ProductionObject`. The concept `ProductionTask` is a particular type of `DUL:Method` SOHO uses to describe production procedures. This and other correlated concepts are described with more details in the next sections.

$$\begin{aligned}&\texttt{ProductionObject(o)} \wedge \texttt{Function(f)} \wedge \\ &\texttt{DUL:Agent(a)} \wedge \texttt{DUL:hasQuality(o, q)} \wedge \\ &\texttt{hasEffectOn(f, q)} \wedge \texttt{hasCapability(a, c)} \wedge \\ &\texttt{requiresCapability(f, c)} \\ &\qquad\qquad \rightarrow \texttt{hasTarget(f, o)} \wedge \\ &\qquad\qquad\quad \texttt{canBePerformedBy(f, a)}\end{aligned} \tag{7}$$

This logic formula shows the kind of reasoning supported by SOHO. The propose semantics and the explicit representation of agents' capabilities and their correlation with general (known) functions supports the contextualization of *behavioral dynamics* of production scenarios. Specifically, following Equation 7, it is possible to infer the instances of `Function` that `DUL:Agent` can actually perform by taking into account associated `Capability`.

Production Context. Considering the production perspective, SOHO defines concepts and properties that characterize production procedures in terms of objectives and operations humans and robots should perform to successfully achieve them. A proper representation of this knowledge is crucial to support collaborative processes and establish human and robot *commitment* to production goals [70, 71] and a level of *agreement* about the way they achieve these goals [72, 73]. Furthermore, it is necessary to characterize *events* that may occur in a production environment and that are relevant with respect to the execution of production procedures. SOHO introduces production-related knowledge by leveraging the foundational concepts `DUL:Event` and `DUL:Description`. They are mainly used to distinguish between general descriptions of the procedures and constraints necessary to achieve production objectives (i.e., sub-classes of `DUL:Description`) and the temporal occurrences implementing such procedures (i.e., sub-classes of `DUL:Event`).

SOHO defines the concept `ProductionGoal` as a particular type of `DUL:Goal` to characterize situations that acting agents desire to achieve in order to satisfy production requirements. Each `ProductionGoal` is associated with (at least) one `ProductionMethod` which is a particular type of `DUL:Method` describing valid procedures.

$$\begin{aligned}\texttt{ProductionGoal} \sqsubseteq\ & \texttt{DUL:Goal} \sqcap \\ & \exists\, \texttt{DUL:hasConstituent.ProductionMethod}\end{aligned} \tag{8}$$

Goals are achieved through plans each composed by a number of actions implementing a particular production method. SOHO defines the concept `ProductionPlan` to describe the way a certain `ProductionGoal` is achieved. Specifically, a `ProductionPlan` describes a particular `ProductionProcess` which implements a particular `ProductionMethod` through the actual execution of a number of `ProductionAction`.

$$\begin{aligned}\texttt{ProductionPlan} \sqsubseteq\ & \texttt{DUL:Plan} \sqcap \\ & \exists\, \texttt{DUL:hasComponent.ProductionGoal} \sqcap \\ & \exists\, \texttt{DUL:isDescribedBy.ProductionMethod} \sqcap \\ & \exists\, \texttt{DUL:describes.ProductionProcess}\end{aligned} \tag{9}$$

where a ProductionProcess is the description of a dynamic (physical) event involving the execution of a number of actions by participating agents. SOHO defines this concept as a particular type of DUL:Process which is in turn a particular type of DUL:Event.

$$\begin{aligned}\texttt{ProductionProcess} \sqsubseteq\ & \texttt{DUL:Process} \sqcap \\ & \exists\, \texttt{DUL:hasPart.(ProductionAction} \sqcup \\ & \qquad \texttt{ProductionRelatedEvent)} \sqcap \\ & \exists\, \texttt{DUL:hasParticipant.DUL:Agent} \sqcap \\ & \exists\, \texttt{DUL:isDescribedBy.ProductionPlan}\end{aligned} \tag{10}$$

Considering the HRC domain, a particular type of ProductionProcess is CollaborativeProcess where exactly one autonomous robot (Cobot) and one human worker (WorkOperator) jointly contribute.

The temporal and physical occurrence of production processes entail the execution of a number of ProductionAction and the occurrence of a number of associated ProductionRelatedEvent. SOHO defines a ProductionAction as a particular type of DUL:Action describing the actual execution in time and space of production operations (e.g., an instance of a Function) by a Cobot or a WorkOperator (i.e., a DUL:Agent).

$$\begin{aligned}\texttt{ProductionAction} \sqsubseteq\ & \texttt{DUL:Action} \sqcap \\ & \exists\, \texttt{DUL:hasParticipant.DUL:Agent} \sqcap \\ & \exists\, \texttt{DUL:isDescribedBy.(DUL:InteractionModality} \sqcup \\ & \qquad \texttt{ProductionTask)}\end{aligned} \tag{11}$$

SOHO defines the concept ProductionRelatedEvent and EnvironmentRelatedEvent as more general type of DUL:Event describing respectively events resulting from the execution of an action (e.g., execution failures or results) and exogenous events concerning the state of the *observable* features of the environment.

Actions of a production process represent the actual execution of a procedure. The concept ProductionMethod is central to the description of procedures. It is a particular type of DUL:Method resulting from the composition of ProductionTask. Each production tasks characterizes a particular operaiton at a certain level of abstraction.

$$\begin{aligned}\texttt{ProductionTask} \sqsubseteq\ & \texttt{ProductionMethod} \sqcap \\ & \exists\, \texttt{DUL:isDescribedBy.ProductionNorm}\end{aligned} \tag{12}$$

where the concept of ProductionNorm is a particular type of ExecutionNorm which is a sub-class of DUL:Norm. This concept is used to describe general rules that determine the way tasks (and resulting actions) should be executed by the participating agents in order to correctly achieve associated goals.

$$\begin{aligned}\texttt{ProductionNorm} \sqsubseteq\ & \texttt{ExecutionNorm} \sqcap \\ & \exists\, \texttt{DUL:describes.ProductionTask} \sqcap \\ & \exists\, \texttt{constrains.ProductionTask}\end{aligned} \tag{13}$$

Next section describes with more details how production procedures are actually modeled and the type of production norms considered to constrain associated operations.

3.2. *Human Factor and Worker Profiles*

An interesting novel aspect of SOHO is the support to the explicit representation of the *human factor* within the defined contexts. SOHO interprets the *human factor* as the set of physical or abstract features that characterize the expected behavior and skills of a worker and that can directly (or indirectly) affect the interactions with a robot and the whole production as well. Specifically, SOHO characterizes the human factor with a set of DUL:Quality concerning WorkOperator. It defines concepts that “qualitatively" characterize different workers and correlate associated physical and behavioral aspects with production needs. Figure 2 shows the taxonomic structure defined

to represent behavioral and physical features of workers. Such concepts characterize the representational space of qualitative aspects of a worker (i.e., types of `DOLCE:Quality`) that are relevant with respect to production.

Concepts characterizing the qualities of the physical body of a worker, depicted in Figure 2(a), model physical, health, and cognitive parameters. Information about these variables enables the detection and monitoring of anomalous or dangerous working conditions, such as bad ergonomics, body position in hazardous areas or mental, and physical fatigue. Concepts concerning the qualities of the behavior of a worker, depicted in Figure 2(b), instead model his/her performance in a given production scenario (e.g., the expertise level or the average time taken to perform a task).

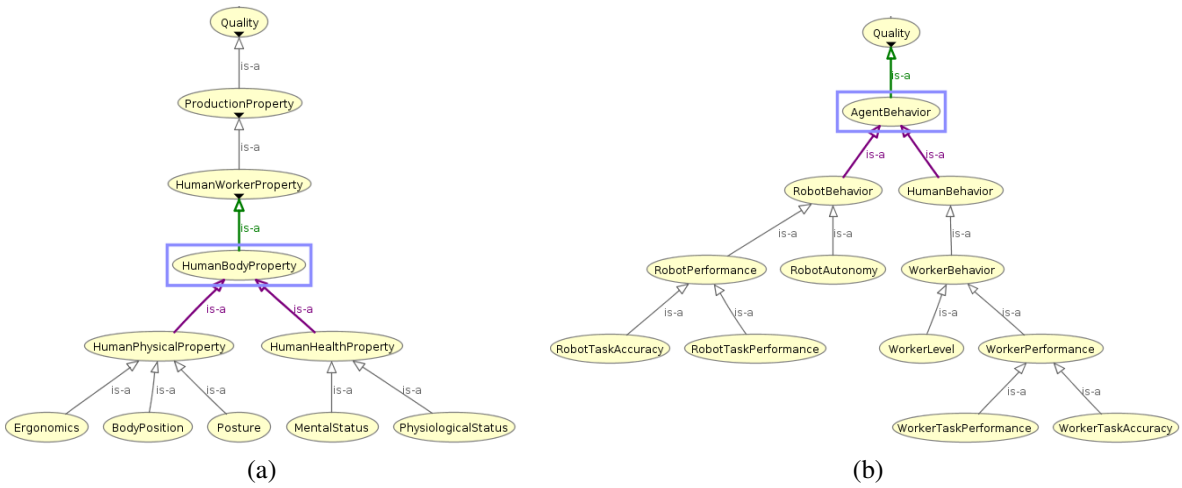


Figure 2. Excerpt of SOHO concerning physical and behavioral qualities of human workers

The concept `WorkerLevel` represents a measure of the level of knowledge of a human worker about a specific production scenario and the reliability of her performance. On the one hand, the expertise level determines the (sub)set of production tasks a human worker can carry out. For example, some tasks may require a certain minimum level of experience to be performed by a worker. On the other hand, the expertise level characterizes the expected *uncertainty* about the performance of a worker. Low experience determines a higher *variance* in the performance and thus a higher amount of *uncertainty* in terms of execution time and accuracy. Higher experience instead determines performance with lower variance in terms of "expected" execution time and achieved accuracy.

3.3. *Production Procedures and Interaction Modalities*

The definition of a `ProductionMethod` through a number of `ProductionTask` follows a hierarchical task-oriented approach [74, 75]. The top-level element is the `ProductionGoal` that is associated with a number of `ProductionMethod` defining the rules that must be considered to successfully achieve the desired production goal. SOHO specifies the associations between goals, methods and related tasks using the property `DUL:hasConstituent` which is a non-transitive relation supporting a layered description of the procedure. *Constituency* depends on some layering of the world described by the ontology. Intuitively, a constituent is a part belonging to a lower layer. Since layering is actually a partition of the world described by the ontology, constituents are not properly classified as parts, although this kinship can be intuitive for common sense. A desirable advantage of this distinction is that we are able to talk e.g. of physical constituents of non-physical objects (e.g. systems), while this is not possible in terms of parts. Following this layering of a procedure, SOHO defines three types of `ProductionTask` characterizing procedures at different levels of abstraction: (i) `ComplexTask` that can be either *disjunctive* or *conjunctive*); (ii) `SimpleTask` and; (iii) `Function`. Figure 3 shows an excerpt of SOHO pointing out the taxonomical structure of the concepts representing different types of production tasks.

A `ComplexTask` is a `ProductionTask` (i.e., an instance of `DUL:Method`) representing a compound logical operation. SOHO further defines two types of complext tasks that are `ConjuncitveComplexTask`

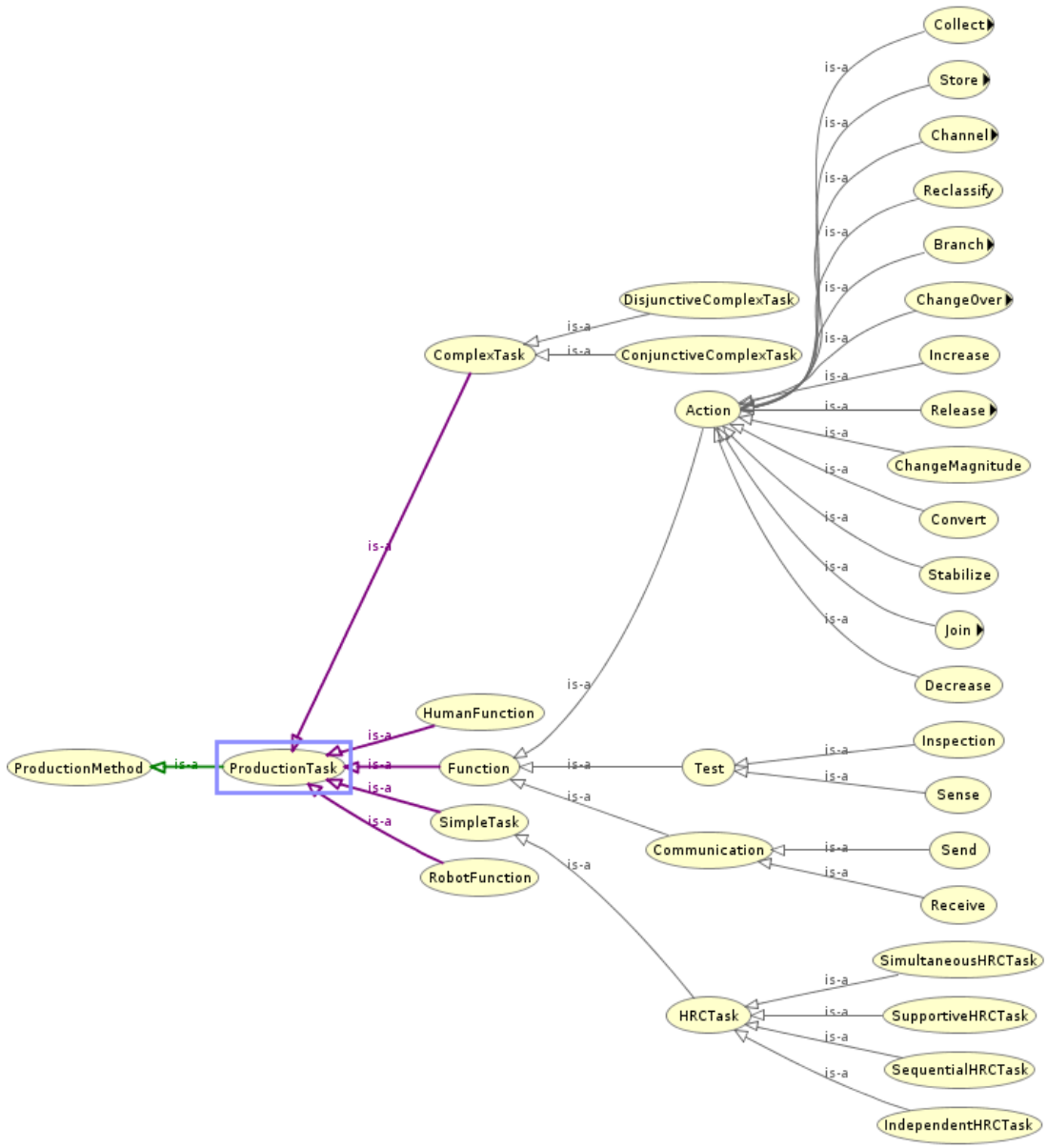


Figure 3. Excerpt of SOHO showing the taxonomical structure of production tasks. In particular the picture shows the integrated taxonomy of functions introduced in [59]

and `DisjunctiveComplexTask`. Complex tasks are generally interpreted as `ConjunctiveComplexTask` meaning that underlying associated tasks (i.e., production tasks associated through `DUL:hasConstitutent`) should all be part of a plan implementing it. Disjunctions are instead represented as `DisjunctiveComplexTask` representing alternative ways of executing a procedure. Plans should in this case implement one of available disjunctions (i.e., only one of the production tasks associated through `DUL:hasConstituent`).

$$\begin{aligned}\texttt{ComplexTask} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\,\texttt{DUL:hasConstituent}.(\texttt{ComplexTask} \sqcup \texttt{SimpleTask}) \sqcap \\ & \exists\,\texttt{DUL:isDescribedBy}.\texttt{OperativeConstraint}\end{aligned} \tag{14}$$

A SimpleTask represents a *leaf* of the hierarchical structure of a ProductionMethod. This concept describes primitive production operations that could be carried out leveraging the functional capabilities of the agents (i.e., the set of Function agents can actually implement in a given scenario). A SimpleTask requires thus the execution of a number of Function by one or more participating agents.

$$\begin{aligned}\texttt{SimpleTask} \sqsubseteq\ & \texttt{ProductionTask} \sqcap \\ & \exists\, \texttt{DUL:hasConstituent}.(\texttt{Function} \sqcup \texttt{ProductionObject}) \sqcap \\ & \exists\, \texttt{DUL:isDescribedBy}.(\texttt{InteractionModality} \sqcup \\ & \qquad \texttt{OperativeConstraint})\end{aligned} \tag{15}$$

According to the definition of SimpleTask and taking into account the definition of Function given in Equation 6, it can be observed that both concepts are associated with a ProductionObject representing the “target physical entity" of the environment. Consistency between the constituent object of a SimpleTask and the target object of the associated Function can be thus enforced through the following rule:

$$\begin{aligned}&\texttt{SimpleTask(t)} \wedge \texttt{DUL:hasConstituent(t, o)} \wedge \\ &\texttt{DUL:hasConstituent(t, f)} \wedge \texttt{ProductionObject(o)} \wedge \\ &\qquad \texttt{Function(f)} \\ &\qquad\qquad \rightarrow\ \texttt{hasTarget(f, o)}\end{aligned} \tag{16}$$

The concepts OperativeConstraint and InteractionModality are two types of ExecutionNorm SOHO defines as DUL:Norm constraining the behaviors of acting entities when participating to some production process. The former describes general rules that may constrain the execution of two or more ProductionTask. SOHO defines in particular two types of operative constraints that are: (i) PrecedenceConstraint and; (ii) ParallelExecutionConstraint. The latter describes more specific rules constraining the behavior of a human and a robot when realizing collaborative tasks. Although the “boundaries" of the *representation space* are well delimited within a domain ontology [58], there is a multitude of behaviors that can be described with a production scenario and a multitude of design choices to take into account. However accurate and complete an ontological model is, the correct definition of all necessary information and constraints is not always straightforward. Ontology design patterns [24] can play a role in supporting knowledge definition. Patterns can indeed specialize an ontological model without losing generality but defining useful “structures" that may facilitate knowledge definition.

In the considered domain, a crucial point is the representation of the hierarchical structure of production processes and the correlations between production tasks and the functions the human and the robot should perform. Ontological patterns in this case should characterize typical and/or recurrent associations between tasks and functions. Namely, they should characterize structures describing typical collaborative behaviors of human workers and robots in production scenarios. SOHO introduces HRC ontological patterns by taking into account interaction schema known in the literature [26]. First, SOHO defines the concept HRCTask or specifically describe a particular type of SimpleTask requiring the tight interaction of a human worker (i.e., a WorkOperator) and a collaborative robot (i.e., a Cobot). The basic assumption is that a HRCTask should be performed through the execution of a minimum of one Function and a maximum of two Function. Each required Function should be performed by a WorkOperator or by a Cobot. If only one function is necessary then it can be performed either by a human worker or a robot. If two functions are necessary then one function should be performed by the human and the other by the robot. Furthermore, the two functions should have effect on the same target entity of the environment (i.e., ProductionObject).

Each HRCTask is associated with a InteractionModality specifying behavioral norms constraints the way underlying Function should be executed to correctly perform the collaborative task.

$$\begin{aligned}\texttt{InteractionModality} \sqsubseteq\ & \texttt{ProductionNorm} \sqcap \\ & \exists\, \texttt{DUL:describes.ProductionTask} \sqcap \\ & \exists\, \texttt{constrains.Function}^{*}\end{aligned} \tag{17}$$

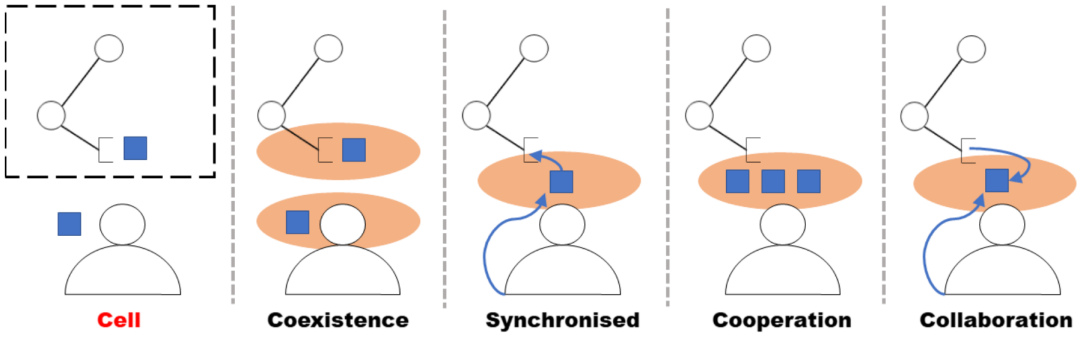


Figure 4. General structure of a HRC cell and types of collaboration modalities as defined in [25]: (i) Coexistence/Independent; (ii) Synchronised/Sequential; (iii) Cooperation/Simultaneous; (iv) Collaboration/Supportive.

* The ontological model of SOHO uses cardinality restrictions to specify that a maximum of two individuals of type `Function` should be associated to a `InteractionModality` through the property `constraints`

According to [26], the execution of a collaborative task (i.e., an individual of `HRCTask`) entails one of four different collaboration modalities: (i) *Independent*, human and robot perform their tasks on different workpieces without collaboration; (ii) *Simultaneous*, human and robot perform distinct tasks on the same workpiece at the same time, still without physical interaction; (iii) *Supportive*, human and robot perform the same task on the same workpiece and they work simultaneously and cooperatively on the same task. (iv) *Sequential*, human and robot should complete sequential tasks on the same workpiece. Figure 4 shows a graphical representation of these four types of collaborative tasks.

Following this classification, four types of `InteractionModality` have been defined as four patterns characterizing specific knowledge structures in terms of associated concepts and cardinality restrictions. An interaction modality of type `Independent` requires a human or a robot working on a particular target object independently from each other but in a shared space. It describes one `HRCTask` and constrains one `Function` which can be either a `HumanFunction` (i.e., a `Function` that can be performed by a `WorkOperator`) or a `RobotFunction` (i.e., a `Function` that can be performed by a `Cobot`).

$$\begin{aligned}\texttt{Independent} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\,\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\,\texttt{constrains.(HumanFunction} \sqcup \\ & \qquad\qquad \texttt{RobotFunction)}\end{aligned} \tag{18}$$

An interaction modality of type `Simultaneous` requires a human and a robot performing two different operations on a same target object at a same time. It describes a `HRCTask` and constrains exactly one `RobotFunction` and one `HumanFunction` to be executed at the same time. Namely, the execution of the two operations may overlap in time and should not follow a specific ordering (e.g., precedence constraint).

$$\begin{aligned}\texttt{Simultaneous} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\,\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\,\texttt{hasConstituent.HumanFunction} \sqcap \\ & \exists!\,\texttt{hasConstituent.RobotFunction} \sqcap\end{aligned} \tag{19}$$

An interaction modality of type `Sequential` requires a human and a robot to perform operations on the same object according to a strict order. The pattern in this case forces the necessary `RobotFunction` and the `HumanFunction` to be executed according to a specified precedence constraint. It is therefore associated to one `PrecedenceConstraint` which specifies the `Function` to be executed as first and the `Function` to be

executed as second.

$$\begin{aligned}\texttt{Sequential} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\,\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\,\texttt{DUL:hasConstituent.HumanFunction} \sqcap \\ & \exists!\,\texttt{DUL:hasConstituent.RobotFunction} \sqcap \\ & \exists!\,\texttt{DUL:isDescribedBy.PrecedenceConstraint}\end{aligned} \tag{20}$$

An interaction modality of type `Supportive` requires a human and a robot to perform the same operation on a same object at a same time. The pattern in this case forces the execution of a `RobotFunction` and a `HumanFunction` to start and end at the same time. Although the operation is the same, it is necessary to model two distinct instances of human and robot `Function` to comply with the ontological model. Namely, a human and a robot execute two distinct operation of the same type (i.e., two instances of the same type of `Function`) and they are executed in strict parallelism. The required temporal constraint is described by the `ParallelExecutionContraint` which enforces a strict parallel execution among all associated functions (the human and the robot function in this case).

$$\begin{aligned}\texttt{Supportive} \sqsubseteq\ & \texttt{InteractionModality} \sqcap \\ & \exists!\,\texttt{DUL:describes.HRCTask} \sqcap \\ & \exists!\,\texttt{DUL:hasConstituent.HumanFunction} \sqcap \\ & \exists!\,\texttt{DUL:hasConstituent.RobotFunction} \sqcap \\ & \exists!\,\texttt{DUL:isDescribedBy.ParallelExecutionConstraint}\end{aligned} \tag{21}$$

The described interaction modalities can be used as general schema to structure the interactions of human workers and collaborative robots within collaborative processes. Following these schema, SOHO defines four specific types of `HRCTask` each associated with a specific `InteractionModality`: (i) `IndependentHRCTask`; (ii) `SimultaneousHRCTask`; (iii) `SequentialHRCTask` and; (iv) `SequentialHRCTask`. These tasks provide designers with generic and reusable concepts suitable to facilitate the description of human-robot collaborative dynamics in manufacturing scenarios.

4. Knowledge Definition and Automated Synthesis of Plan-based Control Models

A knowledge base (ABox) compliant with the described ontological model (TBox) completely characterizes a HRC scenario from different perspectives. The use of semantic technologies based on RDFS [76], OWL [60] and SPARQL [77] supports accessibility, interoperability and thus the integration of resulting knowledge in different production-related processes and use for different purposes. In this work we are especially interested in showing how the proposed semantics and related knowledge bases would contribute to the enhancement of *awareness*, *flexibility* and *autonomy* of collaborative robots and HRC cells as a whole. On the one hand the obtained production knowledge supports flexible configuration of robot controllers through the automatic synthesis of *task planning models*. On the other hand, it supports *cognitive processes* suitable to adapt robot actions and human-robot coordination policies to the interpreted states of a production environment (i.e., observed events). More specifically, this section shows how production knowledge is used to automatize the synthesis of *task planning* models and to coordinate human and robot operations through deliberative plan-based control [11, 27]. A knowledge extraction procedure bridges the gap between knowledge representation and task planning for robot control, supporting the realization of a cognitive "perceive, reason, act" loop [16, 20].

Task planning generally relies on AI Planning and Scheduling technologies [37, 38, 78] in order to allow robotic agents (or more in general artificial agents) to autonomously synthesize and execute plans that achieve some desired goal. Such plans are generally seen as sequences of actions to be executed starting from an initial state. Several planning paradigms and tools have been introduced in the literature each making different assumptions and supporting different reasoning capabilities e.g., classical symbolic reasoning [37, 38, 79], numeric and temporal reasoning

[78, 80] or hierarchical reasoning [81, 82]. Considering the HRC domain, it is crucial to consider the simultaneous execution of human and robot actions, taking into account the quality of the resulting collaborative process e.g., cycle time, idle time of the robot or risks of collision. Furthermore, the tight and continuous interaction between the human and the robot introduces a significant source of *uncertainty* from a control perspective. The actual behavior of human agents and the actual duration of their actions are indeed unpredictable and cannot be always controlled (*controllability issues* [83, 84]). Therefore, planners should properly deal with this level of uncertainty in order to synthesize plans that are effective and reliable when executed in the real environment. For these reasons this work specifically considers the *timeline-based planning formalism*, as introduced in [85], and proposes a knowledge extraction procedure mapping production knowledge to timeline-based specifications. The timeline-based approach is a temporal planning formalism introduced in early 90s [86] capable of dealing with concurrency, numeric and temporal constraints. It has been successfully applied in many real-world scenarios e.g., [87–89]. The work [85] formalizes timeline-based planning by introducing the notions of *temporal uncertainty* and *controllability issues* [90]. This formalism has been applied to HRC scenarios [22, 91] properly dealing with human-related uncertainty. It supported indeed the synthesis and execution of plans achieving suitable trade-offs among cycle time (i.e., efficiency of the collaborative process), reliability and human safety. Before entering into the details of the developed knowledge extraction procedure, next sub-section briefly introduces the main concepts regarding timelines, as introduced in [85].

4.1. Plan-based Control through Timeline-based Planning and Execution

A timeline-based planning specification describes valid temporal behaviors of a number of domain features to be controlled. The planning process consists in synthesizing valid flexible behaviors (i.e., timelines) describing *how* these features should evolve over time to achieve some given *objectives* (i.e., *which* states/actions assumes/executes and *when*). According to [85], *state variables* model domain features by specifying valid temporal behaviors in terms of allowed timed sequences of states/actions (generally denoted as *state variable values*).

Definition 1. *A* State Variable *is a tuple* $SV = \langle V, T, D, \gamma \rangle$ *describing valid behaviors of a domain feature:*

- V *is a set of* values $v_i \in V$ *representing states of actions the feature can perform or assume over time.*
- $T : V \rightarrow 2^V$ *is a* state transition function *describing for each value* $v_i \in V$ *possible successors on a timeline and thus valid transitions.*
- $D : V \rightarrow \mathbb{T} \times \mathbb{T}$ *is a* duration function *specifying for each value* $v_i \in V$ *its expected duration bounds, expressed in some temporal domain* $\mathbb{T}$ *(typically* $\mathbb{N}^+$*).*
- $\gamma : V \rightarrow \{c, pc, u\}$ *is a* controllability tagging function *specifying the controllability property of a value.*

Controllability properties characterize the execution of SVs' values with respect to the dynamics of the environment.

Definition 2. *A value* $v_i \in V$ *of a state variable* $SV = \langle V, T, D, \gamma \rangle$ *is:*

- Controllable *(*$\gamma(v_i) = c$*) if the control system can decide both the start and end times for its execution.*
- Partially-controllable *(*$\gamma(v_i) = pc$*) if it can only be started by the control system, while its end time and, i.e., its actual duration, can be only observed.*
- Uncontrollable *(*$\gamma(v_i) = u$*) the control system can neither decide its start nor its end times.*

Information about controllability and temporal flexibility are crucial to solve planning problems while dealing with *temporal uncertainty* to support robust plan execution [83, 92, 93].

A *flexible timeline* for a state variable SV_i is a sequence of (flexible) temporal intervals called *tokens* that describe an *envelope* of valid temporal behaviors.

Definition 3. *If* $SV_i = (V, T, \gamma, D)$ *is a state variable, a* token x_j *for the variable has the form:*

$$x_j = (v_k, [e_j, e_j^{'}], [d_j, d_j^{'}], \gamma(v_k))$$

where $v_k \in V$ *is the value assumed by the token* x_j*,* $[e_j, e_j^{'}]$ *is the end-time interval of* x_j *(with* $e_j \leqslant e_j^{'}$*) and* $[d_j, d_j^{'}]$ *is the minimum and maximum duration of* x_j *(with* $d_j \leqslant d_J^{'}$*).*

A token x_j represents a specific allocation of a value $v_k \in V$ over a certain (flexible) temporal interval. The duration of a token must be consistent with the duration bounds of the associated value v_k. A *timeline* is a *continuous sequence of tokens* x_i describing the behavior of a domain feature from a *temporal origin* to (at least) a *planning horizon* $\mathtt{H} \in \mathbb{T}$. The start-time interval of a token is not explicitly represented since it coincides with the end-time interval of the previous token in the timeline (the first token of a timeline starts at the temporal origin [0, 0]).

Definition 4. *A* timeline *FTL_i for a state variable $SV_i = (V, T, D, \gamma)$ is a continuous and finite sequence of tokens of the form:*

$$x_1 = (v_1, [e_1, e_1^{'}], [d_1, d_1^{'}], \gamma(v_1)),$$
$$\ldots,$$
$$x_m = (v_m, [e_m, e_m^{'}], [d_m, d_m^{'}], \gamma(v_m)),$$

where $v_1, \ldots, v_m \in V$ and for all $j = 1, \ldots, m-1$, $v_{j+1} \in T(v_j)$. Denoting with $start(x_j)$ the computed start time interval of a token x_j then, for all $j = 1, \ldots, m-1$, $[e_j, e_j^{'}] = start(x_{j+1})$ (i.e., *a timeline is a continuous sequence of non-overlapping tokens).*

Synchronization rules specify additional constraints that are necessary to synthesize timelines that achieve desired objectives (e.g., planning goals).

Definition 5. *A* synchronization rule *has the form*

$$a_0[SV_0 = v_0] \rightarrow a_1[SV_1 = v_1], \ldots, a_n[SV_n = v_n].\mathcal{C}$$

where every $a_i[SV_i = v_i]$ is a token variable *denoting a temporal interval in which a state variable SV_i assumes the value v_i. The left-hand part of the synchronization rule ($a_0[SV_0 = v_0]$) is called the* trigger of the rule. The set $\mathcal{C}$ specifies *temporal relations* between token variables.

Synchronization rules with the same trigger are treated as disjunctions and represent alternative constraints that should hold between different sets of token variables.

A planning problem then consists of a set of partially instantiated timelines that specify known *facts* about the initial state of domain features (i.e., tokens specifying the values assumed by each state variable at plan origin) and *goals* constraining their temporal evolution (i.e.,. tokens specifying the desired values that one or more state variables should assume during certain temporal intervals). A planning process should synthesize valid and complete temporal behaviors of all (controllable) state variables (i.e., *timelines*) such that all *duration constraints*, *value transition constraints* and *temporal constraints* of applied synchronization rules are satisfied. The reader may refer to [85] for a complete definition of planning concepts.

4.2. *Model Generation through Knowledge Extraction*

The objective of the knowledge extraction procedure is to synthesize a valid timeline-based specification of a task planning problem to coordinate human and robot agents in a efficient and reliable way. The procedure leverages SOHO to extract information from a knowledge base and organize them into appropriate structures. The main components of a timeline-based model are the *state variables* and the *synchronization rules* that should describe valid temporal behaviors of physical/logical entities responsible for the actual implementation of the production dynamics of a HRC cell. Synchronization rules coordinate the behavior of modeled state variables by imposing constraints necessary to carry out production processes correctly. Figure 5 shows the general structure of the knowledge extraction pipeline which is organized into two macro-steps. First, it extracts knowledge useful to generate the state variable of the model. Then, it extracts knowledge useful to generate the synchronization rules. States variables should correlate the production requirements of a HRC cell with the particular functioning of acting entities. A number of state variables should thus describe supported production goals and related production tasks at different levels of abstraction. Other state variables then should describe production operations a worker and a robot can perform. Production knowledge is characterized in a hierarchical way correlating high-level production objectives to low-level

tasks and supported manufacturing functions. The resulting timeline-based specification follows a hierarchical specification decomposing high-level goals into increasingly simple/primitive logical tasks and then physical operations directly implemented by a worker and a robot [22, 23].

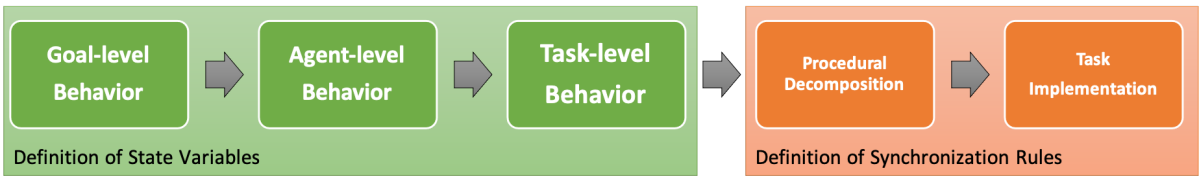


Figure 5. General structure of the knowledge extraction pipeline developed for the automatic generation of timeline-based task planning models

4.2.1. *Knowledge Extraction Pipeline*

The first step of the pipeline extracts knowledge about production goals to define a *goal state variable* SV_G. Each value $v_{G,i} \in V_G$ of the state variable denotes an individual of `ProductionGoal` and represents a particular production process that can be carried out within a HRC cell. These values represent possible high-level planning requests requiring the synthesis of suitable worker and robot behaviors implementing the related production process. The second step extracts knowledge about the functional behavior of the agents that would take part to production procedures. For each individual A_x of `DUL:Agent` found in the knowledge base, the procedure defines a dedicated state variable SV_{A_x}. Each state variable SV_{A_x} describes the set of low-level operations the agent A_x can actually perform in the given context. Each operation corresponds to an individual of `Function` the agent A_x can perform as inferred by Equation 7. For each agent A_x and set of associated functions $F_{A_x} = f_{A_x,1}, ..., f_{A_x,m}$, the procedure defines a state variable SV_{A_x} whose values $v_{A_x,i} \in V_{A_x}$ correspond to the functions $f_{A_x,i} \in F_{A_x}$. Considering the typical dyadic structure of HRC cells, this step generates a state variable SV_H for the worker (i.e., SV_H is associated with the individual of `WorkOperator`) and a state variable SV_R for the robot (i.e., SV_R is associated with the individual of `Cobot`). The sets of values V_H and V_R contain then the inferred instances of `Function` that respectively the human and the robot can perform, according to their capabilities.

It is worth noticing that values of the human state variable $v_{H,i} \in V_H$ are all defined as *uncontrollable* (i.e., $\gamma(v_{H,i}) = u$). Indeed, the planner can only assume that the execution of these functions would follow a certain (expected) schedule but cannot actually control their duration. The values of the robot state variable $v_{R,i} \in V_R$ are instead all defined as *partially controllable* (i.e., $\gamma(v_{R,i}) = pc$). The planner can decide when to start the execution of these functions but cannot control the end because of the co-existence with the human (e.g., the human can indeed slowdown or even cause the stop of robot motions generating uncertainty with respect to their actual duration).

The third step of the pipeline defines state variables modeling the behavior of a production procedure at different hierarchical levels. These state variables (together with the synchronization rules) encapsulate a hierarchical structure correlating high-level objectives to the functions of the worker and the robot. This step in particular leverages the property `DUL:hasConstituent` associating individuals of `ProductionGoal` to `ProductionMehtod` and in turn to individuals of `ProductionTask`. According to the definition of `ProductionGoal` and `ComplexTask` in respectively Equation 8 and Equation 14, the property `DUL:hasConstituent` enforces a layered description of production procedures. Following the relationships established by this property, this step first builds a *decomposition graph* encapsulating the procedural decomposition of goals in simpler tasks. The nodes of the graph are associated with the individuals of `ProductionGoal` (the roots of the graph) and with the individuals of `ComplexTask` or `SimpleTask`. Nodes associated with individuals of `SimpleTask` are the leaves of the decomposition graph. Each layer of the graph contains a number of individuals of `ProductionTask` describing the production procedure at a certain level of abstraction. Given this data structure the procedure defines a *production state variable* SV_{L_x} for each layer L_x of the decomposition graph. Each value $v_{L_x,i} \in V_{L_x}$ of a state variable SV_{L_x} corresponds to an individual of `ProductionTask` belonging to the hierarchical level L_x.

Once all the state variables are defined, the second macro-step of the procedure generates the synchronization rules that constrain the values of the different state variables. Two types of synchronization rules are defined with two dedicated steps as shown by Figure 5. A first step defines synchronization rules correlating high-level objectives

(i.e., values $v_i \in V_G$ of SV_G) with production tasks (i.e., values $v_{L_x,i} \in V_{L_x}$ of SV_{L_x}). A second step defines synchronization rules correlating values associated with `SimpleTask` (i.e., the leaves of the hierarchical decomposition graph) to the values associated with the `Function` the worker and the robot can actually perform.

The first set of rules specify the temporal and causal constraints necessary to correctly implement the modeled procedures. Layering relationships established by `DUL:hasConstitutent` are encoded using the temporal constraint `CONTAINS` [94] correlating each `ProductionGoal` or a `ProductionTask` to simpler `ProductionTask` (i.e., a decomposition). Let us suppose that a value $v_{L_x,i} \in V_{L_x}$ is associated to one or more values $v_{L_y,i} \in V_{L_y}$ where $L_x > L_y$ (i.e., SV_{L_x} is at a higher level of abstraction than SV_{L_y}), the procedure defines the following rule: (i) the value $v_{L_x,i}$ is the trigger of the rule; (ii) the values $v_{L_y,i}$ are part of the body of the rule; (iii) a temporal constraint `CONTAINS` [94] is specified between the triggerer (i.e., $v_{L_x,i}$) and each value $v_{L_y,i}$ of the body. A rule so defined enforces the hierarchical decomposition of the task at a higher level of abstraction into a number of tasks at a lower level of abstraction.

The second set of rules constrains possible task allocations and possible interactions between a worker and a robot. It is at this level that ontological patterns about known collaboration modalities (i.e., Equation 18, Equation 19, Equation 20 and Equation 21) are used to define temporal constraints determining specific human-robot coordinated behaviors (*behavioral patterns*).

4.2.2. *Detailed Implementation and Generation of Timeline-based Structures*

Given the general procedure in Figure 5, Algorithm 1 shows the implemented steps and how the associated timeline-based structures are created. The procedure extracts the necessary information from the knowledge base and generates the state variables and the synchronization rules composing the timeline-based model. This section describes with further details how the ontological model is used to extract necessary information from the knowledge base and how this information is encapsulated into the associated timeline-based model. The knowledge base and the extraction procedure have been developed using the open-source software Apache Jena [5]. The knowledge base is therefore stored and manipulated as a RDF Knowledge Graph (KG). Triples of the KG are accessed/filtered using *RDF pattern matching* mechanisms exposed by Jena APIs. Algorithm 1 follows the steps of the pipeline in Figure 5 and thus can be divided into two macro-steps. Rows 1-16 concern the first macro-step of the procedure implementing the first three steps of the pipeline. Rows 17-29 concern the second macro-step of the procedure implementing the last two steps of the pipeline. This part of the procedure creates the synchronization rules of the timeline-based specification.

The first step of the pipeline concerns the creation of the state variable SV_G describing the high-level goals that can be achieved within the considered HRC cell. The procedure first extracts from the knowledge base the set of *individuals* of `ProductionGoal` G (row 2). The procedure extract_goals $(\mathcal{KB})$ specifically retrieves all the triples matching a pattern of the form $\langle$?g rdf:type soho:ProductionGoal$\rangle$. Such triples contain the RDF ID of the individuals of `ProductionGoal` in place of the variable ?g. Each individual $g \in G$ represents a high-level planning request the task planner could receive over time. Each planning request triggers a planning process to synthesize collaborative plans coordinating human and robot skills in order to achieve the requested objective

The procedure create_statevariable $(\mathcal{KB}, G)$ (row 3) uses these individuals to generate the goal state variable SV_G of the planning model. Algorithm 2 shows the general procedure generating a state variable description. Algorithm 2 creates a state variable following Definition 1. Given a set of individuals $\mathcal{I}$, it defines the set of values $\mathcal{V}$, the transition function $\mathcal{T}$, the duration function $\mathcal{D}$ and the controllability tagging function γ The procedure initializes the set of values V with a default *Idle* value denoting a stable state of the modeled feature (row 1) (e.g., no operation or no production process is being executed). For each individual $I_i \in \mathcal{I}$ the procedure then adds a value v_i to the set of state variable values V (rows 3-4). It uses the *data property* `hasLabel` to extract a symbol identifying the individual I_i within the knowledge base (row 3). For each value added to the state variable, it is necessary to update the transition function $\mathcal{T}$ accordingly (rows 5-6). The procedure adds a transition constraint from the value v_i to the default value *Idle* (row 5) and a transition constraint from *Idle* to v_i (row 6). The resulting "shape" of a state variable is an automaton where all values $v_i \in V$ go to *Idle* and *Idle* goes to all values $v_i \in V$. The rest of the procedure extracts information about duration bounds and controllability properties of the values (rows 7-8). Information about average

[5] https://jena.apache.org/

Algorithm 1 Timeline-based specification extraction procedure

Require: $\mathcal{KB}$ ▹ Production knowledge
Ensure: $\mathcal{M} : \langle \mathcal{SV}, \mathcal{R} \rangle$ ▹ Timeline-based model
$\mathcal{SV}, \mathcal{R} \leftarrow \{\}$
$G \leftarrow$ extract_goals $(\mathcal{KB})$ ▹ Step 1.1 - goal-level behavior of Figure 5
$SV_G \leftarrow$ create_statevariable $(\mathcal{KB}, G)$ ▹ See Algorithm 2
$\mathcal{SV} \leftarrow \{SV_G\}$
$F \leftarrow$ extract_functions $(\mathcal{KB})$ ▹ Step 1.2 - agent-level behavior of Figure 5
$SV_H \leftarrow$ create_statevariable $(\mathcal{KB}, \text{human}\,(F))$ ▹ See Algorithm 2
$SV_R \leftarrow$ create_statevariable $(\mathcal{KB}, \text{robot}\,(F))$ ▹ See Algorithm 2
$\mathcal{SV} \leftarrow \{SV_H, SV_R\}$
$\mathcal{G} = \langle \mathcal{N}_\mathcal{T}, \mathcal{E}_\mathcal{T} \rangle \leftarrow$ decomposition_graph $(\mathcal{KB})$ ▹ See Algorithm 3
$T = \{\{t_1^0, ..., t_n^0\}, ..., \{t_1^H, ..., t_m^H\}\} \leftarrow$ hierarchy $(\mathcal{G})$
$L \leftarrow 1$ ▹ Skip the hierarchical level 0 composed by production goals already in SV_G
while $L \leqslant H$ **do** ▹ Step 1.3 - task-level behavior of Figure 5
  $T^L = \{t_1^L, ..., t_k^L\} \leftarrow T[L]$
  $SV_L \leftarrow$ create_statevariable $\left(\mathcal{KB}, T^L\right)$
  $\mathcal{SV} \leftarrow \{SV_L\}$ ▹ See Algorithm 2
  $L \leftarrow L + 1$
$L \leftarrow 0$ ▹ Definition of synchronization rules
while $L \leqslant H$ **do**
  $T^L = \{t_1^L, ..., t_k^L\} \leftarrow T[L]$
  for $t^L \in T^L$ **do**
    if *is* $\left(\mathcal{KB}, t^L, (\texttt{ComplexTask} \vee \texttt{ProductionGoal})\right)$ **then** ▹ Step 2.1 - Procedural decomposition of Figure 5
      $T^{L'} = \{\{t_1^{L',1}, ..., t_n^{L',1}\}, ..., \{t_1^{L',S}, ..., t_m^{L',S}\}\} \leftarrow$ subtasks $\left(\mathcal{G}, t^L\right)$ ▹ Extract alternative subtasks of t^L
      for $T^{L',i} = \{t_1^{L',i}, ..., t_n^{L',i}\} \in T^{L'}$ **do** ▹ Check $|T^{L'}| = S$ possible decomposition
        $R^L \leftarrow$ create_decomposition_rule $\left(\mathcal{KB}, t^L, T^{L',i}\right)$ ▹ See Algorithm 5
        $\mathcal{R} \leftarrow \{R^L\}$
    if *is* $\left(\mathcal{KB}, t^L, \texttt{SimpleTask}\right)$ **then** ▹ Step 2.2 - Task implementation of Figure 5
      $F = \{f_1, ..., f_n\} \leftarrow$ extract_constituents $\left(\mathcal{KB}, t^L\right)$
      $R^L \leftarrow$ create_decomposition_rule $\left(\mathcal{KB}, t^L, F\right)$ ▹ See Algorithm 5
      $\mathcal{R} \leftarrow \{R^L\}$
  $L \leftarrow L + 1$
return $\mathcal{M} : \langle \mathcal{SV}, \mathcal{R} \rangle$

execution time is retrieved through the data property `hasDuration`. If no information is associated with the considered individual I_i a default range $(1, +\infty)$ is considered (e.g., individuals of `ProductionGoal` do not have such information associated). Similarly, controllability information is set according to the type of the considered individual I_i. In the case of individuals of `ProductionGoal` or `ProdcutionTask` values are generally declared as *controllable*. Listing 1 shows a generic example of a resulting timeline-based description.

Algorithm 2 State Variable creation procedure

Require: $\langle \mathcal{KB}, \mathcal{I} \rangle$ ▹ Production knowledge and a set of individuals
Ensure: $\mathcal{SV} : \langle \mathcal{V}, \mathcal{T}, \mathcal{D}, \gamma \rangle$ ▹ A state variable description compliant with Definition 1
$\mathcal{V} \leftarrow \{\text{Idle}\}$
for $I_i \in \mathcal{I}$ **do**
  $v_i \leftarrow$ extract_label $(\mathcal{KB}, I_i)$
  $\mathcal{V} \leftarrow \{v_i\}$
  $\mathcal{T}(v_i) \leftarrow \{\text{Idle}\}$
  $\mathcal{T}(\text{Idle}) \leftarrow \{v_i\}$
  $\mathcal{D}(v_i) \leftarrow$ extract_bounds $(\mathcal{KB}, I_i)$
  $\gamma(v_i) \leftarrow$ controllability $(\mathcal{KB}, I_i)$ ▹ Set controllability information according to the type of the individual I_i
return $\mathcal{SV} : \langle \mathcal{V}, \mathcal{T}, \mathcal{D}, \gamma \rangle$

Listing 1: Example of a state variable generated by Algorithm 2 and described according to the language syntax given in [95]

```
COMP_TYPE StateVariable GeneratedStateVariable (Idle(), value_1(), value_2(), ..., value_N()) {

        VALUE <c> Idle() [1, +INF]          // example of value declared as controllable
        MEETS {
            value_1();
            value_2();
            ...
            value_N();
        }

        VALUE <c> value_1() [min, max]      // example of value declared as controllable
        MEETS {
            Idle();
        }

        VALUE <pc> value_2() [min, max]      // example of value declared as partially controllable
        MEETS {
            Idle();
        }

        ...

        VALUE <u> value_N() [min, max]      // example of value declared as uncontrollable
        MEETS {
            Idle();
        }
}
```

Going back to Algorithm 1, the procedure continues the generation of a timeline-based planning model by creating the state variables related to the human and the robot (rows 5-8). It extracts the set F of individuals of `Function` that can be performed by some `DUL:Agent` in the environment (row 5) as inferred through the rule of Equation 7. The procedure extract_functions $(\mathcal{KB})$ specifically retrieves all the triples matching a pattern of the form $\langle\ ?f$ soho:canBePerformedBy $?a\rangle$. To create the state variable of the human (SV_H) and the robot (SV_R), the individuals of `Function` in F are filtered according to the agent that can perform them. In general a distinct "agent state variable" SV_A is created for each distinct individual of `DUL:Agent` capable of performing some `Function` in the given scenario.

Considering the specific focus of this work we assume here that only one agent of type `WorkOperator` and only one agent of type `Cobot` can perform functions in the production scenario. Therefore, the set F is partitioned according to these two specific types of agents. The sub-procedures human (F) and robot (F) in row 6 and row 7 respectively filter the individuals of `Function` that can be performed by agents of type `WorkOperator` and `Cobot`. The set of functions that can be performed by an agent of type `WorkOperator` is used to create the state variable of the human SV_H (row 6). The subset of individuals of `Function` that can be performed by an agent of type `Cobot` is used to create the state variable of the robot SV_R (row 7).

State variables are created through the same procedure of Algorithm 2 which generates an outcome similar to the code in Listing 1. In the case of the human state variable SV_H values are all tagged as *uncontrollable* since the task planner could not decide their actual execution duration. In the case of the robot state variable SV_R values are all tagged as *partially controllable* since the task planner can decide the start of their execution but cannot control their actual duration. Furthermore, information about the expected duration of these values is set according to the known average execution time stored into the knowledge base (hasDuration). In the case of the human, this information takes into account also "profile-level" information about the worker in order to estimate the reliability of her/his performance. In this regard, information about the associated `WorkerLevel` determines a higher or lower *uncertainty index* used to compute the expected lower and upper bound of the duration. Intuitively, the higher the level of the worker (higher experience) the lower the uncertainty about the execution time of associated functions. Conversely, the lower the level of the worker the higher the uncertainty about the execution time of associated functions

The last set of state variables created by Algorithm 1 concerns the set of `ProductionTask` correlating `ProductionGoal` to supported `Function`. These set of state variables (together with synchronization rules) are thus crucial to correctly model operational constraints and allow the task planner to synthesize plans that are valid and effectively implement the desired production procedures. Such production procedures are generally encoded in

a hierarchical way into the knowledge base through the property `DUL:hasConstituent`. To correctly define the needed state variables is therefore necessary to properly encode the procedure and the hierarchical relationships between the involved `ProductionTask`. To this aim, the procedure extracts a *decomposition graph* $\mathcal{G} = \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ and a *hierarchy of tasks* $\mathcal{T}$ from the knowledge base (rows 9-10).

The graph $\mathcal{G}$ is an AND/OR graph composed by three types of nodes: (i) *task nodes*; (ii) *AND nodes* and; (iii) *OR nodes*. A *task node* is connected to a *AND node*, to a *OR node* or to no nodes (leaves). Edges thus represents decomposition relationships between nodes. If a node n is connected to a *AND node* then all connected *task nodes* n_i (i.e., the task nodes n_i reachable from n through the *AND node*) represent a *conjunctive decomposition* of n. Namely, all task nodes n_i refer to `ProdcutionTask` that must be executed in order to correctly implement the "parent" `ProductionTask` denoted by n. Instead, if a node n is connected to a *OR node* then all connected *task nodes* n_i (i.e., the task nodes n_i reachable from n through the *OR node*) represent a *disjunctive decomposition* of n. Namely, all task nodes n_i refer to alternative `ProductionTask` that can be executed to implement the "parent" `ProductionTask` denoted by n. Broadly speaking, the root nodes of the graph are *task nodes* associated with individuals of `ProductionTask`. Leaves of the graph are instead *task nodes* associated with individuals of `SimpleTask`. Such tasks are the leaves of production procedures and are implemented by the `Function` supported by the human and the robot.

Algorithm 3 Construction of a AND/OR graph encapsulating task decomposition within production procedures

Require: $\mathcal{KB}$ ▹ Production knowledge
Ensure: $\mathcal{G} : \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ ▹ Directed graph representing modeled production procedures
1: $\mathcal{N}_T, \mathcal{E}_T \leftarrow \{\}$
2: $G \leftarrow$ extract_goals $(\mathcal{KB})$ ▹ Retrieve individuals of `ProductionGoal`
3: **for** $g \in G$ **do**
4: $n^g \leftarrow$ create_node (g) ▹ Create a root node of the graph associated with the `ProductionGoal` g
5: $\mathcal{N}_T \leftarrow \{n^g\}$
6: $M^g \leftarrow$ extract_constituents $(\mathcal{KB}, g)$ ▹ Retrieve individuals of `ProductionMethod` associated with g
7: **if** $|M^g| > 1$ **then**
8: $n^g_{OR} \leftarrow$ create_or_node $()$ ▹ Disjunctive decomposition of g through multiple methods $m \in M^g$
9: $\mathcal{N}_T \leftarrow \{n^g_{OR}\}$
10: $\mathcal{E}_T \leftarrow \langle n^g, n^g_{OR} \rangle$ ▹ Add an edge form the root node n^g to the disjunctive OR node n^g_{OR}
11: **for** $m \in M^g$ **do**
12: $n^{g,m}_{AND} \leftarrow$ create_and_node $()$
13: $\mathcal{N}_T \leftarrow \{n^{g,m}_{AND}\}$
14: $\mathcal{E}_T \leftarrow \{n^g_{OR}, n^{g,m}_{AND}\}$
15: $T^m \leftarrow$ extract_constituents $(\mathcal{KB}, m)$ ▹ Retrieve `ProductionTask` that are constituent of m
16: **for** $t \in T^m$ **do**
17: $n^t \leftarrow$ create_node (t) ▹ Create a node associated with the `ProductionTask` t
18: $\mathcal{N}_T \leftarrow \{n^t\}$
19: $\mathcal{E}_T \leftarrow \langle n^{g,m}_{AND}, n^t \rangle$
20: graph_refinement $(\mathcal{KB}, \mathcal{G}, t)$ ▹ See Algorithm 4
21: **else if** $|M^g| == 1$ **then**
22: $n^g_{AND} \leftarrow$ create_and_node $()$ ▹ Direct decomposition of g through a single method in M^g
23: $\mathcal{N}_T \leftarrow \{n^g_{AND}\}$
24: $\mathcal{E}_T \leftarrow \langle n^g, n^g_{AND} \rangle$
25: $T^m \leftarrow$ extract_constituents $(\mathcal{KB}, m = M^g[0])$
26: **for** $t \in T^m$ **do**
27: $n^t \leftarrow$ create_node (t)
28: $\mathcal{N}_T \leftarrow \{n^t\}$
29: $\mathcal{E}_T \leftarrow \langle n^g_{AND}, n^t \rangle$
30: graph_refinement $(\mathcal{KB}, \mathcal{G}, t)$ ▹ See Algorithm 4
31: **else**
32: $\mathcal{N}_T \leftarrow \{\}, \mathcal{E}_T \leftarrow \{\}$ ▹ No decomposition for g since no `ProductionMethod` found in $\mathcal{KB}$
33: **return** $\mathcal{G} : \langle \mathcal{N}_T, \mathcal{E}_T \rangle$

Algorithm 3 describes with further details how the graph $\mathcal{G} = \langle \mathcal{N}_T, \mathcal{E}_T \rangle$ is actually built from the knowledge base $\mathcal{KB}$. The procedure extract_goals $(\mathcal{KB})$ first extracts the set G of individuals of `ProductionGoal`

from $\mathcal{KB}$ (row 2). The graph $\mathcal{G}$ is incrementally refined by exploring their possible decomposition through the property `DUL:hasConstituent` (rows 3-32). For each individual $g \in G$, first the procedure create_node (g) creates and adds to $\mathcal{G}$ a root *task node* n^g associated with the individual g (rows 4-5). Then, the procedure extract_constituents $(\mathcal{KB})$ retrieves the set of `ProductionMethod` M^g associated to the goal g (row 6). Specifically, this procedure retrieves triples from $\mathcal{KB}$ that match the patter $\langle$?g dul:hasConstituent ?$m\rangle$ replacing pattern variable ?g with the ID of individual g.

If more than one method is found $M^g > 1$ then each method describes an alternative procedure associated with `ProductionGoal` g (rows 8-20). If only one method is found $|M^g| == 1$ then a single conjunctive decomposition is associated with `ProductionGoal` g (rows 21-30).

- In the first case, the procedure creates an *OR node* n^g_{OR} and associates it to the root task node n^g through the directed edge $\langle n^g, n^g_{OR}\rangle$ (rows 8-10). The *OR node* n^g_{OR} should in turn be connected to the alternative decomposition encapsulated by each method m. Thus, for each method $m \in M^g$ the procedure creates a *AND node* $n^{g,m}_{AND}$ connected to n^g_{OR} through the edge $\langle n^g_{OR}, n^{g,m}_{AND}\rangle$ (rows 11-14). The creation of this *AND node* is necessary because a method can be be associated with multiple `ProductionTask`, all representing a conjunctive decomposition of the original production goal g. Given a particular individual m of `ProductionMethod` the procedure extract_constituents $(\mathcal{KB})$ retrieves all associated individuals of `ProductionTask` T^m (row 15). For each associated task $t \in T^m$ thus a new *task node* n^t is created and associated with the *AND node* of the related method $n^{g,m}_{AND}$ through the edge $\langle n^{g,m}, n^t\rangle$ (rows 16-19). At this point, the graph $\mathcal{G}$ is refined going deeper into the procedural decomposition of each `ProductionTask` t through the recursive procedure graph_refinement $(\mathcal{KB}, \mathcal{G}, t)$ described by Algorithm 4 (row 20).
- In the second case, only one method $m = M^m[0]$ is found and therefore there is no need to encode a disjunctive decomposition of `ProductionGoal` g into the graph $\mathcal{G}$. It thus first creates a *AND node* n^g_{AND} associated with the root node n^g of the goal through the edge $\langle n^g, n^g_{AND}\rangle$ (rows 22-24). Then it extracts all the triples matching the pattern $\langle$?m dul:hasConstituent ?$t\angle$ in order to find the set of individuals of `ProdcutionTask` T^m to consider for the decomposition of `ProductionGoal` g (row 25). Such tasks T^m are part of a conjunctive decomposition and must all be taken into account when implementing the goal g. For each task $t \in T^m$ therefore a *task node* n^t is created and is associated with the *AND node* n^g_{AND} of associated goal through the edge $\langle n^g_{AND}, n^t$ [6]. At this point, the graph $\mathcal{G}$ is refined going deeper into the procedural decomposition of each `ProductionTask` t through the recursive procedure graph_refinement $(\mathcal{KB}, \mathcal{G}, t)$ described by Algorithm 4 (row 30).

Algorithm 4 recursively refines the graph $\mathcal{G}$ by navigating the property `DUL:hasConstituent` between individuals of `ProductionTask`. As can be seen, the procedure starts from an individual of `ProductionTask` t and it distinguishes among three cases depending on the particular type of task t:

- If the task t is of type `DisjunctiveTask` (row 1) then the graph $\mathcal{G}$ is refined interpreting sub-tasks of t as disjunctive decomposition (rows 1-10). The procedure creates a *OR node* n^t_{OR} connected to the task node n^t of the starting task t through the edge $\langle n^t, n^t_{OR}\rangle$ (rows 2-4). The sub-tasks T of t are then extracted navigating the property `DUL:hasConstituent` (row 5). Specifically the procedure extract_constituents $(\mathcal{KB}, t)$ retrieves all the triples matching the pattern $\langle$?t dul:hasConstituent ?$t'\rangle$. For each sub-task $t' \in T$ then a task node $n^{t'}$ is created and connected to the disjunctive node n^t_{OR} through the edge $\langle n^t_{OR}, n^{t'}\rangle$ (rows 6-9). A recursive call to the procedure then refines the graph $\mathcal{G}$ deeper navigating the procedural description through the sub-task t' (row 10).
- If the task t is of type `ConjunctiveTask` (row 11) then the graph $\mathcal{G}$ is refined interpreting sub-tasks of t as conjunctive decomposition (rows 12-20). The procedure creates a *AND node* n^t_{AND} connected to the task node n^t of the starting task t through the edge $\langle n^t, n^t_{AND}\rangle$ (rows 12-14). The sub-tasks T of t are then extracted navigating the property `DUL:hasConstituent` (row 15). Specifically the procedure extract_constituents $(\mathcal{KB}, t)$

[6] There is no need to explicitly add a node associated with the individual of `ProductionMethod` m since there is only one method $|M^g| == 1$ in this case. Namely, all individuals of `ProductionTask` associated with m must be considered into a conjunctive decomposition of `ProductionGoal` g.

Algorithm 4 Recursive refinement of the decomposition graph navigating the task procedural implementation into the knowledge base

Require: $\langle \mathcal{KB}, \mathcal{G}, t \rangle$ ▹ Production knowledge, decomposition graph and the `ProductionTask` t to decompose
1: **if** is $(\mathcal{KB}, t, \texttt{DisjunctiveTask})$ **then** ▹ Interpret the decomposition of task t as disjunctive
2: $n^t_{OR} \leftarrow$ create_or_node ()
3: $\mathcal{N}_T \leftarrow \{n^t_{OR}\}$
4: $\mathcal{E}_T \leftarrow \langle n^t, n^t_{OR} \rangle$
5: $T \leftarrow$ extract_constituents $(\mathcal{KB}, t)$
6: **for** $t' \in T$ **do**
7: $n^{t'} \leftarrow$ create_node (t') ▹ Task decomposition of t is interpreted as disjunctive
8: $\mathcal{N}_T \leftarrow \{n^{t'}\}$
9: $\mathcal{E}_T \leftarrow \langle n^t_{OR}, n^{t'} \rangle$
10: grap_refinement $(\mathcal{KB}, \mathcal{G}, t')$ ▹ Recursive call to Algorithm 4 to navigate the procedure deeper
11: **else if** is $(\mathcal{KB}, t, \texttt{ConjunctiveTask})$ **then** ▹ Interpret the decomposition of task t as conjunctive
12: $n^t_{AND} \leftarrow$ create_and_node ()
13: $\mathcal{N}_T \leftarrow \{n^t_{AND}\}$
14: $\mathcal{E}_T \leftarrow \langle n^t, n^t_{AND} \rangle$
15: $T \leftarrow$ extract_constituents $(\mathcal{KB}, t)$
16: **for** $t' \in T$ **do**
17: $n^{t'} \leftarrow$ create_node (t')
18: $\mathcal{N}_T \leftarrow \{n^{t'}\}$
19: $\mathcal{E}_T \leftarrow \langle n^t_{AND}, n^{t'} \rangle$
20: graph_refinement $(\mathcal{KB}, \mathcal{G}, t')$ ▹ Recursive call to Algorithm 4 to navigate the procedure deeper
21: **else**
22: ▹ individual of `SimpleTask` are leaves of the graph and no more decomposition is necessary

retrieves all the triples matching the pattern $\langle ?t$ dul:hasConstituent $?t' \rangle$. For each sub-task $t' \in T$ then a task node $n^{t'}$ is created and connected to the conjunctive node n^t_{AND} through the edge $\langle n^t_{AND}, n^{t'} \rangle$ (rows 16-19). A recursive call to the procedure then refines the graph $\mathcal{G}$ deeper navigating the procedural description through the sub-task t' (row 20).

– If the task t is of type `SimpleTask` (rows 21-22) no further refinement of the graph $\mathcal{G}$ is necessary. This is the *base case* stopping the recursive navigation of the production procedure in $\mathcal{KB}$.

Recalling Algorithm 1, the graph $\mathcal{G}$ generated by Algorithm 3 characterizes the decomposition of `ProductionGoal` into `ComplexTask` and `SimpleTask` through associated `ProductionMehtod` (row 9). The sub-procedure hierarchy $(\mathcal{G})$ creates a hierarchical layering of `ProductionTask` T running a *topological sort algorithm* on $\mathcal{G}$ (row 10). These two auxiliary data structures are then supports the definition of the remaining state variables and synchronization rules of the timeline-based model (respectively rows 11-16 and rows 17-29).

The hierarchical layering of `ProductionTask` T obtained from the graph $\mathcal{G}$ partitions the set of tasks in H sets each describing modeled procedures at a certain level of abstraction. Taking into account the structure of the graph $\mathcal{G}$, the higher level of layering (i.e., the sub-set at $H = 0$) corresponds to the root nodes of $\mathcal{G}$ and thus to the individuals of `ProductionGoal` extracted from the knowledge base. This set of goals is already encoded into SV_G and therefore the construction of intermediate procedural-level state variables starts from the next layering level $L \leftarrow 1$ (row 11). For each layering $L \leqslant H$ of the procedure a new state variable SV_L is created (rows 12-16). The procedure extracts the set of individuals of `ProductionTask` $T^L = \{t^L_1, ..., t^L_k\}$ at the considered level L from the hierarchical layering T (row 13). This set of individuals is then used to create a state variable SV_L through the general procedure described in Algorithm 2. The resulting state variable SV_L contains values characterizing production procedures at level of abstraction L. How these variables are correlated to the goal state variable SV_G, to other production-level state variables $SV_{L'}$ and to the human and robot state variables SV_H and SV_R depends on the structure of $\mathcal{G}$ and is encoded into the synchronization rules of the model generated in the next steps of Algorithm 1 (rows 17-30).

The definition of the synchronization rules starts with defining the rules that constrain the decomposition of goals (i.e., values of SV_G) into increasingly simple production tasks (i.e., values of SV_L, with growing values of L). The procedure iterates over the hierarchical layering of T extracted from $\mathcal{G}$ (rows 18-30). It starts with the individuals

composing the top level of the hierarchy $L \leftarrow 0$ (i.e., the individuals of `ProductionGoal`) and stops with the individuals composing the last hierarhical level $L = H$. At each hierarchical level L the procedure retrieves the set of individuals of production tasks or goals $T^L = \{t_1^L, ..., t_k^L\}$ that compose the associated level of the hierarchy (row 19). The it continues creating needed *synchronization rules* for each task of the current hierarchical level $t^L \in T^L$ (rows 20-29). Namely, the procedure defines a number of synchronization rules where the values associated with the individuals $t^L \in T^L$ (i.e., the state variable values of the corresponding SV_L) are the *triggerers* of the rule (see definition of synchronization rules in Equation 5). Recall that multiple synchronizations with the same triggerer are treated as disjunctive refinements of plans (e.g., disjunctive decomposition of tasks).

The number and "shape" of the defined synchronization rules depend on the particular type of the individual $t^L \in T^L$. Two cases may occur depending on whether the t^L represents a complex task or not:

- In case that t^L is a `ComplexTask` or a `ProductionGoal` then the procedure should take into account all possible decomposition and create a number of synchronization rules representing available alternatives (rows 21-25). The procedure subtasks $(\mathcal{G}, t^L)$ extracts the sets of individuals representing subtasks decomposing t^L. Subtasks $T^{L'}$ are obtained extracting from $\mathcal{G}$ the sets of task nodes reachable from the task node of t^L through connected *AND nodes* and *OR nodes* (row 22). The sets of task nodes reached through *OR nodes* represent alternative decomposition of t^L and thus alternative sets of tasks $T^{L',i} = \{t_1^{L',i}, ..., t_n^{L',i}\}$ that should be considered to decompose t^L. For each of such set $T^{L',i} \in T^{L'}$ the procedure creates a dedicated synchronization rule R^L (rows 23-25). These rules have all the same the triggerer t^L but different body $T^{L',i}$ and are treated as disjunctive refinements of the state variable value associated with t^L.
- In case that t^L is a `SimpleTask` then no disjunctive decomposition may occur since `SimpleTask` are directly associated with individuals of `Function` that implement them (see Equation 15). The procedure creates a single synchronization rule constraining task t^L to associated functions (rows 26-29). In this case, given an individual of `SimpleTask` t^L the procedure extracts individuals of `Function` associated through the property `DUL:hasConstituent` (row 27). The obtained set of individuals F thus represents the set of `Function` the human and/or the robot should execute to correctly implement the task t^L. Given this information, the procedure creates the synchronization rule R^L encapsulating the associated causal and temporal constraints (rows 28-29).

Algorithm 5 shows the detailed procedure creating synchronization rules. The procedure follows the formal definition of synchronization rule of Equation 5. It receives as input an individual of `ProductionTask` t and a set of individuals of `ProductionTask` T representing associated subtasks. The resulting description of a synchronization rule consists of three components: (i) the triggerer or head of the rule R_H; (ii) the body of the rule R_B and; (iii) the set of temporal constraints R_C. It first creates the description of the triggerer from the `ProductionTask` t (row 2). For each subtask $t' \in T$ (rows 3-5) the procedure creates a dedicated *body variable* added to the body of the rule R_B (row 4) and add a temporal relation of type `CONTAINS` [94] between t and $t' \in T$ to the set of temporal constraints R_C. This constraint enforces the hierarchical decomposition and to execute task t' within the execution of the "parent task" t. The description of a `ProductionTask` t may further constrain the execution of associated subtasks $t' \in T$ through `ProductionNorm`. The procedure specifically focuses on `PrecedenceConstraint` in order to extract the set of precedence constraints PC specified within the task (row 6). For each precedence constraint $c \in PC$ found it creates suitable temporal constraints between the tasks of the body of the rule R_B. A `PrecedenceConstraint` specifies the `ProductionTask` to execute as first through the property `hasFirstTask` and the `ProductionTask` to execute as second through the property `hasSecondTask`. These two properties are thus used to extract information about the individual of `ProductionTask` constrained to be first t' and second t'' (rows 8-9). Given this information, the procedure adds a temporal relation of type `BEFORE` [94] between the first task t' and the second task t'' to the set of constraints of the rule R_C (row 10).

Algorithm 5 shows the procedure used to generally creates a synchronization rule modeling the hierarchical decomposition of a `ProductionTask` into simpler `ProductionTask` (i.e., subtasks). When the "parent task" t is of type `SimpleTask` the associated subtasks T are composed by `Function` only (see the definition of `SimpleTask` in Equation 15). Depending on the particular type of t, ontological patterns could facilitate the definition of rules. In case of collaborative tasks indeed ontological patterns determine specific combinations of constraints between associated `Function`.

Algorithm 5 Definition of synchronization rules constraining decomposition of a task into simpler tasks

Require: $\langle \mathcal{KB}, t, T \rangle$ ▹ Production knowledge, production task, set of simpler tasks
Ensure: $\mathcal{R} : \langle R_H, R_B, R_C \rangle$ ▹ Rule composed by a head/triggerer, a body and a set of constraints as defined in 5

```
R_H, R_B, R_C ← {}
R_H ← create_triggerer (KB, t)
for t' ∈ T do
    R_B ← create_body_variable (KB, t')
    R_C ← create_contains_constraint (t, t')
PC ← extract_norms (KB, t, PrecedenceConstraint)
for c ∈ PC do                                   ▹ Check norms specifying precedence constraints between sub-tasks
    t' ← first (KB, c)
    t'' ← second (KB, c)
    R_C ← create_precedence_constraint (t', t'')
return R : ⟨ R_H, R_B, R_C ⟩
```

- In case of a collaborative task t of type `IndependentHRCTask`, according to Equation 18, the set of subtasks T is composed by only one individual of `Function` that can be associated either with the human or the robot agent (i.e., an individual of `WorkOperator` or an individual of `Cobot`). The created synchronization rule contains only one `CONTAINS` temporal constraint between the triggerer state variable value associated with t and the single body human/robot state variable value associated with function $f \in T$, where $|T| = 1$. Listing 2 shows an example of a synchronization rule decomposing a `HRCTask` associated with an interaction modality of type `Independent` (i.e., a `IndependentHRCTask`).
- In case of a collaborative task t of type `SimultaneousHRCTask`, according to Equation 19, the set of subtasks T is composed by two individuals of `Function`, one executed by a human agent (i.e., individual of `WorkOperator`) and one executed by a robot agent (i.e., individual of `Cobot`). The created synchronization rule contains two `CONTAINS` temporal constraints, one between the triggerer state variable value associated with t and the human state variable value associated with the function $f_H \in T$ and, one between the triggerer state variable value associated with t and the robot state variable value associated with function $f_R \in T$, where $|T| = 2$. No additional constraints are necessary between the two functions to enforce the desired collaboration modality. The two functions are only constrained to be executed during the execution of a collaborative task t allowing thus their overlapping over time. Listing 3 shows an example of synchronization rule decomposing a `HRCTask` associated with an interaction modality of type `Simultaneous` (i.e., a `SimultaneousHRCTask`).
- In case of a collaborative task t of type `Sequential`, according to Equation 20, the set of subtasks T is composed by two individuals of `Function`, one executed by a human agent (i.e., individual of `WorkOperator`) and one executed by a robot agent (i.e., individual of `Cobot`). Furthermore, the task t is associated with a production norm of type `PrecedenceConstraint` which specifies a function to be executed as first (it uses the property `hasFirstTask`) and a function to be executed as second (it uses the property `hasSecondTask`). It therefore determines the ordering of the execution of human and robot functions. The created synchronization rule contains two `CONTAINS` temporal constraints, one between the triggerer state variable value associated with t and the human state variable value associated with the function $f_H \in T$ and, one between the triggerer state variable value associated with t and the robot state variable value associated with function $f_R \in T$, where $|T| = 2$. In addition, the rule contains a `BEFORE` temporal constraint which enforces the state variable value associated with the *first task* to be executed before the state variable value associated with the *second task*, according to the declared `PrecedenceConstraint`. Listing 4 shows an example of a synchronization rule decomposing a `HRCTask` associated with an interaction modality of type `Sequential` (i.e., a `SequentialHRCTask`).
- In case of a collaborative task t of type `Supportive`, according to Equation 21, the set of subtasks T is composed by two individuals of `Function`, one executed by a human agent (i.e., individual of `WorkOperator`) and one executed by a robot agent (i.e., individual of `Cobot`). The two functions are in this case forced to be executed at the very same time and thus with a strong parallelism. As can be seen from Equation 21, an interaction modality of type `Supportive` is associated with a production norm of type

`ParallelExecutionConstraint` enforcing a strong parallelism among implementing functions. However, the semantics of such collaborative tasks is not ambiguous and synchronization rules can be defined without reading associated production norms [7]. The created synchronization rule contains two `EQUALS` temporal constraints, one between the triggerer state variable value associated with t and the human state variable value associated with the function $f_H \in T$ and, one between the triggerer state variable value associated with t and the robot state variable value associated with function $f_R \in T$, where $|T| = 2$. These two constraints enforce a strong parallelism between the task and the functions. In particular it enforces the two functions to be executed during the same (flexible) temporal interval, starting and and ending their execution at the very same time. Listing 5 shows an example of a synchronization rule decomposing a `HRCTask` associated with an interaction modality of type `Supportive` (i.e., a `SupportiveHRCTask`).

Listing 2: Example of a synchronization rule decomposing an `IndependentHRCTask`

```
SYNCHRONIZE ProductionStateVariable {           // Synchronization associated with a production state variable

    VALUE value_t() {                           // state variable value denoting a simple task

        func HumanStateVariable.value_2();      // decomposition requiring a human function

        CONTAINS [0, +INF] [0, +INF] func;
    }

    VALUE value_t() {                           // state variable value denoting a simple task

        func RobotStateVariable.value_3();      // decomposition requiring a robot function

        CONTAINS [0, +INF] [0, +INF] func;
    }
}
```

Listing 3: Example of a synchronization rule decomposing an `SimultaneousHRCTask`

```
SYNCHRONIZE ProductionStateVariable {           // Synchronization associated with a production state variable

    VALUE value_t() {                           // state variable value denoting a simple task

        h_func HumanStateVariable.value_function_3();
        r_func RobotStateVariable.value_function_1();

        CONTAINS [0, +INF] [0, +INF] h_func;
        CONTAINS [0, +INF] [0, +INF] r_func;
    }
}
```

Listing 4: Example of a synchronization rule decomposing an `SequentialHRCTask`

```
SYNCHRONIZE ProductionStateVariable {           // Synchronization associated with a production state variable

    VALUE value_t() {                           // state variable value denoting a simple task

        h_func HumanStateVariable.value_function_2();
        r_func RobotStateVariable.value_function_5();

        CONTAINS [0, +INF] [0, +INF] h_func;
        CONTAINS [0, +INF] [0, +INF] r_func;

        h_func BEFORE [0, +INF] r_func;         // decomposition requiring to execute human function first
    }


```

[7] This is a different case from an interaction modality of type `SequentialHRCTask` where it is not possible to know in advance the ordering specified between functions.

```
        VALUE value_t() {                          // state variable value denoting a simple task

            h_func HumanStateVariable.value_function_4();
            r_func RobotStateVariable.value_function_2();

            CONTAINS [0, +INF] [0, +INF] h_func;
            CONTAINS [0, +INF] [0, +INF] r_func;

            r_func BEFORE [0, +INF] h_func;        // decomposition requiring to execute robot function first
        }
    }
```

Listing 5: Example of a synchronization rule decomposing an `SupportiveHRCTask`

```
    SYNCHRONIZE ProductionStateVariable {          // Synchronization associated with a production state variable

        VALUE value_t() {                          // state variable value denoting a simple task

            h_func HumanStateVariable.value_function_3();
            r_func RobotStateVariable.value_function_1();

            EQUALS h_func;
            EQUALS r_func;
        }
    }
```

5. Assessment on Real-World Scenarios

The feasibility of the ontology-based representation and reasoning approach proposed here has been assessed on a set of real HRC scenarios from the pilot use cases of the EU H2020 Sharework project [8]. The ontology has been defined in OWL [60], using Protégé [9], and is publicly available on a GitHub repository [10] For each use case a dedicated knowledge base has been defined by gathering the information from production engineers and domain experts.

In absence of a suitable knowledge engineering tool, a number of specifically designed forms have been administrated to domain experts in order to collect data and properties suitable for the definition of each knowledge base. Collected information has been then used by a knowledge engineer to concretely build the knowledge bases of the different scenarios, using Protégé [11]. The reasoning mechanisms and the knowledge extraction procedure of Algorithm 1 have been developed in Java using Apache Jena [12]. Representation and reasoning functionalities have been integrated into ROS [13] through ROSJava [14] to support deployment on (industrial) robots and and implement the envisaged cognitive control loop. ROS modules and services developed within Sharework are publicly available on GitHub [15].

The evaluation considers different collaborative scenarios representing realistic production situations, needs and constraints. Such scenarios are well suited to assess the generality of the proposed ontological model as well as its efficacy in capturing the requirements of real-world applications and synthesizing valid task planning models. The scenarios are the following: (i) AUTOMOTIVE; (ii) METAL; (iii) CAPITAL-GOODS; (iv) RAILWAYS. These four scenarios are extracted from the pilots of Sharework and characterize actual production processes. They constitute a realistic benchmark to assess the proposed reasoning capabilities of the developed AI-based technologies. In addition, we consider an additional scenario where task allocation and possible alternative behaviors of the human and the

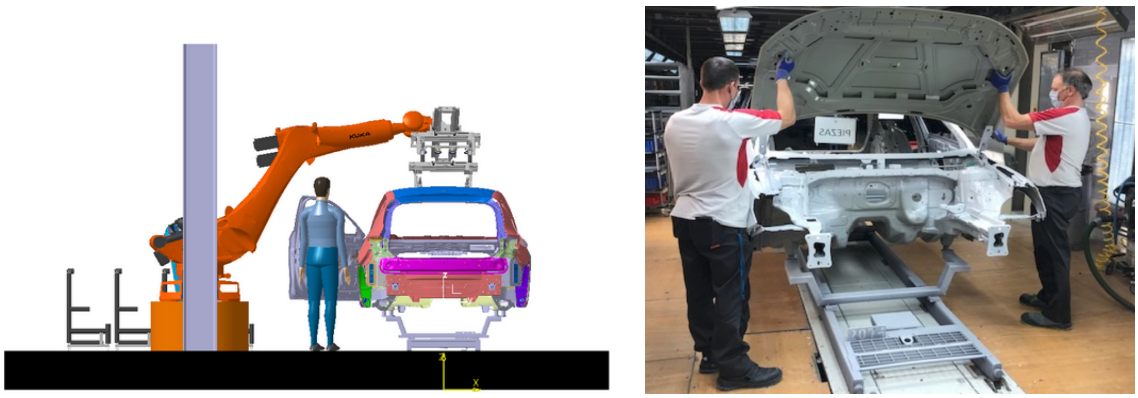
[8] https://sharework-project.eu
[9] https://protege.stanford.edu
[10] https://github.com/pstlab/SOHO.git
[11] Defined models can be found on the GitHub repository of SOHO under the folder “instances”.
[12] https://jena.apache.org
[13] Distribution ROS Melodic - http://wiki.ros.org/melodic
[14] http://wiki.ros.org/rosjava
[15] https://github.com/pstlab/sharework_knowledge.git

robot are largely more variable. This scenario is called MOSAIC and takes inspiration from a typical collaborative assembly scenario [91]. Although it does not correspond to a concrete production process, MOSAIC describes a highly flexible (collaborative) production process entailing the representation of various possible (alternative) behaviors of the robot and the human into the knowledge and the resulting task planning model.

5.1. Industrial Scenarios

5.1.1. The AUTOMOTIVE Scenario

This scenario takes into account a specific station of an assembly line of vehicles. The considered collaborative process specifically focuses on a door assembly task of chassis. The collaborative robot is in charge of moving and holding the heavy parts of the vehicle (i.e., *pick-and-place* of front and rear doors to be assembled on the chassis) while the human carries out assembly tasks in the same working-area of the robot (i.e., fix the doors to the body of the vehicle). Figure 6 shows some pictures of the layout of the working are and “mount point" of the front door on the chassis.



(a) (b)

Figure 6. Design of the collaborative cell for the AUTOMOTIVE scenario (a) and the production line for the assembly of the chassis (b).

This scenario is characterized by a flat production process where the human and the robot play different roles and carry out tasks autonomously but following a strict order. Door assembly is correctly performed only if the robot and the human execute their task correctly and at the correct time (e.g., the human cannot start her task if the robot does not place the door in the expected position). Furthermore, roles are not interchangeable therefore it would not be possible to carry out collaborative process without the correct coordination of the two actors. More specifically, the robot (a robotic arm) can perform only `PickPlace` functions on the `rear` and `front` doors of the vehicles (i.e., the `front` and `rear` doors are two `WorkPiece` of the environment). The robot (individual of `Cobot`) can perform two instances of `PickPlace`, one function on the target door `front` and the other one on the target door `rear`. The human instead carries out the actual assembly operations and should therefore perform functions of different type like e.g., `Assemble`, `ManualGuidance`, `ChangeOver` and `Screw` on the `vehicle body` (`WorkPiece`). These types of `Function` are defined in SOHO as a “specialization" of the integrated Taxonomy of Functions [59] (e.g., `Assemble` and `Join` are specialization of `Join`).

All production tasks entailing the direct involvement of human and robot operations (i.e., the `SimpleTask` of the procedure) are in this case modeled as individuals of `IndependentHRCTask` since both agents can carry out their functions in complete autonomy. Such tasks are indeed implemented by a single `Function` a human or a robot carries out *independently* from each other. For example the robotic task of moving a door to the front assembly area of the layout is modeled as and independent collaborative task (i.e., instance of `IndependentHRCTask`) and implemented by a pick-place operation of the robot (i.e., an instance of `PickPlace` function). However, `ProductionNorm` (i.e., `ExecutionNorm`) constraint sequencing some `ProductionTask` of the procedure are necessary to correctly coordinate robot and human operations. The `Assemble` functions of the human that

physically mount the doors on the chassis can be performed only if the robot has correctly placed the door in the expected position. Individuals of `PrecedenceConstraint` are thus necessary to constrain the execution of `PickPlace` functions of the robot to occur before the execution of `Assemble` functions of the human.

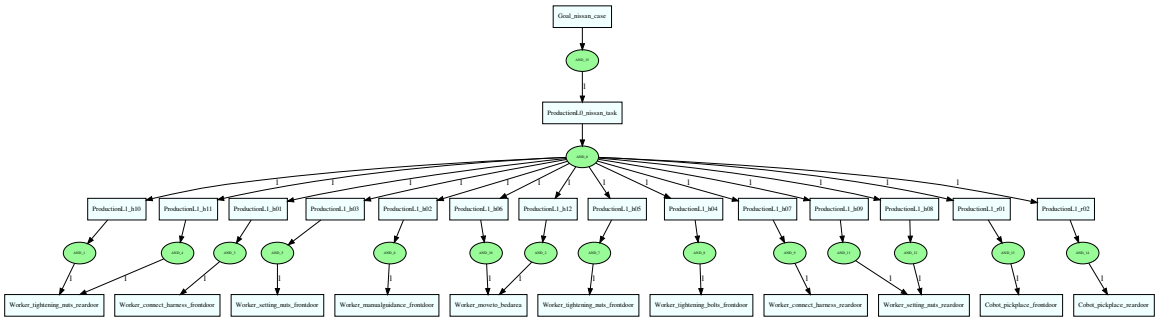


Figure 7. Decomposition graph of the production process of AUTOMOTIVE. The top-level goal is decomposed into a single `ComplexTask` which is in turn decomposed into a number of `IndependentHRCTask` that compose the state variable `ProductionL1` shown in the hierarchy of Figure 16(a). Each `IndependentHRCTask` entails the execution of a `Function` by the human or by the robot.

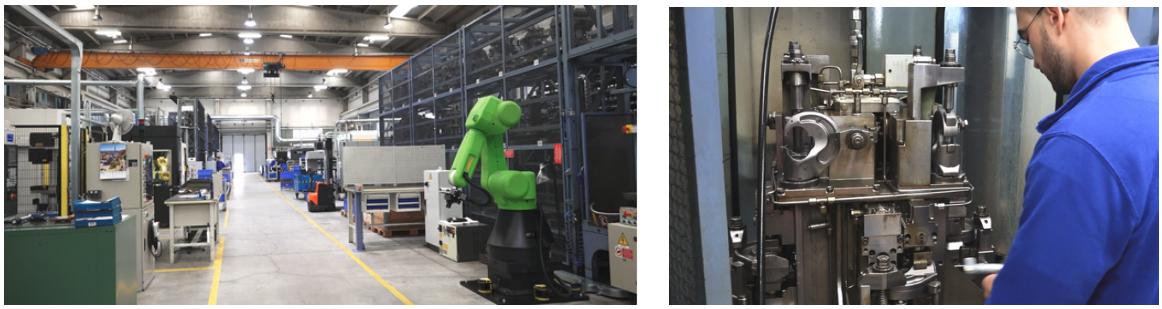
Figure 7 shows the task decomposition graph extracted from the knowledge base through Algorithm 1. Although simple, the graph shows the hierarchical layering of the modeled production procedure. This structure is also highlighted by Figure 16(a) showing the hierarchical structured of the generated (timeline-based) task planning model. According to the general description of Algorithm 7 the top-level elements of the hierarchy are associated with state variable values representing individuals of `ProductionGoal`. The low-level elements of the hierarchy correspond to the state variable values representing the functions the human and the robot can perform. Intermediate levels are instead associated with state variable values representing the hierarchical decomposition of complex tasks.

The production process of this scenario is simple from a control perspective since possible behaviors of the human and the robot are fixed and there is not need for optimization. It just requires the unfolding of a single `ComplexTask` into a number of `SimpleTask` the human and the robot perform sequentially. Each simple task is specifically modeled as `IndependentHRCTask` requiring the execution of a specific `Function` by the human or by the robot. Although simple, the integration of developed representation and planning capabilities contributes to facilitate human-robot interactions. The designed cognitive control approach would provide the human worker with detailed information about requested tasks, when dispatched by the task planner. Furthermore, the system informs the worker about tasks assigned to the robot and when they are going to be executed. This information allows the worker to know his/her plan and the current (and planned) behavior of the robot. In this regard, the human factor context of the ontology supports the customization of the interactions with the human. Information details about tasks as well as the requested feedback can indeed change according to the level of expertise and the preferences of the involved worker [96]. For example, workers with low expertise may benefit from receiving frequent and detailed information about tasks and the collaborative plan. Expert workers instead may found an excessively frequent and detailed level of information annoying.

5.1.2. The METAL *Industrial Scenario*

This scenario takes into account the logistic station of the manufacturing system of electrical connectors. The workshop for the assembly of pallets and fixtures in load/unload stations is divided into two main areas: (i) a transporter panel buffer, where pallets are stored and moved, and; (ii) some CNC (Computerized Numerical Control) machines, where the pallets are moved to perform the machining operations. In this scenarios operators are generally responsible for transporting pallets and components to be mounted in a *tombstone* that goes inside the Flexible Manufacturing System where each part is machined. The scenario is characterized by a high variability of parts to be produced. Operators therefore should be highly trained in order to correctly perform the suitable assembly procedure for each different product as well as perform the quality inspection on the pallets before and after machining. The

collaborative robot is in charge of assisting operators when moving across the station, understanding operator's behavior and anticipating tasks in order to facilitate their work and speedup the production, i.e., increasing the throughput.



(a) (b)

Figure 8. Structure of the shop-floor of the METAL industrial scenario (a) and the fixturing system where collaboration takes place (b).

Figure 8 shows some pictures of the layout of the logistic station. The working area is characterized by a central/shared conveyor where different types of products are loaded and processed in order to be machined. The worker and the robot (a UR10 robotic arm) are placed at two distinct sides of the conveyor and works simultaneously on the products. Products are placed and move on the central conveyor which represents the shared working area where operations take place and where the human and the robot physically interact. The considered production process is characterized by different types of operations depending on the specific types of workpiece entering the collaborative cell. From task planning perspective, the structure of the assembly/disassembly process is similar in all the cases, since the human and the robot can have the same capabilities (e.g., screw, unscrew, pick, place, etc.). Unlike the AUTOMOTIVE scenario, here the worker and the robot are two interchangeable agents capable of performing the same tasks. The human and the robot represent therefore two autonomous peer actors that work simultaneously on the workpieces performing assembly/disassembly tasks. The synthesis of collaborative processes thus concerns the correct allocation of tasks to these two *resource* (i.e., autonomous agents).

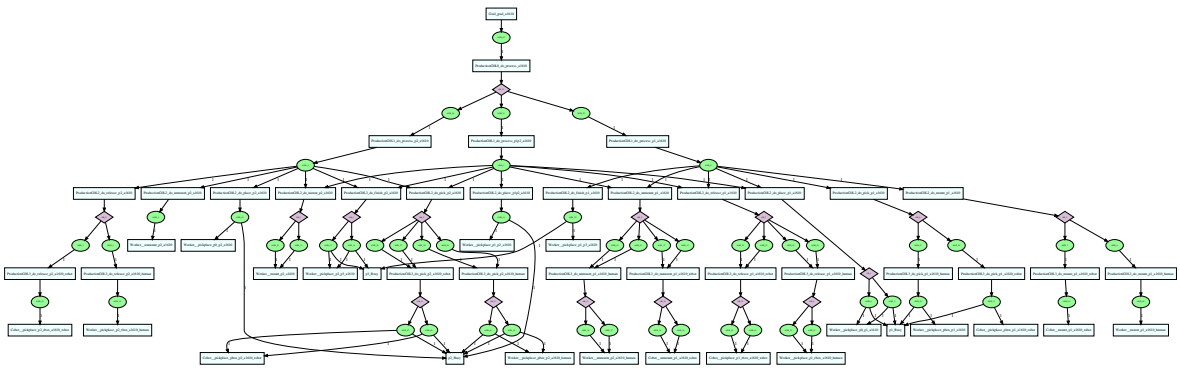


Figure 9. Decomposition graph of the production process of METAL. The top-level goal is decomposed into a single `ComplexTask` which is further decomposed into a number of `IndependentTask` that compose the state variable `ProductionL1` shown in the hierarchy of Figure 16(a). Each `IndependentTask` entails the execution of a `Function` by the human or by the robot.

Figure 9 shows the decomposition graph with respect to three possible workpieces, each associated with a different individual of `ProductionGoal` (i.e., the root nodes of the graph). The resulting production procedures entail a number of tasks requiring the execution of `PickPlace` and `Assemble`/`Disassemble` functions by the human and/or the robot. The process consists in picking and placing workpieces transported on a pallet over a conveyor. Each workpiece is placed over a pallet which enters the the conveyor from an initial position (`position0`).

Broadly speaking, the procedure consists in replacing the workpiece transported by the pallet. There are then two working positions where the pallet can be moved (`position1` and `position2`). The human moves the pallet to one of the two working position. Here the pallet is first unmounted in order to release the worked workpiece which is then placed into a dedicated box. Then, a new raw workpiece is picked from another box and mounted into the pallet. At this point, the human moves the pallet to the last position of the conveyor (`position3`) where it can be sent to other stations of the shop-floor to be machined. Although the general procedure is the same for all the types of workpieces considered within the pilot, the geometry of the workpieces and the pallets are different. For this reason, each type of workpiece is associated with a dedicated `ProductionGoal` and thus different individuals of `ProductionTask` and `Function`. This fine grained distinction allows the resulting cognitive control approach to considered the specific geometries of the different pieces and properly inform both the human and the robot controller (e.g., the motion planner) about the specific low-level operations the should perform.

The replacement of a workpiece on a pallet is realized through pick & place operations that can be performed by both the human and the robot. The human or the robot `Pick` the current `WorkPiece` from the "base" and `Place` it into available containers e.g., `Box-A`, `Box-B` that are modeled as `CapacityResource` (i.e., they can "store" a limited number of `WorkPiece`). In this case functions of type `Pick` and `Place` are represented separately (i.e., not as atomic instances of `PickPlace`) in order to capture operational requirements and properly characterize possible planning choices (e.g., which box to use to place a workpiece). The production knowledge and the resulting decomposition graph of Figure 9 contain several `DisjunctiveTask` (OR nodes) representing alternative decomposition choices. Each task can be performed either by the worker or by the robot. Such tasks are represented as `DisjunctiveTask` and decomposed into human and robotic pick & place `ComplexTask` through OR nodes. Such tasks are further decomposed into two distinct pick and place `ProductionTask` to characterize the specific `Pick` and `Place` functions the worker or the robot should perform. Tasks that concern pick operations are generally represented as `IndependentHRCTask` implemented instances of `Pick` functions. Similarly, tasks that concern place operations are represented as `DisjunctiveTask`. Each decomposition represents an alternative choice of assigning task of placing a `WorkPiece` into one of the available boxes (i.e., `Box-A` and `Box-B`) to the human or to the robot. Such tasks are represented as `IndependentHRCTask` and implemented by individuals of `Place` functions that can be performed by the human or by the robot.

The integration of developed representation and planning capabilities contributes to the optimization and safe coordination of human and robot behaviors. The designed cognitive control approach would automatically optimize human and robot behaviors through the generated (timeline-based) task planning model. Domain experts describing the desired function and requirements of the production scenario are implicitly specifying a task planning model coordinating human and robot behaviors. The automation introduced by Algorithm 1 facilitates the design and deployment of task planning deliberative control without the involvement of a planning expert thus reducing the risk of modeling errors [97]. The resulting task planner then would take into account described human and robot capabilities (in terms of functions they can perform) and assign tasks to the human and the robot (e.g., pick & place functions) in order to optimize the resulting collaboration with respect to safety and performance [22, 91]. Furthermore, similar to AUTOMOTIVE, the cognitive control loop would provide workers with customized information about the tasks he/she should perform. This is especially relevant in this scenario involving a high variety of workpieces and pallets, each with different geometries and low-level operations. For example, the way `Assemble` or `Disassemble` functions are actually implemented by the worker (or by the robot) may significantly change according to the specific shape of the workpiece/pallet. Detailed information enriching dispatched tasks and showing for example contextualized instructions about required manipulation operations would help the human in doing her/his jobs.

5.1.3. The CAPITAL-GOODS *Industrial Scenario*

This scenario takes into account the shop-floor of a company offering differential and global solutions in power transmission and spraying components. This scenario considers a servo rotary table that is assembled in seven fixed assembly stations. In each station there is an operator performing a specific task of the assembly process. All tasks are carried out manually, just using cranes and lifters to transport the heavy components from one station to another. The collaboration between the human and the robot concerns three out of seven tasks of the rotary table assembly process. In the current assessment we specifically consider the task *bolt tightening and torque measuring*. Figure

10 shows the designed physical environment with the rotary table equipped with the collaborative robot (a UR10 robotic arm). The operator applies adhesives on the bolts of the rotary table to allow the robot to simultaneously determines their position and dimension through perception modules developed with the project. Information about detected bolts is then used by the robot to automatically screw them.

(a) (b)

Figure 10. Working area of the CAPITAL-GOODS scenario (a) and structure of the workpiece (b) where a collaborative robot is deployed to support workers in repetitive screwing/unscrewing tasks.

The developed cognitive approach supports a *reactive* behavior where the robot relies on *perception capabilities* [98] to autonomously recognize human tasks (i.e., “bolt placing" tasks) and act accordingly. The developed knowledge representation capabilities indeed allow the robot to recognize new events concerning the placement of bolts by the human and interpret these events as signal triggering the execution of robot tasks (i.e., `ProductionGoal`). On the one hand, knowledge reasoning thus supports continuous adaptation of robot behaviors by automatically triggering planning requests based on the observed state of the human, and execute suitable tasks. On the other hand, knowledge reasoning validates perception outcome by checking the validity of a detected human task. It could be the case for example that a perception module (wrongly) detects a placing tasks of a bolt already placed. In such a case the knowledge base would detect this inconsistency when interpreting the observation and no `ProductionGoal` would be triggered to the task planner.

The reactive behavior described above represents the actual way the developed knowledge representation and reasoning modules have been deployed into this pilot case. To evaluate the representation capabilities of the ontological model and the task planning model, we here model the whole production process assuming a deliberative approach similar to other use cases. Namely, we do not consider perception outcomes and assume the planner should synthesize screwing tasks of all bolts of the workpiece. The process thus consists of screwing a number $N = 8$ of bolts distributed over a rotary table. We keep the joint collaboration implemented in the real scenario and thus assume that a worker places the bolts on the holes of the rotary table and a robot screws them simultaneously. Although simple, an interesting aspect of this scenario is the *synchronization* required between the human and the robot. The robot can start screwing a bolt only after the worker has placed the bolt on one of the holes of the rotary table. This means that production task of “screwing a bolt" should implement a `SequentialHRCTask`: (i) required human and robotic functions have the same target i.e., a particular *hole-X* that can be seen as a `SimpleWorkPiece` composing the *CompoundWorkPiece rotary table*; (ii) robot and human operations should follow a strict temporal ordering. In this case the `Screw` function of the robot must always be performed *after* the `PickPlace` function of the human has been executed.

Figure 11 shows the decomposition graph of the described production process. The high-level production goal `rotary-table-assembly` is decomposed into a number of (complex) `ConjunctiveTask`, one for each *hole* of the rotary table to be screwed. Each simple task (i.e., screwing tasks) is represented as `SequentialHRCTask` and thus decomposed into two functions. A `PickPlace` function like e.g., `pick-place-H3` requiring the operator to pick and place a bolt on a particular hole (e.g., `H3` represented as `SimpleWorkPiece`). A `Screw` function like e.g., `screw-H3` requiring the operator to screw the bolt placed by the worker. The ontological pattern describing this collaborative behavior is translated into a well-defined structure of temporal constraints into the task planning model. As shown in 4, the pattern determines a *synchronization rule* requiring; (i) the state variable of the

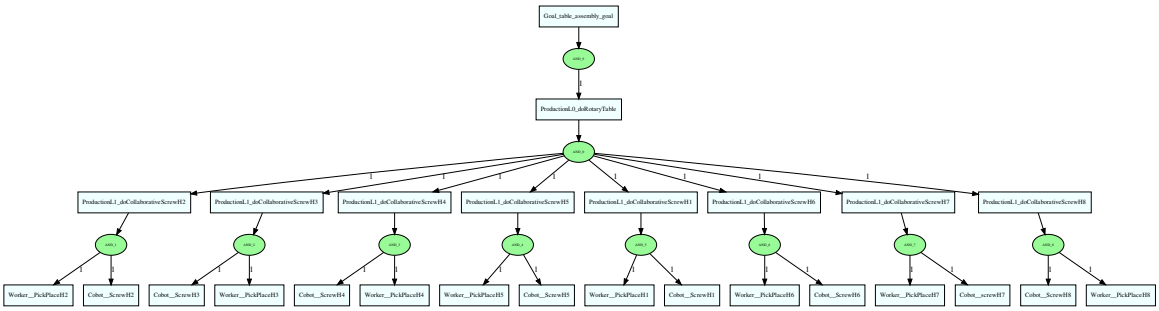


Figure 11. Decomposition graph of the production process of CAPITAL-GOODS.

worker to assume the value associated to function of picking and placing a bolt on `hole-X`; (ii) the state variable of the robot to assume the value associated to the function of screwing a bolt on `hole-X`; (iii) to schedule (and thus execute) the value of the human *before* the value of the robot.

5.1.4. The RAILWAYS *Industrial Scenario*

This scenario takes into account the shop-floor of a railways transportation company supplying rolling stocks, services and system infrastructure. The workshop is composed by six main stations each one dedicated to a specific set of operations concerning the assembly of trains. The project specifically focuses on the pre-assembly process of tram-way's windows and door frames. Among the tasks involved into the considered processes, *riveting* represents a repetitive and demanding task for human workers. It consists of the insertion of rivets in drilled holes along the metal pieces of window frames. This task is especially critical from *safety perspective* since it may cause significant injuries after a prolonged utilization of the riveting tool which weights up to 5 kg.

(a) (b)

Figure 12. Design of the collaborative cell of the the RAILWAYS scenario to support workers in the assembly of door frames.

Figure 12 shows the physical layout of the shop-floor sand the structure of the *window frames* that are the *target* of the considered production process. The introduction of collaborative robots into the production line is designed to *relieve* human workers from physically demanding tasks like e.g., the *riveting task* in order to improve their working condition and reduce the risk of injuries. A collaborative robot is thus supposed to work close to the window frame of Figure 12(b) and tightly collaborate with the human worker to carry out the pre-assembly tasks of window frames. The collaboration especially takes place within the *riveting task*. The human operator is in charge of spreading the silicone over the corners of the frame structure then, the robot insert the rivets using a *riveting tool*. It can be observed that the *riveting task* entails a *synchronous behavior* of the two actors since the robot can insert the rivets only *after* the worker has correctly applied the silicone. Similar to the screwing tasks of CAPITAL-GOODS, these tasks are represented as instances of `SequentialHRCTask` i.e., collaborative tasks `HRCTask` entailing a `Sequential` execution modality.

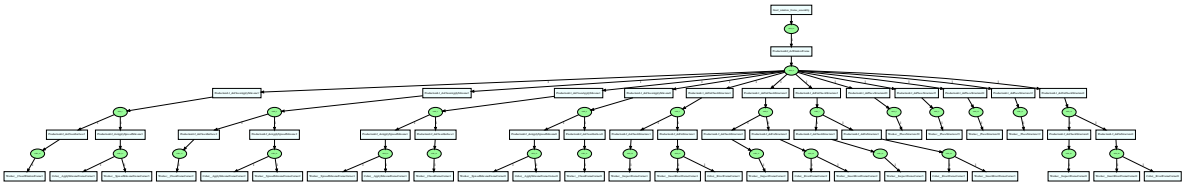
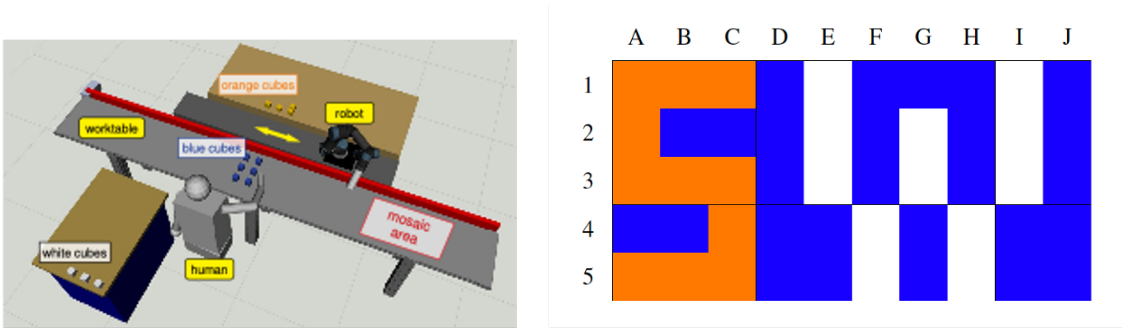


Figure 13. Decomposition graph of the production process of the RAILWAYS scenario.

Figure 13 shows the decomposition graph of the considered production process. Also in this case there are not disjunctive tasks to be considered into the decomposition since the roles and "responsibilities" of the worker and the robot are quite fixed. The key aspect in this case similar to the CAPITAL-GOOD scenario is the use of `SequentialHRCTask` to represent *riveting tasks*. Such tasks are indeed associated to a `Join` function of the worker representing the application of the silicone over a particular corner of the frame structure and, to a `Join` function of the robot representing the use of the *rivet* `Tool` to insert the rivets. Given the needed synchronization the execution of these two functions is constrained by a *before temporal constraint* and thus constrain the robot to wait for the completion of the function of the worker. As seen for the CAPITAL-GOODS scenario, such *behavioral pattern* is properly encoded into the task planing model to correctly synthesize and execute the collaborative process.

5.1.5. The MOSAIC *Collaborative Scenario*

This scenario considers a general collaborative assembly of a compound workpiece. The layout of the work-cell is characterized by a shared central space where the workpiece is placed and where the human and the robot simultaneously carry out assembly operations. Figure 14 shows the layout of the designed collaborative environment and the layout of the *mosaic*. The *mosaic* is modeled as a `CompoundWorkPiece` since it can be seen as composed by simpler parts like e.g., *rows* (still `CompoundWorkPiece`) and *cells* (`SimpleWorkPiece`).



(a) (b)

Figure 14. Configuration of the MOSAIC use case: (a) Structure of the ROS-based simulation of the collaborative cell; (b) Layout of the mosaic collaboratively assembled by the human and the robot

The collaborative process consists of the execution of a set of pick & place operations. Each pick & place is performed by a single agent autonomously but their execution (and assignment) should satisfy some physical constraints. Pick & place operations whose targets are coloured objects placed in different areas: *blue objects* can be handled by both the human and the robot; *orange objects* for short) can be performed by the robot only; *white area* (i.e., *white objects* for short) can be performed by the human only.

The MOSAIC scenario is characterized by a more articulated process where the human and the robot play the same role and carry out tasks autonomously without a specific order. The process can be seen as the problem of moving some `WorkPiece` from an initial location to a desired one in order to form a desired shape. The structure of the process is given by the structure of the desired shape. The shape can be seen as a *mosaic* composed by a certain number of *rows* (e.g., 5 rows) and a certain number of columns (e.g., 10 columns). Each specific location of the mosaic (i.e., *cell(1,1)*, ..., *cell(5,10)*) represents a specific destination of a `PickPlace`

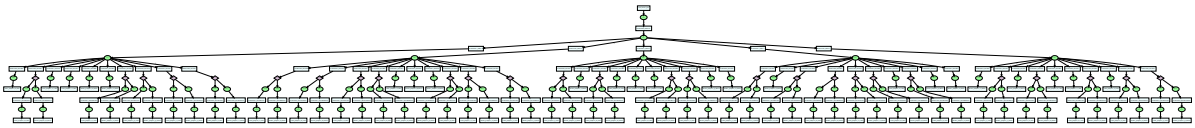


Figure 15. Decomposition graph of the production process of MOSAIC. The top-level goal `mosaic_goal` is decomposed into a number of `ComplexTask` hierarchically organized as shown in Figure 16(b). State variable `ProductionL0` concerns a single `ComplexTask doMosaic`. This task is further decomposed into a number of row-level `ComplexTask doRow1`, ..., `doRow5` that are part of state variable `ProductionL1`. Each row-level task is further decomposed into a number of `IndependetTask` like e.g., `doCellI3` and *disjunctive* `ComplexTask` like e.g., `doCellJ3`. These tasks composed the state variable `ProductionL2`. The state variable `ProductionL3` thus contains `IndependnetTask` representing alternative decomposition of disjunctive cell-level tasks. Each `IndependentTask` entails the execution of a specific `PickPlace` by the human or by the robot.

function, moving a specific `WorkPiece`. The human and the robot have the same capabilities and should perform the same type of `Function` i.e., `PickPlace`. Each `PickPlace` moves a particular instance `wp1`, ..., `wpN` of `WorkPiece` to a specific location of the mosaic like e.g., `pickplace-wp1-cell(1,1)-human` and `pickplace-wp1-cell(1,1)-robot`. Each `SimpleTask` of the production process is defined as an instance of `IndependentHRCTask` which should be associated to a `PickPlace` function of the human or a `PickPlace` function of the robot.

The resulting production process should contain a disjunction of tasks each time the needed pick & place functions can be performed by both agents. For example, a disjunctive `ComplexTask doCell(1,1)` would be associated to (alternative) `IndependentHRCTask human-doCell(1,1)` (entailing `PickPlace pickplace-wp1-cell(1,1)-human`) and `IndependentTask robot-doCell(1,1)` (`PickPlace pickplace-wp1-cell(1,1)-robot`). No disjunction should be considered if a particular task can be performed only by one agent. Suppose for example that `cell(5,10)` cannot be reached by the robot, only the human would be able to perform that task. The production process will therefore specify only a `IndependentTask doCell(5,10)`(`PickPlace` function `pickplace-wp1-cell(5,10)-human`).

Figure 16(b) shows the hierarchical structure of the designed production process. The top-level element of the hierarchy describes the production goals of the scenario. The low-level elements of the hierarchy corresponds to the description of the pick-and-place functions the human and the robot should perform. The intermediate levels describe the hierarchical decomposition of complex tasks in simple tasks. More in details Figure 15 shows the task decomposition graph extracted from the knowledge and used to build the (timeline-based) planning model through Algorithm 1.

5.2. Generation of Planning Specification from Knowledge

Given the knowledge bases of the different scenarios, this section assesses the technical feasibility of the implemented representation and reasoning processes. This section in particular evaluates the validity of the semantic models and the resulting timeline-based specifications. For each scenario it shows the time needed for the synthesis of the task planning models in order to measure the introduced production latencies. Table 1 thus summarizes and compare data about the defined knowledge bases, the characteristics of the synthesized planning models and reasoning time of Algorithm 1. As a general comment, the obtained knowledge bases have the same number of defined classes and properties since they share the same ontological framework but for each of them it can be observed a different number of *individuals* and a different number of synchronization rules, temporal constraints and state variable values in the resulting timeline-based planning models.

The structure and the size of the obtained planning models indeed differ significantly from each other (see "Planning Model" columns in Table 1) leading to different model generation time. For example, The planning model generated for AUTOMOTIVE is characterized by a lower number of predicates and synchronization rules/constraints. The planning model generated for MOSAIC instead, given its structural complexity, entails a higher number of predicates (195) as well as synchronization rules and constraints (respectively 137 and 186). In all cases the ontology was capable of capturing all the needed information and that the obtained planning models were valid and feasible for a correct deployment of the task planning module in the associated scenario. The performances of the generation

Table 1

Data about the knowledge quality, size of knowledge bases and planing models, and performance of Algorithm 1. Coherence and Consistency of the knowledge bases have been evaluated using the Protégé Debug Tool- https://protegewiki.stanford.edu/wiki/OntoDebug.

Domain	Knowledge Base					Planning Model				Model Synthesis
	Coherent	Consistent	#Classes	#Properties	#Individuals	#SVs	#Values	#Rules	#Constraints	Time (msecs)
AUTOMOTIVE	✓	✓	284	186	61	5	34	15	28	4189
METAL	✓	✓	284	186	45	8	38	57	68	7885
CAPITAL-GOODS	✓	✓	284	186	42	5	31	9	16	4165
RAILWAYS	✓	✓	284	186	75	6	64	29	48	8879
MOSAIC	✓	✓	284	186	215	7	195	137	186	20408

process are reasonable (e.g., 4 seconds for AUTOMOTIVE and 20 seconds for MOSAIC) demonstrating the feasibility of the proposed approach. The model generation time should be seen as a “constant" *setup time* of the task planner. Although it may vary significantly according to the size of the knowledge base, this cost is negligible with respect to the online functioning of a task planner since it is compatible with the usual control latency and it does not affect the overall efficiency of the system.

The results have been obtained by configuring the Apache Jena framework with an ontological model compliant with OWL-DL semantics [16]. This semantics supports many OWL features and represents a good trade-off between expressiveness and efficiency of the resulting functionalities. Examples are *disjoint classes* and *all different axioms* that allow the reasoner to correctly interpret individuals of the same class (*siblings*) as *unique* knowledge entities. Generated planning models have been validated with PLATINUm [17], a timeline-based planning and scheduling framework [99]. Figure 16 shows the hierarchical structures of the different planning models, automatically synthesized through Algorithm 1.

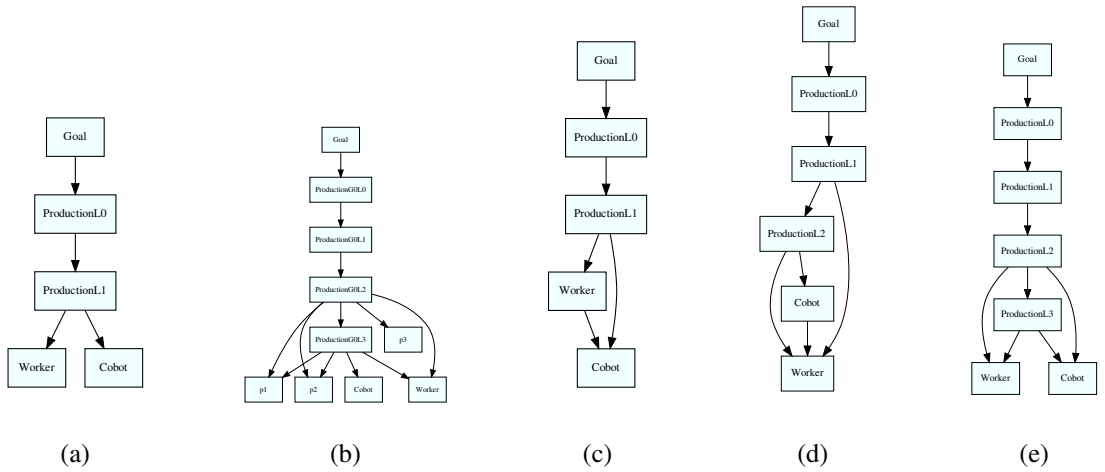


Figure 16. Hierarchical structure of the generated planning models. The two graphs show the inferred hierarchical relationships between the *state variables* generated for: (a) AUTOMOTIVE; (b) METAL; (c) CAPITAL-GOODS; (d) RAILWAYS; (e) MOSAIC

Each level represents a specific abstraction level of the defined hierarchical production procedures. The highest level of the hierarchy characterizes the high-level production goals that are incrementally decomposed in lower-level tasks (i.e., production operations). The lowest level of the hierarchy characterizes the tasks the worker and the robot can perform to support the associated production procedures. Following this hierarchy, as shown in Section 4.2.2, each hierarchy level is associated with a dedicated *state variable*. Values of such state variables describe tasks

[16] More specifically, we have defined a basic ontological model with OWL-DL-MEM specification combined with an rule-based inference model based on OWL-MEM-MICRO-RULE-INF which encapsulates optimized rule-based reasoner with OWL rules. See the official Apache Jena documentation for further details - https://jena.apache.org/documentation

[17] https://github.com/pstlab/PLATINUm.git

that should be performed at the associated abstraction level of the production procedure. State variables of the last hierarchical layer describe the concrete operations (i.e., functions) the worker and the human can perform over time and thus define their possible behaviors within a specific production scenario.

Given a timeline-based model, plans specify for each state variable of the model a sequences of *tokens* determining the production tasks performed and the low-level production operations carried out by the worker and the operator. Such sequences of tokens (i.e., *timelines*), as shown in other works [22, 23, 91], describe the planned decomposition of modeled production procedures and planned *temporal behaviors* of collaborating actors (i.e., the worker and the robot). Following [85, 90] then, each token instantiates a value of a state variable to a flexible execution interval (the intervals associated to each token represent respectively the end-time interval and the duration interval of the execution). Timelines therefore are said to encapsulate *envelopes* of temporal behaviors. One interesting aspects to point out is how the defined *ontological patterns* that formally characterize the representation structure of collaborative tasks, entail a clear and well defined structure of the defined synchronization constraints that implement that *collaborative behavior*.

5.3. Discussion of Results and Impact

The developed representation and reasoning capabilities impact different aspects concerning the design and deployment of collaborative robots. The proposed approach would facilitate modeling, maintenance and adaptation of control dynamics to the different (evolving) requirements of industrial scenarios. Table 2 lists the aspects concerning HRC production systems that are supported by the proposed approach. The table in particular shows the relevance of each aspect taking into account the production and interaction features of each pilot. In this way, it points out the flexibility of the approach according to the requirements of different scenarios.

Table 2

Aspects concerning the design and deployment of collaborative robots that are supported by the proposed ontology-based approach, and their impact on the considered HRC scenarios

	AUTOMOTIVE	METAL	CAPITAL-GOODS	RAILWAYS	MOSAIC
Knowledge Engineering	M	H	M	H	H
Personalization and Human Factor	L	H	M	L	H
Multi-objective Optimization	L	M	L	M	H
Awareness and Proactive Support	L	H	H	M	H
Modular Robot Programming	M	H	M	M	M

H = high relevance, M = medium relevance, L = low relevance

The ontology proposes a kind of standardization of production knowledge and describes collaborative scenarios based on a clear and well structured formalism. Defined concepts and properties characterize production requirements, interacting features and skills of robotic and human actors according to a clear semantics. The assessment shows that SOHO is suitable to describe both scenarios requiring simple and strict interactions between e.g., AUTOMOTIVE or `capital-goods`, and scenarios requiring more complex constraints and decomposition procedures e.g., METAL, RAILWAYS and MOSAIC.

The integrated representation of the *human factor* allows SOHO and the developed ontology-based control approach to contextualize production dynamics to the known features and skills of human workers. This knowledge combined with task planning supports the synthesis of personalized collaborative plans adapted to the expected performance (e.g., expected average time of task execution) and expertise (e.g., worker experience) [100]. This knowledge and related reasoning capabilities are especially important in scenarios characterized by a high variety of workpiece and operations e.g., METAL or MOSAIC. The automatically configured task planner would thus assign tasks and dispatch instructions that are suitable and contextualized to the known *profile* of a worker. For examples, the task planner would provide workers with low level of experience with a higher number of instruction in order to better support the execution of assigned tasks. Furthermore, the tight integration with task planning technologies supports the automatic synthesis of *optimized* collaborative plans. Timeline-based planning capabilities in particular

support multi-objective optimization by simultaneously reasoning on different metrics concerning for example the cycle time of a collaborative process, the risk of collisions and the workload distribution between the human and the robot [22]. In this regard, the automatic synthesis of (valid) task planning models supports a direct encoding of domain expert knowledge into the robot controller, reducing modeling effort and risk of inconsistencies. An optimization approach to the synthesis of collaborative plans is crucial to effectively combine human and robot skills in scenarios characterized by a high number of task allocation choices (e.g., MOSAIC) as well as multiple sets of operations and workpiece (e.g., METAL).

Another important aspect is the enhanced awareness of robot controllers. Developed knowledge representation and reasoning capabilities allow robot controllers to build and maintain an updated description of the production dynamics and observed state of the production environment. The semantics of SOHO in particular supports abstraction and contextualization of sensing data that would be useful to recognize relevant production situations and proactively trigger robot actions. Considering for example the scenario `capital-goods` these capabilities allows the developed controller to properly interpret perception outcome and contextualize human activities (i.e., bolt positioning) with respect to the known production procedures and autonomously trigger suitable task planning goals. These events are indeed automatically translated in robot tasks necessary to proactively support the human worker in the collaborative task of screwing bolts on the rotary table.

More in general the developed approach supports a modular description of robot capabilities and production requirements that can be easily extended and refined over time. For example, the knowledge base can be enriched with additional robot capabilities, human skills, production goals and procedures. This new knowledge would automatically contextualized with respect to production requirements and thus integrated into the reasoning and task planning processes. From the robot perspective, the ontology-based approach supports modular programming allowing roboticists to focus on the definition of new capabilities/skills (i.e., `Function`) without considering more general production aspects or defining complex (and static) behaviors. The synthesis of such behaviors would be indeed responsibility of the integrated task planner that would automatically evaluate new skills/capabilities and dynamically synthesize (optimized) collaborative plans.

6. Conclusions and Future Works

SOHO (*Sharework Ontology for Human Robot Collaboration*) is a novel domain ontology for Human-Robot Collaboration defined within the Sharework H2020 research project. It formally characterizes HRC manufacturing scenarios by considering different perspectives. Indeed, its main original feature relies on the use of a context-based approach to ontology design, supporting flexible representation of collaborative production processes.

This paper proposes an extension of SOHO by defining *ontology design patterns* that formally characterize collaboration dynamics that are typical in many Human-Robot Collaboration manufacturing scenarios. Furthermore we have defined a general knowledge extraction procedure that relies on the *semantics* proposed by SOHO to analyze *production knowledge* and automatically synthesize timeline-based plan-based controllers that are suitable to effectively coordinate human and robot behaviors [22, 91]. An experimental evaluation of the developed representation and reasoning technology shows the efficacy of properly capturing the complexity of real industrial collaborative scenarios and the capability of automatically “compiling" such knowledge into suitable planning domains.

Future research directions will focus on further extensions of SOHO to better characterize the *human factors* and support the representation of preferences, expertise levels, physical and cognitive conditions of human workers that are crucial to serve advanced *personalization* and finer *adaptation* features in collaborative processes. In addition, we plan to integrate developed ontology-based representation and reasoning into a *knowledge engineering tools* to facilitate domain experts in the design of collaborative process as well as in the deployment of AI-based task planning technologies for the coordination of collaborative cells [101].

Acknowledgement

Authors are partially supported by the European Commission within the Sharework project (H2020 Factories of the Future GA No. 820807). Authors are grateful to EURECAT, STIIMA-CNR, STAM, MCM, CEMBRE, Goizper

and ALSTOM, partners in Sharework and sharing their knowledge and experience about the productive scenarios that inspired our work.

References

[1] H.-P. Wiendahl, H.A. ElMaraghy, P. Nyhuis, M.F. Zäh, H.-H. Wiendahl, N. Duffie and M. Brieke, Changeable Manufacturing - Classification, Design and Operation, *CIRP Annals* **56**(2) (2007), 783–809.

[2] N. Carvalho, O. Chaim, E. Cazarini and M. Gerolamo, Manufacturing in the fourth industrial revolution: A positive prospect in Sustainable Manufacturing, *Procedia Manufacturing* **21** (2018), 671–678, 15th Global Conference on Sustainable Manufacturing.

[3] P. Fantini, M. Pinzone and M. Taisch, Placing the operator at the centre of Industry 4.0 design: Modelling and assessing human activities within cyber-physical systems, *Computers & Industrial Engineering* **139** (2020), 105058.

[4] M. Flatscher and A. Riel, Stakeholder integration for the successful product–process co-design for next-generation manufacturing technologies, *CIRP Annals* **65**(1) (2016), 181–184.

[5] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn and K. Ueda, Cyber-physical systems in manufacturing, *CIRP Annals* **65**(2) (2016), 621–641.

[6] W. Terkaj, T. Tolio and A. Valente, A Review on Manufacturing Flexibility, in: *Design of Flexible Production Systems*, Springer, 2009, pp. 41–61.

[7] L. Wang, M. Törngren and M. Onori, Current status and advancement of cyber-physical systems in manufacturing, *Journal of Manufacturing Systems* **37** (2015), 517–527.

[8] B. Dworschak and H. Zaiser, Competences for Cyber-physical Systems in Manufacturing – First Findings and Scenarios, *Procedia CIRP* **25** (2014), 345–350, 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

[9] S. Makris, *Cooperating Robots for Flexible Manufacturing*, Vol. 1, Springer International Publishing, 2021.

[10] K. Rajan and A. Saffiotti, Towards a science of integrated AI and Robotics, *Artificial Intelligence* **247** (2017), 1–9.

[11] F. Ingrand and M. Ghallab, Deliberation for autonomous robots: A survey, *Artificial Intelligence* **247** (2017), 10–44, Special Issue on AI and Robotics.

[12] S. Lemaignan, M. Warnier, E.A. Sisbot, A. Clodic and R. Alami, Artificial cognition for social human-robot interaction: An implementation, *Artificial Intelligence* **247** (2017), 45–69, Special Issue on AI and Robotics.

[13] I. Awaad, G.K. Kraetzschmar and J. Hertzberg, The Role of Functional Affordances in Socializing Robots, *International Journal of Social Robotics* **7**(4) (2015), 421–438.

[14] A. Umbrico, A. Cesta, G. Cortellessa and A. Orlandini, A Holistic Approach to Behavior Adaptation for Socially Assistive Robots, *International Journal of Social Robotics* **12**(3) (2020), 617–637.

[15] D.S. Chang, G.H. Cho and Y.S. Choi, *Ontology-Based Knowledge Model for Human-Robot Interactive Services*, in: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, Association for Computing Machinery, New York, NY, USA, 2020, pp. 2029–2038–. ISBN 9781450368667.

[16] S. Borgo, A. Cesta, A. Orlandini and A. Umbrico, Knowledge-based Adaptive Agents for Manufacturing Domains, *Engineering with Computers* **35**(3) (2019), 755–779.

[17] M. Diab, A. Akbari, M. Ud Din and J. Rosell, PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation, *Sensors* **19**(5) (2019).

[18] D. Beßler, M. Pomarlan and M. Beetz, OWL-Enabled Assembly Planning for Robotic Agents, in: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2018, pp. 1684–1692–.

[19] A.R. Sadik and B. Urban, An Ontology-Based Approach to Enable Knowledge Representation and Reasoning in Worker–Cobot Agile Manufacturing, *Future Internet* **9**(4) (2017).

[20] S. Borgo, A. Cesta, A. Orlandini and A. Umbrico, A Planning-based Architecture for a Reconfigurable Manufacturing System, in: *The 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 2016, pp. 358–366.

[21] A. Umbrico, A. Orlandini and A. Cesta, An Ontology for Human-Robot Collaboration, *Procedia CIRP* **93** (2020), 1097–1102.

[22] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico and T. Tollio, Motion planning nad scheduling for human and industrial-robot collaboration, *CIRP Annals - Manufacturing Technology. To appear.* (2017).

[23] A. Cesta, A. Orlandini and A. Umbrico, Fostering Robust Human-Robot Collaboration through AI Task Planning, *Procedia CIRP* **72** (2018), 1045–1050, 51st CIRP Conference on Manufacturing Systems.

[24] A. Gangemi, Ontology Design Patterns for Semantic Web Content, in: *The Semantic Web – ISWC 2005*, Y. Gil, E. Motta, V.R. Benjamins and M.A. Musen, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 262–276. ISBN 978-3-540-32082-1.

[25] E. Matheson, R. Minto, E.G.G. Zampieri, M. Faccio and G. Rosati, Human–Robot Collaboration in Manufacturing Applications: A Review, *Robotics* **8**(4) (2019).

[26] E. Helms, R.D. Schraft and M. Hagele, rob@work: Robot assistant in industrial environments, in: *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002, pp. 399–404.

[27] M. Ghallab, D. Nau and P. Traverso, The actor's view of automated planning and acting: A position paper, *Artificial Intelligence* **208** (2014), 1–17.

[28] K. Dautenhahn, M. Walters, S. Woods, K.L. Koay, C.L. Nehaniv, A. Sisbot, R. Alami and T. Siméon, How May I Serve You? A Robot Companion Approaching a Seated Person in a Helping Context, in: *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, HRI '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 172–179–. ISBN 1595932941.

[29] S. Rossi, F. Ferland and A. Tapus, User profiling and behavioral adaptation for HRI: A survey, *Pattern Recognition Letters* **99** (2017), 3–12, User Profiling and Behavior Adaptation for Human-Robot Interaction.

[30] B. Bruno, C.T. Recchiuto, I. Papadopoulos, A. Saffiotti, C. Koulouglioti, R. Menicatti, F. Mastrogiovanni, R. Zaccaria and A. Sgorbissa, Knowledge Representation for Culturally Competent Personal Robots: Requirements, Design Principles, Implementation, and Assessment, *International Journal of Social Robotics* **11**(3) (2019), 515–538. ISBN 1875-4805.

[31] S. Manzoor, Y.G. Rocha, S.-H. Joo, S.-H. Bae, E.-J. Kim, K.-J. Joo and T.-Y. Kuc, Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications, *Applied Sciences* **11**(10) (2021).

[32] P. Turaga, R. Chellappa, V.S. Subrahmanian and O. Udrea, Machine Recognition of Human Activities: A Survey, *IEEE Transactions on Circuits and Systems for Video Technology* **18**(11) (2008), 1473–1488.

[33] S.K. Chandrasegaran, K. Ramani, R.D. Sriram, I. Horváth, A. Bernard, R.F. Harik and W. Gao, The evolution, challenges, and future of knowledge representation in product design systems, *Computer-Aided Design* **45**(2) (2013), 204–228, Solid and Physical Modeling 2012.

[34] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A.K. Bozcuoğlu and G. Bartels, Know Rob 2.0 — A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 512–519.

[35] M. Tenorth and M. Beetz, Representations for robot knowledge in the KnowRob framework, *Artificial Intelligence* **247** (2017), 151–169, Special Issue on AI and Robotics.

[36] D. Beßler, R. Porzel, M. Pomarlan, M. Beetz, R. Malaka and J. Bateman, A formal model of affordances for flexible robotic task execution, in: *ECAI 2020*, IOS Press, 2020, pp. 2425–2432.

[37] R.E. Fikes and N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial intelligence* **2**(3–4) (1971), 189–208.

[38] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld and D. Wilkins, PDDL - The Planning Domain Definition Language, Technical Report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[39] M. Diab, M. Pomarlan, D. Beßler, A. Akbari, J. Rosell, J. Bateman and M. Beetz, SkillMaN — A skill-based robotic manipulation framework based on perception and reasoning, *Robotics and Autonomous Systems* **134** (2020), 103653.

[40] F. Rovida, M. Crosby, D. Holz, A.S. Polydoros, B. Großmann, R.P.A. Petrick and V. Krüger, *SkiROS—A Skill-Based Robot Control Platform on Top of ROS*, in: *Robot Operating System (ROS): The Complete Reference (Volume 2)*, A. Koubaa, ed., Springer International Publishing, Cham, 2017, pp. 121–160. ISBN 978-3-319-54927-9.

[41] S. Lemaignan, R. Ros, L. Mosenlechner, R. Alami and M. Beetz, ORO, a knowledge management platform for cognitive architectures in robotics, in: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 3548–3553. ISSN 2153-0858.

[42] P.J.S. Gonçalves and P.M.B. Torres, Knowledge representation applied to robotic orthopedic surgery, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 90–99, Special Issue on Knowledge Driven Robotics and Manufacturing.

[43] I.H. Suh, G.H. Lim, W. Hwang, H. Suh, J.-H. Choi and Y.-T. Park, Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence, in: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 429–436.

[44] P. Ribino, M. Bonomolo, C. Lodato and G. Vitale, A Humanoid Social Robot Based Approach for Indoor Environment Quality Monitoring and Well-Being Improvement, *International Journal of Social Robotics* **13**(2) (2021), 277–296.

[45] L. Sabri, S. Bouznad, S. Rama Fiorini, A. Chibani, E. Prestes and Y. Amirat, An integrated semantic framework for designing context-aware Internet of Robotic Things systems, *Integrated Computer-Aided Engineering* **25**(2) (2018), 137–156. ISBN 1875-8835.

[46] E.M. Sanfilippo, Y. Kitamura and R.I.M. Young, Formal ontologies in manufacturing, *Applied Ontology* **14** (2019), 119–125.

[47] W. Terkaj and M. Urgo, Ontology-based modeling of production systems for design and performance evaluation, in: *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 748–753.

[48] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case and J.A. Harding, Towards a formal manufacturing reference ontology, *International Journal of Production Research* **51**(22) (2013), 6553–6572.

[49] S. Borgo, An Ontological View of Components and Interactions in Behaviorally Adaptive Systems, *Journal of Integrated Design and Process Science* **23**(1) (2019), 17–35. ISBN 1875-8959.

[50] S. Balakirsky, Ontology based action planning and verification for agile manufacturing, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 21–28, Special Issue on Knowledge Driven Robotics and Manufacturing.

[51] L. Solano, P. Rosado and F. Romero, Knowledge representation for product and processes development planning in collaborative environments, *International Journal of Computer Integrated Manufacturing* **27**(8) (2014), 787–801.

[52] A. Hristoskova, C.E. Agüero, M. Veloso and F.D. Turck, Heterogeneous Context-Aware Robots Providing a Personalized Building Tour, *International Journal of Advanced Robotic Systems* **10**(1) (2013), 14.

[53] D. Calisi, L. Iocchi, D. Nardi, C.M. Scalzo and V.A. Ziparo, Context-based design of robotic systems, *Robotics and Autonomous Systems* **56**(11) (2008), 992–1003, Semantic Knowledge in Robotics.

[54] Z. Kootbally, T.R. Kramer, C. Schlenoff and S.K. Gupta, Implementation of an Ontology-Based Approach to Enable Agility in Kit Building Applications, *International Journal of Semantic Computing* **12**(01) (2018), 5–24.

[55] A. Olivares-Alarcos, S. Foix, S. Borgo and Guillem Alenyà,, OCRA – An ontology for collaborative robotics and adaptation, *Computers in Industry* **138** (2022), 103627.

[56] R. Patel, M. Hedelind and P. Lozan-Villegas, Enabling robots in small-part assembly lines: The "ROSETTA approach" - an industrial perspective, in: *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1–5.

[57] M. Stenmark and J. Malec, Knowledge-based instruction of manipulation tasks for industrial robotics, *Robotics and Computer-Integrated Manufacturing* **33** (2015), 56–67, Special Issue on Knowledge Driven Robotics and Manufacturing.
[58] N. Guarino, *Formal Ontology in Information Systems*, IOS Press, 1998.
[59] S. Borgo, M. Carrara, P. Garbacz and P.E. Vermaas, A formal ontological perspective on the behaviors and functions of technical artifacts, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **23**(1) (2009), 3–21.
[60] G. Antoniou and F. van Harmelen, *Web Ontology Language: OWL*, in: *Handbook on Ontologies*, S. Staab and R. Studer, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 67–92. ISBN 978-3-540-24750-0.
[61] L. Jansen and S. Schulz, The ten commandments of ontological engineering, in: *Proceedings of the 3rd Workshop of Ontologies in Biomedicine and Life Sciences*, 2011, pp. 1–6.
[62] V. Mascardi, V. Cordì and P. Rosso, A Comparison of Upper Ontologies, in: *Workshop from Objects to Agents (WOA)*, 2007, pp. 1–10.
[63] J.N. Otte, J. Beverley and A. Ruttenberg, BFO: Basic Formal Ontology, *Applied Ontology* **17** (2022), 17–43.
[64] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E.M. Sanfilippo and L. Vieu, DOLCE: A descriptive ontology for linguistic and cognitive engineering, *Applied Ontology* **17** (2022), 45–69.
[65] I. Niles and A. Pease, Towards a Standard Upper Ontology, in: *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, Association for Computing Machinery, 2001, pp. 2–9–. ISBN 1581133774.
[66] E. Prestes, J.L. Carbonera, S.R. Fiorini, V.A.M. Jorge, M. Abel, R. Madhavan, A. Locoro, P. Goncalves, M.E. Barreto, M. Habib, A. Chibani, S. Gérard, Y. Amirat and C. Schlenoff, Towards a core ontology for robotics and automation, *Robotics and Autonomous Systems* **61**(11) (2013), 1193–1204.
[67] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W.D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth and K. Taylor, The SSN Ontology of the W3C Semantic Sensor Network Incubator Group **17**(C) (2012), 25–32–.
[68] H.-M. Huang, K. Pavek, B. Novak, J.S. Albus and E.R. Messina, A Framework For Autonomy Levels For Unmanned Systems (ALFUS), in: *Association of the Unmanned Vehicle System International (AUVSI)*, 2005.
[69] C. Bicchieri, *The Grammar of Society: The Nature and Dynamics of Social Norms*, Cambridge University Press, 2005.
[70] C. Castelfranchi, Commitments: From Individual Intentions to Groups and Organizations, in: *Proceedings of the First International Conference on Multiagent Systems*, 1995, pp. 41–48.
[71] R. Alami, A. Clodic, V. Montreuil, E.A. Sisbot and R. Chatila, Toward Human-Aware Robot Task Planning., in: *AAAI spring symposium: to boldly go where no human-robot team has gone before*, 2006, pp. 39–46.
[72] E.A. Sisbot, A. Clodic, L.F. Marin-Urias, M. Fontmarty, L. Brethes and R. Alami, Implementing a Human-Aware Robot System, in: *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006, pp. 727–732.
[73] A. Clodic, R. Alami and R. Chatila, Key Elements for Human-Robot Joint Action, in: *Sociable Robots and the Future of Social Relations*, 2014.
[74] J.A. Marvel, J. Falco and I. Marstio, Characterizing Task-Based Human-Robot Collaboration Safety in Manufacturing, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **45**(2) (2015), 260–275.
[75] J.T.C. Tan, Feng Duan, Ye Zhang and Tamio Arai, Task decomposition of cell production assembly operation for man-machine collaboration by HTA, in: *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 1066–1071.
[76] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, The Semantic Web: the roles of XML and RDF, *IEEE Internet Computing* **4**(5) (2000), 63–73.
[77] J. Pérez, M. Arenas and C. Gutierrez, Semantics and Complexity of SPARQL, *ACM Trans. Database Syst.* **34**(3) (2009).
[78] M. Fox and D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, *J. Artif. Intell. Res. (JAIR)* **20** (2003), 61–124.
[79] J. Hoffmann, FF: The Fast-Forward Planning System, *AI Magazine* **22**(3) (2001), 57–62.
[80] A.J. Coles, A. Coles, M. Fox and D. Long, Forward-Chaining Partial-Order Planning, in: *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 2010, pp. 42–49.
[81] F. Dvorák, R. Barták, A. Bit-Monnot, F. Ingrand and M. Ghallab, Planning and Acting with Temporal and Hierarchical Decomposition Models, in: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2014, pp. 115–121. ISSN 1082-3409.
[82] D. Nau, T.C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu and F. Yaman, SHOP2: An HTN Planning System, *Journal on Artificial Intelligence Research* **20** (2003).
[83] P.H. Morris, N. Muscettola and T. Vidal, Dynamic Control of Plans With Temporal Uncertainty, in: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, pp. 494–502.
[84] T. Vidal and H. Fargier, Handling Contingency in Temporal Constraint Networks: From Consistency To Controllabilities, *JETAI* **11**(1) (1999).
[85] M. Cialdea Mayer, A. Orlandini and A. Umbrico, Planning and execution with flexible timelines: a formal account, *Acta Informatica* **53**(6–8) (2016), 649–680.
[86] N. Muscettola, S.F. Smith, A. Cesta and D. D'Aloisi, Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture, *IEEE Control Systems* **12**(1) (1992), 28–37.
[87] A. Cesta, A. Oddi and S.F. Smith, A Constraint-based method for Project Scheduling with Time Windows, *Journal of Heuristics* **8**(1) (2002), 109–136.
[88] A.K. Jonsson, P.H. Morris, N. Muscettola, K. Rajan and B. Smith, Planning in Interplanetary Space: Theory and Practice, in: *AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling*, 2000.

[89] F. Py, K. Rajan and C. McGann, A Systematic Agent Framework for Situated Autonomous Systems, in: *AAMAS-10. Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2010.

[90] M. Cialdea Mayer and A. Orlandini, An Executable Semantics of Flexible Plans in Terms of Timed Game Automata, in: *The 22nd International Symposium on Temporal Representation and Reasoning (TIME)*, IEEE, 2015.

[91] M. Faroni, M. Beschi, S. Ghidini, N. Pedrocchi, A. Umbrico, A. Orlandini and A. Cesta, A Layered Control Approach to Human-Aware Task and Motion Planning for Human-Robot Collaboration, in: *IEEE Int. Conf. on Robot and Human Inter. Comm.*, Naples (Italy), 2020.

[92] T. Vidal, A Unified Dynamic Approach for Dealing with Temporal Uncertainty and Conditional Planning, in: *AIPS-00. Proc. of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling*, 2000, pp. 395–402.

[93] A. Cesta, A. Finzi, S. Fratini, A. Orlandini and E. Tronci, Analyzing Flexible Timeline Plan, in: *ECAI 2010. Proceedings of the 19th European Conference on Artificial Intelligence*, Vol. 215, IOS Press, 2010.

[94] J.F. Allen, Maintaining knowledge about temporal intervals, *Commun. ACM* **26**(11) (1983), 832–843.

[95] A. Cesta and A. Oddi, DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains,, in: *New Directions in AI Planning*, M. Ghallab and A. Milani, eds, IOS Press: Amsterdam, 1996.

[96] A. Umbrico, A. Orlandini, A. Cesta, S. Koukas, A. Zalonis, N. Fourtakas, D. Andronas, G. Apostolopoulos and S. Makris, Towards User-Awareness in Human-Robot Collaboration for Future Cyber-Physical Systems, in: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.

[97] C.L. Viola, A. Orlandini, A. Umbrico and A. Cesta, ROS-TiPlEx: How to make experts in A.I. Planning and Robotics talk together and be happy, in: *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1–6.

[98] N. Mandischer, T. Huhn, M. Hüsing and B. Corves, Efficient and Consumer-Centered Item Detection and Classification with a Multicamera Network at High Ranges, *Sensors* **21**(14) (2021).

[99] A. Umbrico, A. Cesta, M. Cialdea Mayer and A. Orlandini, PLATINUm: A New Framework for Planning and Acting, in: *AI*IA 2017 Advances in Artificial Intelligence*, 2017, pp. 498–512. ISBN 978-3-319-70169-1.

[100] A. Umbrico, A. Orlandini, A. Cesta, M. Faroni, M. Beschi, N. Pedrocchi, A. Scala, P. Tavormina, S. Koukas, A. Zalonis, N. Fourtakas, P.S. Kotsaris, D. Andronas and S. Makris, Design of Advanced Human-Robot Collaborative Cells for Personalized Human-Robot Collaborations, *Applied Sciences* **12**(14) (2022).

[101] E. Foderaro, A. Cesta, A. Umbrico and A. Orlandini, Simplifying the A.I. Planning modeling for Human-Robot Collaboration, in: *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, 2021, pp. 1011–1016.