Open University of Cyprus

Faculty Of Pure and Applied Sciences

Postgraduate (Master's) *Cognitive Systems*

Postgraduate (Master's) Dissertation

Argumentation Based Coaching of an Industrial Robotic Arm

Mohammad Anas Salman

Supervisor
Dr. Isidoros Perikos

May 2022

Open University of Cyprus

Faculty Of Pure and Applied Sciences

Postgraduate (Master's) *Cognitive Systems*

Postgraduate (Master's) Dissertation

Argumentation Based Coaching of an Industrial Robotic Arm

Mohammad Anas Salman

Supervisor
Dr. Isidoros Perikos

The present Postgraduate (Master's) Dissertation was submitted in partial fulfilment of the requirements for the postgraduate degree
in Cognitive Systems
Faculty Of Pure and Applied Sciences
of the Open University of Cyprus.

May 2022

BLANK PAGE

Summary

Since the advent of the first industrial revolution, the need for machines that would help to increase production in order to fulfill market demands has increased exponentially. Industrial robots have sparked a lot of attention since then. In order to cope with industrial needs, engineers and machine designers have endeavored to construct machines that would work on the kinematics inspired by the human arm. With the developments in technology, industrial robotic arms have changed over time. Although the initial models were more hydraulic and hardwire driven, the recent robotic arms incorporate highly sophisticated mechanics, electronics, and software. With the dawn of the Fourth Industrial Revolution, industries have increased their technology benchmark and are in need of smart technology that can learn, infer, and explain their behavior. This has expanded the research in the Human Machine Interaction domain where scientists have managed to propose such systems where interacting with industrial machines has become easier. Building automation systems through no code or low code approaches has further alleviated the technology benchmarks. In this Master's dissertation, we propose an approach under the shadow of the Human Machine Interaction domain to coach an industrial robotic arm through the PRUDENS interface that facilitates machine coaching through argumentation and machine-learning theories, which appear to be useful in monitoring the machine's behavior and guiding it to adapt itself under exceptional settings. PRUDENS is a software tool that has been developed by the Computational Cognition Lab of the Open University of Cyprus led by Dr. Loizos Michael. We implement a real-time human-robot interaction system that facilitates machine coaching within industrial boundaries, in addition to discussing recent trends in the human-robot interaction domain and the implications of AI, ML, and argumentation techniques on it.

Περίληψη

Από την έλευση της πρώτης βιομηχανικής επανάστασης, η ανάγκη για μηχανές που θα βοηθούσαν στην αύξηση της παραγωγής προκειμένου να ικανοποιηθούν οι απαιτήσεις της αγοράς έχει αυξηθεί ραγδαία. Τα βιομηχανικά ρομπότ έχουν προκαλέσει μεγάλη προσοχή από τότε. Προκειμένου να ανταποκριθούν στις βιομηχανικές ανάγκες, οι μηχανικοί και οι σχεδιαστές μηχανών προσπάθησαν να κατασκευάσουν μηχανές που θα λειτουργούσαν στην κινηματική, εμπνευσμένες από τον ανθρώπινο βραχίονα. Με τις εξελίξεις στην τεχνολογία, οι βιομηχανικοί ρομποτικοί βραχίονες έχουν αλλάξει με την πάροδο του χρόνου. Αν και τα αρχικά μοντέλα ήταν πιο υδραυλικά και με σκληρό καλώδιο, οι πρόσφατοι ρομποτικοί βραχίονες ενσωματώνουν εξαιρετικά εξελιγμένους μηχανισμούς, ηλεκτρονική και λογισμικό. Με τον ερχομό της Τέταρτης Βιομηχανικής Επανάστασης, οι βιομηχανίες έχουν αυξήσει το τεχνολογικό σημείο αναφοράς τους και χρειάζονται έξυπνη τεχνολογία που θα μπορούσε να μάθει, να συμπεράνει και να εξηγήσει τη συμπεριφορά του. Αυτό επέκτεινε την έρευνα στον τομέα της Αλληλεπίδρασης Ανθρώπων - Μηχανών που οι επιστήμονες κατάφεραν να προτείνουν τέτοια συστήματα στα οποία η αλληλεπίδραση με τις βιομηχανικές μηχανές έχει γίνει ευκολότερη. Η κατασκευή συστημάτων αυτοματισμού χωρίς κώδικα ή η προσέγγιση με ελάχιστο κώδικα έχει μετριάσει περαιτέρω τα τεχνολογικά σημεία αναφοράς. Σε αυτή τη μεταπτυχιακή διατριβή, προτείνουμε μια προσέγγιση υπό το πρίσμα του τομέα Αλληλεπίδρασης Ανθρώπινων Μηχανών για την καθοδήγηση ενός βιομηχανικού ρομποτικού βραχίονα μέσω του περιβάλλοντος PRUDENS που διευκολύνει την καθοδήγηση μηχανών μέσω επιχειρημάτων και θεωριών μηχανικής μάθησης, που φαίνεται να είναι χρήσιμες για την παρακολούθηση της συμπεριφοράς και την καθοδήγηση της μηχανής να προσαρμόζεται σε εξαιρετικές ρυθμίσεις. PRUDENS είναι ένα εργαλείο λογισμικού που έχει αναπτυχθεί από το Εργαστήριο Υπολογιστικής Νόησης του Ανοικτού Πανεπιστημίου Κύπρου με επικεφαλής τον Δρ Λοΐζο Μιχαήλ. Εκτός από τη συζήτηση των πρόσφατων τάσεων στον τομέα αλληλεπίδρασης ανθρώπου-ρομπότ και των επιπτώσεων της τεχνητής νοημοσύνης, μηχανικής μάθησης και τεχνικών επιχειρηματολογίας σε αυτό, εφαρμόζουμε ένα σύστημα αλληλεπίδρασης ανθρώπου-ρομπότ σε πραγματικό χρόνο που διευκολύνει την καθοδήγηση μηχανών εντός των βιομηχανικών ορίων.

Acknowledgments

I am highly grateful to The Almighty who gave me the strength to pursue this masters education. I would like to thank my supervisor Dr. Isidoros Perikos for accepting me to complete my Dissertation under his valuable guidance and for his consistent support during this entire phase. I would like to convey my special gratitude to Mr. Vasileios Markos for his valuable inputs regarding the PRUDENS tool. Furthermore, I would like to thank my family for being a constant support system without whom I wouldn't have come so far. Lastly, I am highly grateful to my daughter Erina Salman for her constant motivation.

Table of Contents:

Chapter 1 Introduction

With the advent of the 4th industrial revolution, human-machine interaction is increasing, and smart technologies are rapidly penetrating every aspect of our lives. Today, smartphones have become part of our lives, and it is natural for people to interact with each other anytime and anywhere using these internet enabled smartphones. With Industry 4.0, this situation is developing further, and the concept of the internet of things is being clearly adopted alongside the concept of the internet of people. As a result of all of this, technology will reach a point where machines and systems can talk to each other. Humans will play a role that is more like that of a teacher or coach, and this will allow machines and systems to store some knowledge so that they can learn and reason more like humans in an industrial setting.

Under this industrial revolution, it is expected that machines will organize themselves around intelligent connections and distributions and try to minimize the human error rate. With their integrated communication capabilities, machines could make smart decisions and stay alert at all stages of production by being in constant connection and communication with their environment. These self-organization capabilities of machines would pave the way for complex production scenarios that could be run in the future where humans tend to make errors. Special products that need to be produced one by one with the interaction of machines could be produced flexibly and in mass without the need for human intervention and could be quickly released to the market. Such tasks under the present scenario are handled by robots, which are nothing but industrial machines that work in the way they are programmed. These robots could develop reasoning capabilities if they have previous knowledge related to the general world or knowledge that is task-specific, which could be learnt dynamically as they operate in the environment. In a broader sense, this would also enable these robots or machines to explain their actions and as a result they could also be coached by humans through mutual argumentation. This argumentation-based reasoning capability would further enable humans to seek guidance from these machines under some specific scenarios.

Argumentation is a process of reasoning under which conclusions are formed based on a theory. While explanation aims to provide reasons or arguments that support a conclusion against other competing conclusions, argumentation strives to provide reasons or arguments that support a conclusion against other conflicting conclusions (Craven & Toni 2016, Gaertner & Toni 2007). When given a rule "A follows from B" and the observed outcome "A," snatching is frequently referred to as in reverse. We can deduce that the rule's condition "B" is (possibly) true. A set of sentences representing an addictive explanation H for an observation O is returned in a logic-based setting, given a set of sentences representing a theory T that models a domain of interest and a sentence representing an observation O, such that: 1. T H |= O, 2. T H is consistent, where |= denotes the deductive logical entailment relation. For the same observation, there may be several explanations in many circumstances. Extra standards such as minimality may be applied for an explanation to be considered admissible. It could also be said that the argumentation process is used for defending a claim (such as a belief or a choice) with the use of some premises and an argument that connects these premises to the claim. Arguments in support of a claim are supposed to be accepted or legitimate in the sense that they can defend themselves against all other competing arguments, i.e., counter-arguments that challenge the supporting argument. Argumentation takes place in a formal context inside a predetermined framework. e.g, argumentation framework, *AF = (h, A, D)*, where *h* is a set of arguments, *A* is an attribute, and *D* is a function. Furthermore, *D* is a binary defense (or defeat) relation on *h*, whereas *A* is a binary attack relation on *h*. The role of attack relation comes into play when one argument is a counter-argument (i.e., opposing or challenging) to another whereas the role of defense relation comes into play when one argument is powerful enough to defend against another (opposing or challenging) argument. Under the aforementioned argumentation framework, a subset of arguments is acceptable under the following conditions:

- The argument is conflict-free, that is, it does not comprise arguments that are antagonistic to one another.

- The argument defends itself against all additional subsets of arguments that are directed to it.

Therefore, the goal of argumentation is to form a coalition of arguments with other arguments to defend it against arguments that would undermine it in some way, such as by questioning its premises or the appropriateness of its link between its premises.

1.1 What is Human-Machine Interaction?

The way in which a human being interacts with a machine, which could be anything that reduces human effort, is simply termed "Human Machine Interaction (HMI)." There are various aliases present for the term Human Machine Interaction, such as Human Computer Interaction (HCI), Human Machine Interface (HMI), Man Machine Interface (MMI) etc. In today's world, Human-Machine Interaction is directly related to the high-level machines that involve ample software usage and have sophisticated electronics. The need for HMI has evolved greatly over time as we can find such devices right in our homes, offices, shopping centers, etc. and everywhere around us in the ecosystem. Regarding some simple examples of HMI, we have the keyboard, which is a lifesaving invention derived from the typewriter. It plays a key role in the advancement of technology today. Then we have the mouse, the GUI panels present on the electrical home appliances, the KIOSKS present around us in the city, medical devices, computer gadgets, GUI panels in industrial machinery etc. Through HMIs, we can control the machines as per our desire, and with the advancement of AI and machine learning technologies, we could train the machines over a set of inputs. Ergonomics is important in the design and development of HMIs because it keeps the interaction between the human and the machine as simple as possible.

Groundbreaking advancements in electronics technology have made the integration of devices such as cameras, microphones, sensors, etc., possible in order to be used as the interaction interface between humans and machines.

1.2 What is Industrial robotic Arm and Argumentation based Machine Coaching?

An industrial robotic arm is a reprogrammable, autonomous, or semi-independent machine that functions to assist a human worker with numerous tasks in a production environment or is capable of working independently. It can perform mechanically challenging and repetitive tasks as well as manipulate objects through pre-programmed movements. Besides being able to handle risky tasks such as entering into a secondary machine which entails fatality risks for humans, it is also capable of collecting and processing data to optimize the work and adapt its behavior to the surrounding environment through some sophisticated artificial intelligence algorithms.

Unimate was the first industrial robot, which was developed in 1954 by George Charles Devol, who is also known as the father of robotics. A few years later, Devol and entrepreneur Joseph F. Engelberger founded Unimation, the world's first robotics firm. After applying for the patent of Unimate in 1954, Devol personally introduced it, which was sent to General Motors in 1961, who first used this robot for die casting and spot welding of car bodies. Subsequent to this, General Motors bought 66 more Unimates and installed them in its factory. Affected by these developments, Ford and Chrysler companies also became interested in industrial robots. With the increasing interest and investment of the entire automotive industry, the future of industrial robots has begun to shine (Wallén 2008). According to the International Federation of Robotics (IFR) 2020 report, the number of robots used in factories worldwide increased by 12%, to 2.7 million. Today, the industrial robot market is worth $41.23 billion. As per their latest report for the 2021 edition, IFR states that in manufacturing industries, the global average robot density has almost doubled in the last five years, having 126 robots per 10,000 employees, with Asian countries such as South Korea, Singapore, and Japan topping the charts.

With the evolution of these industrial robots over time, the requirement for more sophisticated control algorithms and the development of some human-machine interaction interfaces were born. Although recent industrial robotic arms are outfitted with cutting-edge human-machine interaction interfaces that allow the user to program them to work according to process requirements and create some motion trajectory in space via point teaching, they lack the ability to explain their behavior to the user or learn from some user-generated heuristics in a dynamic setting.

This kind of reasoning and learning could be made possible if these industrial robots could be interfaced with some kind of reasoning framework that is capable of communicating with the user as well as with these robots in a language that is mutually interpretable. Through this, humans could seek guidance on specific scenarios or convey their message to these machines. This would further enhance the role of machine operators in the industry from performing learning supervision tasks to endowing knowledge to the machines as coaches.

Machine coaching is a two-way approach to knowledge acquisition that falls under the human-machine interaction domain. Through Machine Coaching, a human user may transmit his own preferences, expertise, and/or intuition to a machine by giving it guidance in the form of reasons in favor of or against specific behaviors. A machine coaching cycle takes place between two parties, such as a smart system that has a basic knowledge of its own and a supervisor, which could be a human or a highly knowledgeable smart system. Based on the knowledge and the preferences, the smart system is able to make decisions pertaining to a specific goal. The supervisor, on the other hand, could make this decision-making process more accurate by enhancing the knowledge of the smart system about that specific goal. A machine coaching cycle is built up of the following steps:

- Triggering of the smart system or the cognitive agent by the supervisor or advice seeker to seek guidance on a specific task which acts as the context for the agent.

- With the available knowledge base of the cognitive agent, some advice about the context is proposed.

- The advice seeker could then prompt the cognitive agent for the associated reasons pertaining to the proposed advice.

- The cognitive agent provides the explanation backed by the associated reasons from its knowledge base pertaining to its proposal.

- The advice seeker, if it disagreed with the agent’s explanation, could provide its own outlook on the context, which would initiate the knowledge acquisition process through user-specific reasons or simply the associated arguments. However, in the case of an agreement, the advice seeker would accept the agent’s advice and the entailing explanation. Therefore, the agent’s knowledge is not interrupted, and the advice seeker may proceed with another query, reinitializing the entire process.

In a high-level setting, this could be done by, for example, removing, adding, or changing a specific rule from the agent's knowledge base as part of its explanation, or by changing the rule's priority, which is the order in which it appears in the agent's knowledge base.

The goal of coaching an industrial robotic arm through argumentation is to improve the rudimentary forms of human-machine communication and mutual understanding by offering a fundamentally more suitable language of communication that makes the machine's internal reasoning simple to the human. As part of our proposed system, we can use the hard-coded logic-based previous knowledge and inference application through logical arguments to achieve argumentation-based reasoning within the cognitive agent. This makes machine coaching possible when the cognitive agent is connected to the robotic arm.

1.3 The Link between Industrial robotic arm and Argumentation.

Argumentation can be seen as a process of explaining a claim by the set of premises on which the supporting argument is built. From the perspective of argumentation-based coaching of an industrial robotic arm, we believe that there is an existing interface that could convey the arm’s current situation dynamically in real time to the argumentation framework, which consists of some predefined knowledge in the form of rules. These rules could contain the information that is necessary to coach the arm. The user could also seek guidance about the arm’s specific behavior under the current situation or rather the context. This explanation is conveyed in the form of inferences to the user, which are backed by related arguments extracted from the rules in the knowledge base of the argumentation framework. The user could accept the supporting arguments for these inferences or reject them by adding, deleting, or modifying the rules in the available knowledge base in order to achieve the desired explanation. As the robotic arm is communicating with the argumentation framework in real time, it also receives the generated inferences, which enables the arm to change its behavior depending upon the user's arguments. Under the argumentation framework, the existence of an argument for some inference could be understood mathematically as described in (Michael 2019). Let h be some literal, k be a prioritized knowledge base, k = (R, p) and c be some context, which is a set of pairwise non-conflicting literals. Assuming that (I = R U c), we conclude that A is an argument for h in c under k if A is a subset of I, *(*A ⊆ I), i.e., A contains few or all elements in I under the following conditions:

1. A ≠ ø
2. h could be inferred through repeated application of modus ponens starting from literals in A ∩ c using the rules in A ∩ R
3. We cannot infer h from M, which is a proper subset of A, (M ⊂ A), i.e., M contains fewer elements than A, such that ø ≠ M ⊂ A by substituting it in place of A as in (2).

Chapter 2
Explanation

2.1 Objectives & Necessity of Research Study

Embedding cognitive abilities and human intelligence in machines has been a hot topic among researchers and scientists worldwide. Although such features are quite common in humanoid robotics, computers, and mobile applications, etc., the industrial world still starves for them. The main issue with industrial automation systems is their limitation to interacting with humans through a language that both can comprehend in real time. This makes it difficult for the machine operators or learning supervisors to change some of its inappropriate behavior instantly by annotating knowledge that would in turn enhance the machine's performance. The main objective of this research study is to create an interaction interface that would facilitate machine coaching through an argumentation process. It would serve as a bridge between the human and the machine through which the former could interpret the latter's explanation pertaining to some specific behavior and guide it where necessary to achieve maximum performance. In our case, the machine is an industrial robotic arm, which will be coached by a human operator to improve some aspects of its behavior in an industrial environment.

Industrial Robotic arms are widely used across all sorts of industries worldwide, be it automotive, aerospace, metal working, material handling, packaging, medical, pharma, etc. Each sector has its own application requirements, complexities, and challenges. The necessity of this research lies in the fact that manual teaching or application-based teaching of these robotic arms is time-consuming and requires skills and experience that machine or line operators do not possess. This increases machine downtime and incurs production loss for the industry. With the advancement in smart technologies, it is possible to incorporate techniques which could offer some cognitive characteristics to these smart machines. This would make them intelligent enough to converse with the users for easy learning based on their directives and also argue with the users to come up with a correct learning result without the requirement of traditional machine programming. This would enable the operators to interact freely with these robotic arms without the requirement of any special technical skills.

We therefore propose a system that consists of an argumentation assistant and an industrial robot which are connected through an interaction interface that is accessed by the human user. The assistant communicates with the robot over TCP-IP to exchange information that is crucial for its operation. The robot, on the other hand, evaluates the operation scenario through constant monitoring of its system information. This information consists of the robot's model, firmware version, hours of operation, current coordinates in the joint reference frame, current tool center point coordinates in the world reference frame, the mechanical and software limits of the six axes, its operation mode (manual or automatic), the emergency circuit status, etc. As per the reasoning outcome, the assistant could also give commands to the robot, such as starting or stopping the operation cycles, sending the robot to its home position, and so on. The assistant also has access to the robot's application-specific signals such as sensors, actuators, etc.

To achieve the reasoning functionality through a set of rules under a specific situation, the assistant consists of a knowledge base and a context. The assistant, based on the current context that is cyclically updated by the robot, deduces its outcome from the set of rules that have been manually defined in its knowledge base by a human user. These rules describe the robot's standard policy of operation. Depending upon the user's requirements related to the robot's operation, these rules could be modified, deleted, or new rules could be added in the assistant's knowledge base. Argumentation functionality is achieved in such a way that the assistant presents an argument in the form of a rule or a set of rules from its knowledge base that support its explanation in the specified context. These explanations are sent to the robot to actuate the control scenario. The user who acts as a supervisor, on the other hand, could present its counter argument in case of disagreement with the agent's explanation that rebuts the assistant's supporting argument. To do this, the rules in the assistant's knowledge base are added, removed, or changed.

For instance, under the robot's standard operation policy, the assistant's knowledge base consists of the following rule sets:

Rule_0 :: at_home, auto_perm, grip_open implies pick_part;

Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: -sleep, auto implies auto_perm;

Rule_4 :: -powered implies sleep;

Rule_5 :: powered implies -sleep;

Rule_6 :: manual implies -auto;

Rule_7 :: -manual implies auto;

The standard policy states that the robot must move to pick up the part if it is in the home position, in automatic mode, and its gripper status is open. Once the part is picked, then it must proceed to place it, and subsequently it must go to its home position. The robot would repeat this procedure cyclically.

Based on this standard policy, the robot would move to pick up the part even though it is not available. When the human user seeks information from the assistant on this issue, the agent explains that from rules r7, r5, and r3 it is deduced that "the robot has automatic permission active." As it's gripper status is also open, therefore from r0 it is deduced that "it must go to pick up the part."

Considering the control scenario and the user's requirements, the assistant's knowledge base could be manually intervened upon by the user. Depending upon the agreement between the user and the assistant's claim and the explanation related to it, the user could modify the available set of rules in the knowledge base, create new rules or delete the existing ones. For instance, the user does not agree with the agent's explanation above about its standard policy of not moving when the part is absent. Therefore, the user now creates an exception in the knowledge base by modifying r0 and adding a new rule as r3, which can be seen below.

Rule_0 :: at_home, auto_perm, part_avl, grip_open implies pick_part;

Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: auto_perm, -part_avl, grip_open implies wait_part;

Rule_4 :: -sleep, auto implies auto_perm;

Rule_5 :: -powered implies sleep;

Rule_6 :: powered implies -sleep;

Rule_7 :: manual implies -auto;

Rule_8 :: -manual implies auto;

In r0, the user has incorporated a new predicate called *part_avl* which determines the status of the part's availability. Also, in the new rule r3, the user has annotated that if the robot has automatic permission and its gripper is open but the part is not available, then it must not move. Therefore, the following explanation is generated when the user seeks guidance from the agent regarding the robot not moving to pick up the part:

From rules r8, r6, and r4, it is deduced that "the robot has automatic permission to operate". As the part is not available and its gripper is open, then from r3 it is deduced that the robot should wait for the part.

2.2 Machine Coaching

With Machine Coaching, a human user may transmit their own preferences, expertise, and/or intuition to a machine by giving it guidance in the form of arguments in favor of or against specific behaviors. This takes place through a coaching cycle involving two parties, such as the user and the machine/agent, which we have already described in section 1.2 of Chapter 1. For a better understanding, we further elaborate on this in the context of industrial robot control. We presume that all of the following arguments are given in a common language between the human user and the coaching agent. Also, we assume that our agent is provided with a single rule which states that when the robot has automatic permission and the part is available on the conveyor, then it must move to pick the part from the conveyor. Assuming that the robot has automatic permission, and therefore, when we seek advice in the context of part availability, the agent suggests:

“I would suggest the robot to pick or grab the part as it has automatic permission, and the part is available”.

This advice is acceptable pertaining to the agent’s knowledge and the available context. Next, another part arrives on the conveyor and on querying the agent in the similar context we receive the following response:

“I would suggest the robot to pick or grab the part as it has automatic permission, and the part is available”.

Due to our assumption pertaining that our agent has only one guideline which says that when a part is available and the robot has automatic permission then it must pick the part, the above suggestion was quite evident. As such, a suggestion regarding the picking of the part when the robot already has a part in its gripper would result in a crash when the robot moves again to pick the new part from the conveyor. We proceed to advise our agent to be more careful by checking the state of the gripper through the following counterargument to its suggestion:

“Even through a part is available on the conveyor and the robot has automatic permission, it must pick the part only when its gripper is in open state”.

In light of this counterargument, we withdraw our prior advice of picking the part when the robot has automatic permission, and the part is available on the conveyor at the local level. The revised rule in the agent’s knowledge base would allow the robot to pick the part from the conveyor only when its gripper state is open, besides other mandatory requirements as discussed above. We could teach the agent more about our theory of controlling robots by giving it more suggestions in the same way.

When it comes to machine learning and declarative programming, machine coaching falls somewhere in between. One of the advantages of machine learning in this case is that the user's heuristics and preferences may be used to help the machine learn about its own domain. Contrary to popular belief, under the concept of "machine coaching," the machine is not given explicit instructions on where to look for or how to create relevant information. As a result, the user provides explicit instructions to the machine and, if feasible, the agent follows these instructions in some form. Machine coaching, on the other hand, makes the learning process obvious to the system's functionality while also enabling humans to educate machines in a more declarative manner. Every time it makes a decision, the machine notifies the user of the reasoning it used to arrive at that decision. When considering machine coaching as an interpretable learning approach (Arrieta et al. 2020), we can say that the user can grasp every step of the model's reasoning and learning process based on their own advice. This is because of the preceding information. Since each proposal is based on rules that derive from a user's idea about some domain, the user is supposed to be capable of simulating its function given all the information accessible to the machine. This makes machine coaching also a simulatable paradigm.

In the following section, we will discuss in more detail that how we can control and guide the industrial robot in a more declarative manner by grasping every step of the reasoning and learning process based on our own advice in the form of rules entailing a priority relation. Hence, understanding in a better way how Machine Coaching's high levels of interpretability and transparency may be attributed in large part to this. As we proceed, the majority of this dissertation's discussion on machine coaching follows the presentation in (Michael 2019).

2.2.1 Robot control and guidance through Machine Coaching

Moving ahead with the example discussed above in section 2.2 we describe that the robot has a task to pick the part from the conveyor, place it and then return to the home position. This policy is described in the form of three rules in the agent's knowledge base in the following manner.

Argument 1: Rule 1

When the robot has automatic permission, the part is available on the conveyor and its gripper is open then it must move to pick the part from the conveyor.

Under the language of Machine Coaching:

Rule_1 :: auto_perm, part_avl, grip_open implies pick_part;

Argument 2: Rule 2

When the robot has automatic permission, and the part is picked from the conveyor then the robot must move to place it at the place point.

Under the language of Machine Coaching:

Rule_2 :: auto_perm, part_picked implies place_part;

Argument 3: Rule 3

When the robot has automatic permission, and the part is placed then the robot must move to the home position.

Under the language of Machine Coaching:

Rule_3 :: auto_perm, part_placed implies go_home;

Assuming that the robot has automatic permission, and its gripper state is open, when we seek advice from the agent under the context of part availability (part is available), the agent suggests:

"As the robot has automatic permission, its gripper is in the open state and the part is available then it must move to pick the part".

Therefore, it presents Argument 1 which fires Rule 1 as part of its explanation. The head of Rule 1 "pick_part" acts as the control command for the robot that enables it to move and pick the part.

Now assuming that the robot has automatic permission, and its gripper state is closed, when we seek advice from the agent under the context of part availability (part is available) and part status (part is picked), the agent suggests:

"As the robot has automatic permission, and the part is picked then it must move to place the part".

Therefore, it presents Argument 2 which fires Rule 2 as part of its explanation. The head of Rule 2 "place_part" acts as the control command for the robot that enables it to move and place the part.

Further we assume that the robot has automatic permission, and its gripper state is open, when we seek advice from the agent under the context of part availability (part is available) and part status (part is placed), the agent suggests:

"As the robot has automatic permission, its gripper is in the open state and the part is available then it must move to pick the part". **– Rule 1**

"As the robot has automatic permission, and the part is placed then it must move to home position". **– Rule 3**

Therefore, under this circumstance, the agent fires both Rules 1 and 3 as part of its explanation for the relevant arguments, leading to a conflict. Therefore, as Rules 1 and 3 are prioritized differently, the robot's relevant move may or may not be advised depending on the order in which they are prioritized. Hence, in our case, as Rule 1 appears higher in order as compared to Rule 3, therefore it holds a higher priority. Therefore, the robot will move to pick up the part rather than go to its home position.

As the robot's standard task is made of three operations of picking, placing, and moving to its home position, based on the agent's above suggestion according to the rule priority, the robot fails to execute its final operation. So, our counterargument suggests that the robot should only pick up the available part if it is at the home position as below:

"Pick the part from the conveyor only if the robot is present at the home position".

Therefore, we withdraw our prior advice of picking the part when the robot has automatic permission, its gripper state is open, and the part is available on the conveyor at the local level. The revised rule in the agent's knowledge base would allow the robot to pick the part from the conveyor only when it is physically present at the home position, besides other mandatory requirements as discussed above. The revised rule would look like below:

Argument 1: Rule 1
When the robot is at home position, has automatic permission, the part is available on the conveyor and its gripper is open then it must move to pick the part from the conveyor.

Under the language of Machine Coaching:
Rule_1 :: at_home, auto_perm, part_avl, grip_open implies pick_part;

In this manner, we utilize Machine Coaching's functionality to control and guide the industrial robot in a more declarative manner by grasping every step of the reasoning process based on our own advice in the form of rules entailing a priority relation.

2.3 Argumentation & Learning in Machine Coaching

Here, we'll talk about how reasoning works in the context of machine coaching. Before we can proceed, we must define the arguments. Under Machine Coaching, arguments appear in the following manner:

- Internal arguments help the machine decide what actions, behaviors, or items to suggest to the user.

- Interaction with the user is accomplished via the usage of machine arguments. Actually, when the computer gives you advice, it also gives you an explanation of why it gave you that recommendation. In fact, the answer is based on the same internal logic that led the machine to reach this conclusion in the first place. So, one could say that the machine is clear and easy to understand, as described by (Arrieta 2020), since it lets the user see how it works.

- Any argument from the user is accepted by the machine. As we said in section 2.2, the user can give the machine a counterargument if they don't agree with the proposal or explanation it gives.

2.3.1 Description of Arguments under Machine Coaching Language

Under the machine coaching domain, the existence of an argument for some inference could be understood mathematically as described in (Michael 2019). Let *h* be some literal, *k* be a prioritized knowledge base; k = (R, p) and *c* be some context which is a set of pairwise non-conflicting literals. Assuming that (I = R U c), we conclude that A is an argument for *h* in *c* under *k* if A is a subset of I, *(*$A \subseteq I$), i.e., A contains few or all elements in I under the following conditions:

1. $A \neq \emptyset$
2. h could be inferred through repeated application of modus ponens starting from literals in $A \cap c$ using the rules in $A \cap R$
3. We cannot infer h from M, which is a proper subset of A, ($M \subset A$), i.e., M contains fewer elements than A, such that $\emptyset \neq M \subset A$ by substituting it in place of A as in (2).

The unique rule $\gamma \in A$ that has h as its head is also known as the argument's crown rule.

At least one literal from x or one rule from R must be present in an argument in order for it to satisfy the first criterion. For the second condition to hold, we need to be able to derive R from our hypothesis as an argument in support of our hypothesis. At this point, it should be noted that an argument may or may not include any rules at all, in which case h should be included. Since we presume that literals relating to a context are by default evaluated as truths that are true in a specific scenario inside our setting, such harmful arguments might be construed as restating some previously known fact. The last and most critical condition requires that arguments be limited in the sense that they comprise just what is necessary. There is no need to distinguish between arguments that vary by, say, one rule or one literal that does not have any additional consequences relating the target literal h.

Consider the following knowledge base *k* which has the following rules.

Rule_0 :: at_home, auto_perm, grip_open implies pick_part;

Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: -sleep, auto implies auto_perm;

Rule_4 :: -powered implies sleep;

Rule_5 :: powered implies -sleep;

Rule_6 :: manual implies -auto;

Rule_7 :: -manual implies auto;

Let us consider the following context *c*:

-manual; powered; at_home; grip_open, -part_picked; -part_placed;

Suppose we wish to see whether the knowledge base *k* above has any support for the argument for pick_part. We may deduce pick_part from (A = *c* ∪ *k)* ≠ ø by looking at the rules and literals in A thereby proving the first two conditions to be true. Now regarding the third condition, even if we delete -part_picked, -part_placed or rule Rule_1, we may still deduce pick_part from the new reduced set, which indicates the requirement about the argument's minimality is not met.

The following option, which is the only one that satisfies the third requirement listed in the definition of an argument, is the one and only viable alternative for the A.

A = {-manual, powered, at_home; grip_open, Rule_3, Rule_0}

Therefore, pick_part would not be inferred if any of the preceding rules were not applied to A. For instance, eliminating rule Rule_3 would prevent Rule_0 from being triggered, thereby preventing its head pick_part to be inferred.

It should be further understood that, given a context *c*, a prioritized knowledge base *k*, and a target literal *h*, even if there is an argument A for *h* in *c* under *k*, its status as a unique solution is in no way guaranteed, even in the event that it does exist. In order to provide evidence of this, let us extend *k* by including the following rules, each of which has a greater priority as compared to any other rule:

Rule_9 :: auto_perm, test_pick implies pick_part;
Rule_8 :: auto_perm, simulation_active implies test_pick;
Rule_0 :: at_home, auto_perm, grip_open implies pick_part;
Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: -sleep, auto implies auto_perm;

Rule_4 :: -powered implies sleep;

Rule_5 :: powered implies -sleep;

Rule_6 :: manual implies -auto;

Rule_7 :: -manual implies auto;

On extending our context *c* above with the literal simulation_active; we observe that pick_part is also deduced by a second argument B in the following way:

B = {-manual, powered, simulation_active, Rule_3, Rule_8, Rule_9}

A context *c* and a prioritized knowledge base *k* may serve as the basis for the development of arguments in support of, as well as opposition against, the same literal. In addition, it is also feasible to create arguments in both of these directions simultaneously. In point of fact, have a look at the following knowledge base:

Rule_11 :: at_home, auto_perm, grip_open implies pick_part;

Rule_10 :: auto_perm, test_place implies place_part;

Rule_9 :: auto_perm, test_place implies -pick_part;

Rule_8 :: auto_perm, simulation_active implies test_place;

Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: -sleep, auto implies auto_perm;

Rule_4 :: -powered implies sleep;

Rule_5 :: powered implies -sleep;

Rule_6 :: manual implies -auto;

Rule_7 :: -manual implies auto;

Under the following context *c*:
-manual; powered; at_home; grip_open, -part_picked; -part_placed; simulation_active;

The following argument supporting pick_part is contructed:

A = {-manual, powered, at_home; grip_open, Rule_3, Rule_11}

However, under the same context c as above, another argument supporting -pick_part is also constructed.

B = {-manual, powered, simulation_active, Rule_3, Rule_8, Rule_9}

2.3.2 Establishing the Boundaries of an Argumentation Framework

As we have described above in detail the arguments under Machine Coaching, we proceed to building an argumentation framework, as stated in (Dung 1995). Here we define α and ω as an ordered pair (α, ω) where α is an argument set and ω is a binary attack relation on α, such that, ω ⊆ α x α (Dung 1995). Furthermore, let context to be referred as c and prioritized knowledge base as k = (R, p).

We let α be the set of all arguments in c under k in terms of the total number of arguments (Michael 2019). We will make the following decisions within the ASPIC+ framework

(Prakken 2010) in regard to the ω relation of attacks between arguments through the approach described by (Michael 2019).

- It is impossible to refute the premises of context c, so we use it as an axiom set (Prakken 2010). Contextual knowledge is always deemed to be accurate since arguments cannot be challenged on their premises. The robot's working mode (auto/manual) cannot be questioned by any of the users, and the same holds true for all of the environmental facts where the robot operates. This is a way of saying that facts about a given scenario cannot be disputed.

- We have decided that all of the rules that make up our knowledge base should be defeasible. This means that if all of the rule's assumptions are true, then it is possible for the rule's head to be true (Prakken 2010). There are a few rules that hold true in every situation, whether in daily life or in a robot's operation, and our defeasibility factor aims to represent this truth. In reality, the vast majority of rules are context-sensitive.

- In between arguments, we prefer to respond to attacks with rebuttals (Prakken, 2010). According to the definition of refutation, an argument B challenges an argument A when the conclusion of B contradicts some of the conclusions that A has drawn, among other things. If you let attacks be rebutted, that means that for argument A to be true, no other counterargument can be triggered by a certain situation, so that every single conclusion of argument A is accepted in that context.

- Using the last-link approach (Prakken 2010), we can also sort arguments. For instance, if the final rule of A is preferred over the last rule of B according to the priority relation p, we say that an argument A is preferred over another argument B.

To put it another way, if one of the below requirements is met in a context c under a knowledge base k, we may say that an argument A supporting h, attacks another argument B that includes a rule r with head -h i.e. $(A, B) \in ω$.

- (h ∈ c), which means that it is an undeniable fact that the conclusion of argument A, h is correct.
- ($\gamma \nprec r$), where γ being the crown rule of argument A which implies that B′, the sub-argument of B that has r as its crown rule, is no less preferable than A.

Hence for a better understanding of the above, let us consider the two arguments A and B that we have described in the previous section.

A = {-manual, powered, at_home; grip_open, Rule_3, Rule_11}
B = {-manual, powered, simulation_active, Rule_3, Rule_8, Rule_9}

Clearly, A attacks B, but vice-versa. As a matter of fact, Rule 11 of argument A rebuts Rule 9 of argument B. Rule 11 is favored above Rule 9 since it occurs above Rule 9 in the knowledge base, however this is not the case for B because it may include a rule that leads to a conflict with A.

Furthermore, let us consider the following knowledge base where the rules that appear higher have a higher priority.

Rule_1 :: at_home, auto_perm, prod(Shift1), -prod(Shift2), grip_open implies -pick_part;
Rule_2 :: at_home, auto_perm, prod(Shift), start(Shift), grip_open implies pick_part;
Rule_3 :: at_home, auto_perm, prod(Shift), halt_prod, grip_open implies -pick_part;
Rule_4 :: time(X), ?<(X,9) implies halt_prod;

Consider the following context describing the industrial production scenario:
at_home; auto_perm; prod(morning); -prod(afternoon); grip_open; start(morning); time(8);

Under the above context, three arguments are detected:

A = {time(8), at_home, auto_perm, prod(morning), grip_open, Rule_4, Rule_3}
B = {at_home, auto_perm, prod(morning), grip_open, start(morning), Rule_2}
C = {at_home, auto_perm, prod(morning), -prod(afternoon), grip_open, Rule_1}

Given that ($\mathbf{p}_{\mathbf{Rule\_3}}$ < $\mathbf{p}_{\mathbf{Rule\_2}}$ < $\mathbf{p}_{\mathbf{Rule\_1}}$), where **p** describes the priority relation of the rules **R** in knowledge base k, argument B attacks argument A – because $\mathbf{p}_{\mathbf{Rule\_3}}$ < $\mathbf{p}_{\mathbf{Rule\_2}}$ and also argument C attacks argument A, since $\mathbf{p}_{\mathbf{Rule\_2}}$ < $\mathbf{p}_{\mathbf{Rule\_1}}$ among the three arguments above.

2.3.3 Grounded Semantics of an Argumentation Framework

Using a prioritized knowledge base k and a given context c, we'll look at what may be reliably deduced and see whether there's a computationally efficient approach for calculating the inferred literals. For a potential solution, we consider adopting Dung's Grounded extension of an argumentation framework, as proposed in (Dung 1995); Michael 2019). Prior to proceeding with the details, we define certain terms as a prerequisite. It is only admissible to argue that an argument A is acceptable with regard to a set of arguments, S, if for every other attack on A, there is another attack on B by another argument C, such that C ∈ S. (Dung 1995). That is, S has adequate evidence to counter all of A's attacks.

Let the argumentation framework be represented as AF = (α, ω). The characteristic function of AF is represented as a function $Z_{AF} : P(AF) \rightarrow P(AF)$, where P(X) determines set X's powerset such that $Z_{AF}(S) = \{A \in \alpha$: with regard to S, A is acceptable.$\}$ (Dung 1995). Therefore, with Z_{AF}, the first fixed point of Z_{AF}'s grounded extension GE_{AF} of AF is specified with regard to set inclusion (Dung, 1995). In order to throw more light on this notion, let us assume again that the argumentation framework is denoted by AF = (α, ω) and its characteristic function be denoted by Z.

Starting with the empty set ø, the following steps are taken:

- If $Z(\emptyset) = \emptyset$, as our first fixed point of Z is identified here therefore we say $GE_{AF} = \emptyset$ in order to conclude.
- If $Z(\emptyset) \neq ø$, we start finding $Z^2(\emptyset) \coloneqq Z(Z(\emptyset))$. When we identify $Z^2(\emptyset) = Z(\emptyset)$ that means $GE_{AF} = Z(\emptyset)$ in order to conclude.
- If $Z^2(\emptyset) \neq Z(\emptyset)$, similar to above we start finding $Z^3(\emptyset)$ and so forth.

The grounded extension of AF, GE_{AF}, can be computed with certainty since Z maintains set inclusion. For instance, let $X \subseteq Y$ and $A \in Z(X)$. Given that each attack against A is met by a rebuttal from X, the same logic must be applied to Y which ultimately results in $A \in Z(Y)$, and hence $Z(X) \in Z(Y)$. Thus, we may either stop at some fixed point or continue with a bigger set of arguments at each step. Therefore, under this context, proving that there exists at least one fixed point for Z would be sufficient to prove the computation of the grounded extension. Assume that $M \coloneqq \{X \subseteq \alpha: X \subseteq Z(X)\}$. Note that $M \neq \emptyset$ as $\emptyset \in M$ and assume that $Y \coloneqq \bigcup_{X \in M} X$ which is clearly understood from the fact that $M \neq \emptyset$. Initially we prove that $Y \subseteq Z(Y)$. As we know that $X \subseteq Y$ for any $X \in M$ to avail $Z(X) \subseteq Z(Y)$ for any $X \in M$ from the set inclusion fact of Z. Therefore, $Y = \bigcup_{X \in M} X \subseteq Z(Y)$ and, as we know that $Y = \sup M$, we obtain $Y \subseteq Z(Y)$. In order to get the inclusion inversely, note that as $Y \subseteq Z(Y)$, because Z holds set inclusion, we also know that $Z(Y) \subseteq Z(Z(Y))$. Therefore, we get $Z(Y) \in M$ from the description of M. So, as $Y = \sup M$ we also hold $Z(Y) \subseteq Y$. Hence, Y exists as a fixed point of Z through $Y = Z(Y)$.

Grounded semantics are also described as the skeptical semantics in (Dung, 1995) from their objective to record a set of inferences that may be securely established under a knowledge base k in a context c. To shed more light on this in order to determine which argument sets $A \in \alpha$ are acceptable by $\emptyset$, we calculate $Z(\emptyset)$. To put it another way, there is no argument B in $Z(\emptyset)$ attacking any argument A, therefore this clearly captures conclusions that don't require any further support. Hence, there is no argument included in GE_{AF} when $Z(\emptyset) = \emptyset$. If this is the case, we are unable to move further because $\emptyset = Z(\emptyset) = Z(Z(\emptyset)) = \cdots = Z_n(\emptyset) = \cdots$

We further calculate $Z(Z(\emptyset))$ under the situation $Z(\emptyset) \neq \emptyset$ which means that the set of arguments that, although being attacked by other arguments, are able to be defended by arguments that are not themselves attacked by any other argument. Because of this, we may confidently accept such judgments. Our argumentation framework's grounded extension may be produced in the same way as previously, if the following holds: if $Z(Z(\emptyset)) = Z(\emptyset)$ and $Z(Z(Z(\emptyset)))$ equals $Z^3(\emptyset)$, then we have constructed our framework's grounded extension. By extending $Z^2(\emptyset)$ to arguments that may be attacked by arguments

that are in turn attacked by other arguments in Z(∅), we advance our understanding of Z(∅) until we reach Z's first fixed point, proceeding in the same manner.
After our understanding of the above we say that the concepts grounded and skeptical seem reasonable in this situation because an argumentation framework is an ongoing process. Here we begin with arguments that don't need to be supported by other arguments. Then gradually we add new arguments so that they can be defended against other attacks by refuting the attacking arguments. Therefore, under grounded semantics the property of *groundedness* marks its importance in the sense that it permits safe conclusions to be made. Another important reasons for the selection of grounded semantics in the argumentation framework are that they bear resemblance with the human reasoning leading to a single model that is evident from the findings of (Stenning & Lambalgen 2012) and also, they facilitate efficient computation of the grounded extension of an argumentation framework (Michael, 2019).

We will now show an example of a grounded extension in our context of the industrial robot control and guidance after describing the motivations behind the concept of an argumentation framework's grounded extension as well as our personal motive for using it in our setting.

Let us again consider the three arguments and their attach relations that we had described in section 2.3.2 above such that $\alpha \coloneqq \{A, B, C\}$ and $\omega \coloneqq \{(C, B), (B, A)\}$.

A = {time(8), at_home, auto_perm, prod(morning), grip_open, Rule_4, Rule_3}
B = {at_home, auto_perm, prod(morning), grip_open, start(morning), Rule_2}
C = {at_home, auto_perm, prod(morning), -prod(afternoon), grip_open, Rule_1}

The grounded extension of the argumentation framework AF = (α, ω) can be computed as under:

- We begin by calculating $Z_{AF}(\emptyset)$. Given our attack relation, the sole argument that can stand alone without the help of other arguments is argument C, and as a result $Z_{AF}(\emptyset) = \{C\}$. We do not stop here as $Z_{AF}(\emptyset) \neq \emptyset$.

- In the subsequent step we calculate $Z^2_{AF}(\emptyset) = Z_{AF}(Z_{AF}(\emptyset))$. We also get $C \in Z^2_{AF}(\emptyset)$ from the fact that $C \in Z_{AF}(\emptyset)$ and Z_{AF} holds set inclusion. Also as $C \in Z_{AF}(\emptyset)$ defends A from the attack of B, we get $A \in Z^2_{AF}(\emptyset)$. Note that since B is attacked by C therefore $B \notin Z^2_{AF}(\emptyset)$.
- Further we calculate $Z^3_{AF}(\emptyset) = Z_{AF}(Z^2_{AF}(\emptyset))$. Note that as B is being attacked by C, it is impossible to include it in the argumentation framework's grounded extension. We get $Z^3_{AF}(\emptyset) = Z^2_{AF}(\emptyset)$ from the fact that Z_{AF} holds set inclusion therefore, {A, C} emerges as the first fixed point of Z_{AF}.

Therefore, {A, C} is found to be the grounded extension of the argumentation framework as discussed above. Hence, given the aforementioned three arguments and their attack relation, we can't successfully argue in favor of B although, as stated above, we can when it comes to A and/or C.

2.3.4 Learning under Machine Coaching

According to (Michael 2019), PAC semantics (Valiant 1984) are used to characterize learning in the context of Machine Coaching. If an algorithm can learn a feedback class X = X(α, β, γ) using a hypotheses class T for every $\delta, \varepsilon \in (0,1)$, every probability distribution P over inputs in α of size n and every f ∈ X of size s, provided access to δ, ε X, we claim that the algorithm is probably approximately conformant given the fact that it iteratively performs the following task:

i. Either through a random or active choice under P fetch an input (i ∈ α).
ii. From (j ∈ β), choose an output.
iii. Get some guidance f (i, j).

The algorithm finishes and returns a hypothesis h ∈ T after $[q\ (1/\delta\ ,\ 1/\varepsilon\ , n, s)]$ maximum time such that $f(\text{i}, h(\text{i})) = noGuidance$ except with probability δ and ε (Michael, 2019: 84). In addition, we shall argue that the method is an efficient conformant learner if q with regard to its parameters has a polynomial complexity.

According to the definition above, an algorithm that can capture a theory about anything (e.g., an industrial robotic arm) by making predictions about examples it encounters and receiving pieces of advice about them is possible under any desired probability of failure $\delta \in (0,1)$ and any desired probability of accuracy $\varepsilon \in (0,1)$. Then, we get a model of our theory that is correct, allowing for mistakes to occur with a probability ε, provided that the algorithm ends at some point, based on the two stated probabilities, the size of each example and the related pieces of advice. Moreover, the above model of our theory with a probability of δ may not be generated.

Apart from the high-level description of learnability above, we try to shed some light on the efficient learning algorithm described in (Michael 2019) in the form of a learning protocol. Here, our target learning theory, the knowledge base, is denoted by k and c represents the context of our choice. Our specific feedback class X is described as below:

- If a predicted rule is not found in k, it will be regarded *unrecognized* unless it is a rule that exists in our theory.
- If a predicted rule does not add to any argument in c under k, it will be deemed unnecessary or *superfluous*.
- If a rule does not exist in a prediction while it is included in k and its inclusion would lead to extra arguments from the machine, it is deemed *incomplete*.
- If there is no alternative argument in x under k that challenges an argument in the machine's prediction, then the argument is regarded *indefensible*.
- Otherwise, no responsive situation will arise.

The following algorithm is probably approximately conformant learner as described in (Michael 2019) provided the feedback class X as above and the linear order of the conflicting rules in k with respect to their priority relation.

- Let the initial knowledge base of the machine is represented by k = ∅.
- For each input i, that is randomly chosen:
    - Determine the prediction of the corresponding dual representation j.
    - As per the above protocol get the user’s advice f (i, j).
    - Facilitate the deletion of the rules that are deemed to be superfluous or unrecognized. Provide rules that are deemed to be incomplete or result in

the user's counter-arguments having priority higher compared to any existing rule in k.

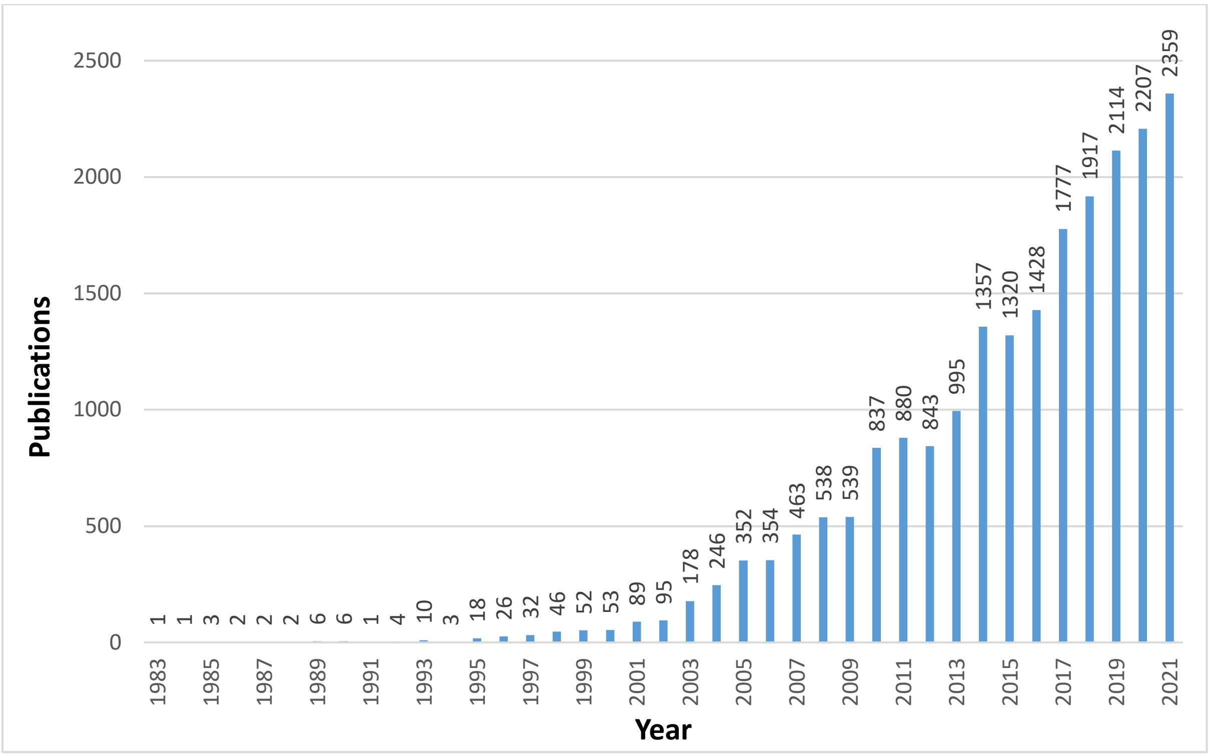
- Repeat the above process until the condition of no response arises for n consecutive cycles. Here n stands for the polynomial as per the PAC learnability definition above.

The description of the learning protocol above enables us to get an idea about how in section 2.2.1, our desired functionality is achieved where the user was able to rebut the machine's explanations through his counterarguments.

Chapter 3
Literature Review

The literature review consists of a detailed analysis of the interaction mechanism between a human and a robot as well as how defeasible logic and argumentation theory aid this domain.

3.1 Human-Robot Interaction (HRI)



Graph 1. HRI publication distribution over the years

As per the Scopus website, Graph 1 above displays the increasing amount of research being carried out over the years in the field of Human-Robot Interaction. With the growing number of publications, HRI researchers have attempted to answer the subject's challenges in a variety of ways. Figure 1 depicts the classification of these techniques.

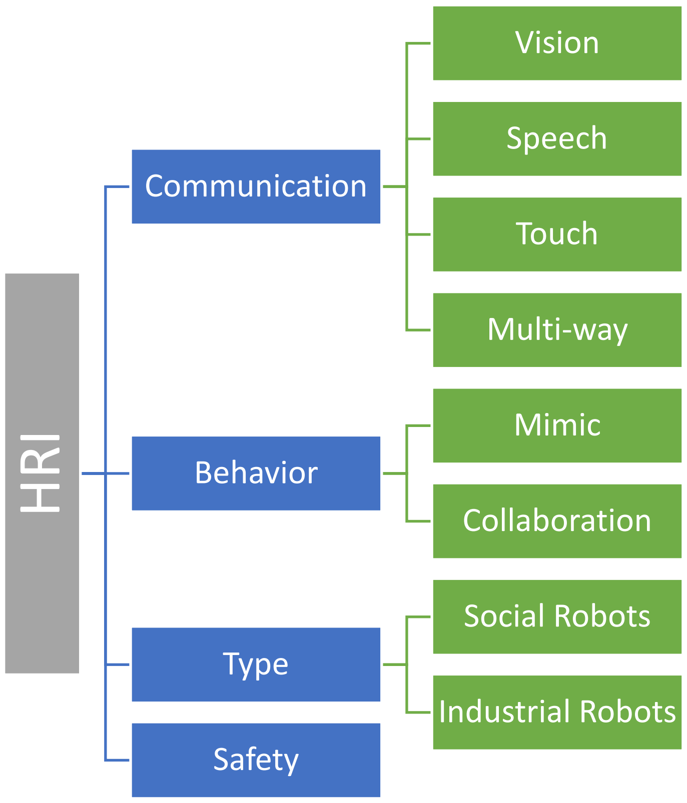


Figure 1. HRI classification

3.1.1 Interaction based on Communication

Various routes can be used to establish the needed communication through vision and speech for the establishment of HRI. The perception of the surroundings by a robot is provided by a collection of algorithms that use image sensors such as a camera. In the work by Xue et al., they used a six-axes robot and an industrial camera to execute gas metal arc welding with an error rate of less than 0.5mm with an operator (Xue et al. 2021). Another investigation was conducted using a plastic robot joint with a camera. People touched this joint, and the direction in which they wished to move the robot according to the propensity of this joint was identified using the camera (Oliveira et al. 2020). In another study, deep learning algorithms were used to locate the spots in the image where the robot could grasp the items (Bergamini et al. 2020). The pictures captured by the Red-Green-Blue-Depth (RGB-D) sensor in Li's work were characterized using deep learning methods to control the robot arm in the simulation environment (Li 2020). In a research similar to that of Li's, the depth picture captured by the RGB-D sensor was recognized using deep learning to operate the Baxter robot using the human arm's movement (Fang et al. 2020).

Speech is another kind of communication that is utilized as frequently as vision in HRI. Chen and his colleagues employed speech-based communication to enable their robot to recognize human emotions (Chen et al. 2020). Another study (Liu et al. 2018) calculated Mel Frequency Cepstral Coefficients (MFCC) over speech with 90.28 percent accuracy to identify emotional interactions between humans and robots using linear discriminant analysis (LDA) and principal component analysis (PCA). It was possible to execute the needed tasks by talking to the Khepera II robot. 2007). Ding and Shi used the microphone array on the RGB-D sensor to drive a humanoid robot using a speech recognition program based on support vector machines (SVM) and Gaussian mixture models (GMM) (Ding & Shi 2017). (Gunawan et al. 2017) used voice recognition to operate another humanoid robot, Rapiro.

Touch is one of the ways that humans communicate with one another. The detection of the robot's touched site and the assessment of the touch kind were explored in research on this communication technology, which used three microphones mounted on a humanoid robot (Gamboa-Montero et al. 2020). A success rate of around 86 percent was attained. Erica, a humanoid robot, was given the ability to anticipate touch and glance in that direction using two RGB-D sensors in another study (Shiomi et al. 2018). Another research on Erica investigated whether joyful and sad sentiments could be communicated to the other individual by touch, and it was successfully discovered (Zheng et al. 2020). Kim and his colleagues used nine touch sensors and one accelerometer mounted on the head of a humanoid robot to detect whether humans touched it with one of the four forms of touch they identified using a temporary decision tree method (Kim et al., 2010).

When interacting with one another, people frequently utilize many communication channels. In HRI, the scenario is similar. These communication channels are frequently used jointly in the literature. For example, the KUKA LBR iiwa industrial robot is controlled using both visual and tactile communication channels (Cherubini et al. 2015). A voice and visual communication channel was employed with a humanoid robot in another study (McColl et al. 2017), and the relevance of body language derived from the picture was underlined. Both the voice and hand gesture detection routines on the GOOGOL GRB3016 and KUKA KR 6 R700 robots have successfully followed the trail (Du et al. 2018). This voice and hand gestures approach was utilized to control a robot arm in another investigation and the only variation between the two experiments above is that the sensors used for hand movements are different (Yongda et al. 2018). In another investigation, a robot that provided human assistance at an event interacted with the participants through speech and vision (Jensen et al. 2005).

3.1.2 Interaction based on Behavior

In order for robots to perform desired tasks, they must imitate humans or collaborate in order to benefit from people's experiences. This difficulty was put forth in one of the studies that required the understanding of the location and orientation of the object first in order for the robot to fetch it. The robot was instructed to fetch the object like humans that was demonstrated by the people in order to tackle this difficulty, and successful results were obtained (Canal et al. 2016). In another study, data from a surface electrode Electromyography (EMG) sensor connected to people's arms was used to create a robotic hand that imitated the human hand (Meattini et al. 2018). The copying of humanoid facial emotions such as pleased, sad, and astonished by a robot has also been studied in the literature (Cid et al. 2013, Ge et al. 2008). In another research, Erica, a humanoid robot, was able to simulate a tourism agency representative owing to the transfer of speech knowledge gathered via human-human conversation (Doering et al. 2019). A laparoscopic robot is used to guide the camera mounted on it according to the head motions of the surgeon doing the surgery to assured that the surgeon enlarged the region that he focused on (Fujii et al. 2018). A robotic assistant has been built to aid people during the cutting process of wooden or plastic objects in other investigations (X. Chen et al. 2020, Peternel et al. 2017). Another field where human-robot collaboration may be seen is welding. According to research in this field, light-weighted robots assist people to achieve smoother welding results (Erden & Billard 2015a, Erden & Billard 2014, Erden & Billard 2015b). The method of transporting heavy goods alongside people was taught to the robot through imitation to achieve collaboration in research focused on robot imitation and collaboration. In extension to this research, the robot was taught to collaborate with humans throughout the furniture assembly process (Rozo et al. 2016).

3.1.3 Interaction based on Robot Type

The most often employed robot kinds in HRI are the social and industrial robots, which will be covered in this section. Robots that can be a part of people's daily lives are known as social robots. In Tseng and his team's investigation, the robot was asked to determine whether or not socially distant people require assistance. In order to do this, the robot would respectfully interrupt a group of people's conversation to ask if they had any questions, or if a person directly inquired, the robot would answer their queries (Tseng et al. 2016). The feeling of trust in the robot under social HRI was investigated in another study, and it was discovered that people's trust in robots increased following a gaming session (Aroyo et al. 2018). A humanoid robot was used to deliver 121 treatment sessions to pediatric patients with autism spectrum disorder over the course of four weeks in another research (Melo et al. 2019). Another research used the Nao robot to help 8 young individuals with autism spectrum disorder play the Tangram puzzle game in instructional and peer mode which had positive outcomes (Bernardo et al. 2016).

Industrial robots are another category of robots that are frequently employed in HRI. Kronader and Billard used KUKA LWR and Barrett WAM industrial robots to construct physical HRI and user interface applications for beverage filling. When both the softwares were tested on the task load index and system usability scale by two groups of 14 people, the physical HRI was observed to be more responsive (Kronander & Billard 2014). In another research on the KUKA LWR robot, it was found that using a combination of variable impedance and kinematic redundancy resolution in physical HRI applications may achieve the best accuracy and processing time (Ficuciello et al. 2015). With the aid of the developed wearable sensors, the ABB YuMi robot was able to imitate human arm movements in another study (Zhou et al. 2020, F. Chen et al. 2019).

3.1.4 Safety in HRI

Governments and employers have agreed on certain safety regulations to mitigate the dangers associated with the human work environment that have evolved as a result of the industrial revolutions. The ISO 10218 standard and the ISO/TS 15066:2016 technical specification (Robla-Gomez et al. 2017) govern the connection between people and robots in terms of safety. More specifically, the safety requirements for industrial robots are established by ISO 10218, whereas safety standards for robot and controller manufacturers are established by ISO 10218-1, and safety standards for robot and assistive device integrators are established by ISO 10218-2. The ISO TS 10566 standard, on the other hand, contains standards for collaborative robots.

Due to the advancement in technology and continuous evolution of the human-robot interaction domain, the vast majority of HRI scientists are focused on developing a safer HRI. In the research conducted by Liu and Wang, RGB-D sensors installed in the robot's working environment were used to identify whether people approached the robot, allowing a new trajectory to be designed to prevent an accident (H. Liu & Wang 2021). Another research (Landi et al. 2019) used a Kinect sensor to predict if a person in the robot's working area would approach the robot. In another investigation, infrared sensors were mounted at intervals over the designated joint of the ABB IRB 140 robot, and the information obtained from these sensors was calculated. With this, the presence of a human in the vicinity who may constitute a threat was perceived (Buizza Avanzini et al. 2014). Raiola and his colleagues, on the other hand, have successfully established a safe HRI by analyzing the energy of the robot joints without the need for an external sensor (Raiola et al. 2018).

3.1.5 Use of Machine Learning & Artificial Intelligence in HRI

Machine learning is an application area of artificial intelligence that learns by analyzing data, detecting patterns, and making inferences without the need of human programming. With the rise of this branch, algorithms have started to leave the field of mathematics and enter the field of natural sciences, as the nature of machine learning is to make statistical inferences from the data representing the existing world and to operate the patterns.

Perhaps the biggest contribution of machine learning algorithms to programmers is that they are very time-saving. For example, if a programmer wants to write a spelling correction program code, instead of describing all the examples of spelling errors one by one, he/she can feed a set of existing spelling error examples into machine learning algorithms and come up with a much more comprehensive program in a very short time. Another area of use is the ability to customize products for specific audiences. Let's say that the spelling correction program produced by the programmer was very successful and he/she wanted to adapt it to other languages. Thanks to machine learning algorithms, the program can be adapted to other languages by operating the same model for the languages closest to the language in which the program is written. Another possibility offered by machine learning is that it can distinguish and recognize data outside the threshold of human perception.

Of course, the contributions of machine learning to a programmer's life are numerous, but what are the most common technologies with which we interact as humans in our daily lives that incorporate these machine learning algorithms?

- *Recommendation Systems*
  One of the areas where machine learning is most commonly used is recommendation systems, which allow websites to offer a more personalized experience to their users. For sites such as Netflix, Youtube, or Amazon, which have a wide range of content and are growing and developing every day, the ability to bring this content to users at any time and in a need-oriented manner is one of the most important factors that keeps these sites ahead of the competition. For example, Netflix can show different content to different users to promote a production on its interface. On online shopping sites such as Amazon, on the other hand, recommendation systems can offer options about product combinations that the user can add to the shopping cart.

- *Spam/Mail Subject Filtering*
  All of us, at some point in our lives, have suffered from junk mail that has fallen into our mailbox. When this is the case, it can be quite difficult to distinguish important messages from others. But now, mail services such as Gmail make the user's job much easier by scanning the subject and content of the mails received by the user with machine learning algorithms and placing them in appropriate categories. A similar system is also used to filter spam and phishing emails.

- *Search Engines & Internet Search*

Search engines such as Google, Yandex, etc. that we use almost every day in our lives to access the resources available on the internet and that act as a bridge between the human user and the internet incorporate machine learning algorithms as their prime working mechanisms. These algorithms are used in many stages, such as indexing data on the internet (web crawling), optimizing the ordering of results, and extracting the most appropriate results for the context of the search term.

- *Smart Personal Assistants*

  If anyone has ever spoken to their smart assistant on their phone or at home, they've experienced natural language processing and deep learning algorithms, which are a subset of machine learning. The difference between these algorithms and classical machine learning is that they can also process unstructured data.

  While 80% of the data we produce is unstructured, only 20% is classified as structured. Unstructured data is a data group with a wide variety such as visual, audio, video, mobile activity, social media activity and is therefore difficult to analyze, whereas structured data includes data that is easy to analyze and collected in accordance with a predetermined data model.

  Voice-activated devices like Siri, Alexa, Cortana, or Google Assistant have already started coming into our lives and making them easier by hearing accurately what is being said, understanding the context, and responding in the most appropriate way for the situation and in a language that we can understand. In order to do all of these and to make sense of the unstructured data in the form of natural language, machine learning algorithms are used.

While most of the above-mentioned systems have the ability to learn from the data supplied explicitly through questionnaires or implicitly through the learning algorithms, some specific systems could also be coached in real time in order to achieve outcomes that are most suitable to the user’s desires. For example, smart personal assistants could be taught to play a certain song when the user is sad instead of a random song that the assistant thinks fits the mood.

At today's technological level, the employment of artificial intelligence and machine learning algorithms in HRI is unavoidable so that the robots can provide the most suitable responses to unanticipated inputs. The DT (Decision tree) algorithm was utilized in Kim and his team's work to classify the touch types in the HRI established using the touch channel (Kim et al. 2010). Emotion categorization was accomplished with the SVM machine utilizing characteristics collected from facial pictures (Ge et al. 2008). In another study, the data collected with the surface EMG sensor was classified using SVM, and the robot reproduced human hand motions (Meattini et al. 2018). The GMM (Gaussian Mixture Model) is used by the robot to learn the movement of the parts required for table construction (Rozo et al. 2016). Fujii and his colleagues used the k-means algorithm and the Hidden Markov Model (HMM) to move the camera on the laparoscopic robot (Fujii et al. 2018). In speaker and speech identification software built for use in humanoid robots, structures such as SVM, GMM, and fuzzy classifier are employed (Ding & Shi 2017). A two-layer fuzzy multiple random forest method was used in another work to investigate emotion prediction without speech (L. Chen et al., 2020). Fang and his colleagues used a Deep Neural Network (DNN) to interpret the depth image and used the human arm angles in the image to alter the arm angles of the Baxter robot (Fang et al. 2020). Using Deep Convolutional Neural Networks (DCNN), the robot was able to locate locations where it could grasp unfamiliar items in other research (Bergamini et al. 2020). Another study utilizing DCNN found that the generated system could recognize two hands with the aid of DCNN and use the motions of these hands to establish human-robot communication (Gao et al. 2019). In another similar work (Li 2020), it was attempted to simulate human arm motions by the robot with the use of DCNN and RGB-D sensors. PoseNET, a DCNN model, ensures the human's safety while working with the robot (H. Liu & Wang 2021). According to research done for a robot to learn humanoid qualities, reinforcement learning allows the robot to have both social and decision-making characteristics similar to humans on certain topics (Qureshi et al. 2018).

3.2 Defeasible Logic

Introduced for the formulation of defeasible reasoning, defeasible logic (Nute 1994) is a logic in which the relationship between the consequences of the logical assertions is not monotonic. Due to this fact, defeasible logic could be referred to as non-monotonic logic.

It can therefore reason on contradictory logical statements by employing defeat relations among defeasible logical propositions, unlike in monotonic logic. Defeasible logic is built up of strict evidence or facts, strict guidelines or rules, defeasible rules, undermining defeaters, and priority relations among defeasible rules.

- *Strict Evidence/Fact*
  It is represented with an atomic formula α whose complement is denoted by $\neg\alpha$.
  E.g. *animal(Tiger)* is an strict evidence.

- *Strict Guidelines/Rules*
  They are denoted by implication representation in the form ($\alpha_0 \Leftarrow \alpha_1 \wedge .... \wedge \alpha_n$) which states that if ($\alpha_1 \wedge .... \wedge \alpha_n$) is true then α_0 should also be true where α_0 is referred to as the rule's claim. Therefore, strict rules are said to be undefeatable rules.
  E.g. *animal(X)* $\Leftarrow$ *tiger(X)* which states that *a tiger is an animal*.

- *Defeasible Rules*
  Like the strict rules, they are also denoted by implication representation in the form ($\alpha_0 \leftarrow \alpha_1 \wedge .... \wedge \alpha_n$). However, they bear weaker connections as compared to the strict rules as they are prone to be defeated by other defeasible rules. Two rules (α and $\neg\alpha$) that make conflicting assertions or claims clash and may defeat each other.
  E.g. *herbivorous(X)* $\leftarrow$ *animal(X)* and $\neg$*herbivorous(X)* $\leftarrow$ *tiger(X)* according to which *animals may be herbivorous* and *tigers may not be herbivorous*.

- *Priority Relations*
  Among the non-strict rules, priority is established by the priority relations, which are acyclic binary relations specified in order to choose amongst the defeasible rules based on their defeat relations or priority. As we know from the previous example, *a tiger is an animal. Animals may be herbivorous* and *tigers may not be herbivorous*. Therefore, we may claim both that *a tiger is an animal, so it may be herbivorous*, or that *a tiger may not be herbivorous*. Hence, to reason on them, a defeat relation between these rules is required. When the defeasible rule that *a*

tiger may not be herbivorous gets a higher priority, it is concluded that even in the presence of the opposing defeasible rules that *a tiger may not be herbivorous*. In a real-time scenario, the definition of these priority relations is user-dependent.

- *Undermining Defeaters*

  These are the less defeasible versions of defeasible rules. Because of their weaker claims, they are not employed as supporting rules in inference. Their objective is to keep us from making decisions that we might otherwise make.

3.2.1 Defeasible Logic Programming

As per (García & Simari 2004), DeLP came into existence due to the need for combined results that involve Logic Programming and Defeasible Argumentation. As DeLP is based on defeasible logic, therefore, it also incorporates its characteristics such as evidence or facts, strict rules, defeasible rules, and priority relations. DeLP enables us to build a defeasible argumentation inference process for queries such as YES, NO, UNDECIDED, and UNKNOWN. A claim or its counterpart is justified if the answer is YES or NO. UNDECIDED indicates that neither the claim nor its complement is supported by evidence. If the claim does not exist in the language, the query returns UNKNOWN.

A Defeasible Logic Program is an endless set of evidence/facts, stringent rules, and defeasible rules that are used to come up with answers to questions. During the inference cycle of DeLP, defeasible argumentation is used. An argument Ai among a group of arguments (Ai ϵ ω) connects a claim, claim(Ai) to a consistent collection of evidences, stringent rules, and defeasible rules. The support set is always the smallest consistent set that can be used to prove the claim.

Table 1 below describes a DeLP example which is made up of facts and defeasible rules as described in (García & Simari 2004).

$a \leftarrow b$	$\neg b \leftarrow e$	$\neg b \leftarrow c \wedge f$	$\neg f \leftarrow i$
$b \leftarrow c$	e	$f \leftarrow g$	i
c	$\neg f \leftarrow g \wedge h$	g	$\neg h \leftarrow k$
$\neg b \leftarrow c \wedge d$	$h \leftarrow j$	k	
d	j		

Table 1. Facts & Defeasible Rules

It is to be noted here that the negation of x is represented by $\neg x$. Also, if X_j contains all of X_i's supports, X_i is considered a sub-argument of X_j. If claim(X_i) = α and claim(Xj) = $\neg\alpha$, the two arguments, X_i and X_j, are said to be in conflict with each other. When argument X_i is in conflict with a sub-argument of X_j, we say that argument X_i attacks X_j. Therefore, we may derive the arguments as in Table 2 below from the facts and defeasible rules mentioned in Table 1 above.

X_i	*Claim(X_i)*	*X_j (Supporting Argument set of X_i)*
X_1	a	$(a \leftarrow b), (b \leftarrow c), c$
X_2	$\neg b$	$(\neg b \leftarrow c \wedge d), c, d$
X_3	$\neg b$	$(\neg b \leftarrow c \wedge f), c, (f \leftarrow g), g$
X_4	$\neg b$	$(\neg b \leftarrow e), e$
X_5	$\neg f$	$(\neg f \leftarrow g \wedge h), (h \leftarrow j), g, j$
X_6	$\neg f$	$(\neg f \leftarrow i), i$
X_7	$\neg h$	$(\neg h \leftarrow k), k$

Table 2. Derived arguments from Table 1

Table 2 only lists the argument (X_1) that asserts a and the arguments that may be used to refute any other argument.

From above, the argument X_2 opposes argument X_1 because it claims $\neg b$, while X_1's sub-argument asserts b. If argument X_i rebuts argument X_j and is better than X_j in terms of the given evaluation criteria, argument X_i becomes a blocking defeater for argument X_j. Therefore, the arguments are blocking defeater for each other if the compared arguments are equivalent in terms of the evaluation criterion.

In DeLP, any claim's conclusion is obtained by developing its appropriate argumentation lines (AL), where each subsequent argument is in conflict with the sub-arguments of the prior one in an ordered sequence of arguments ($X_0, X_1 \ldots X_n$). The claim is considered to be justified if an argument that supports it cannot be refuted in any accepted argumentation line.

As per (García & Simari 2004), a valid argumentation line shows the following properties.

- It is a set of arguments with a finite number of entries.
- No argument that exists in AL is a sub-argument of any of the preceding arguments.
- No conflicting arguments exist between the even indices ($X_0 \sqcup X_2 \sqcup X_4 \sqcup$...) and the odd indices ($X_1 \sqcup X_3 \sqcup X_5 \sqcup$...) argument sets.
- If a blocking defeater exists in AL then the following argument must be a valid defeater

3.2.2 Defeasible Logic Programming Applications

As the primary aim of DeLP was to bring the results of Logic Programming and Defeasible Argumentation together (García & Simari 2004), it paved ample scope for research that offered several expansions to the basic formalism. In the work by (Governatori 2004) defeasible rules were used to expand description logic. As per (Governatori & Terenziani 2007), temporal rules have been added to DeLP to cope with long-term facts and delays between rules. The robotic domain has seen numerous applications of defeasible logic programming. DeLP has been employed in Khepera mobile robots to form a layered framework in order to deal with contradictory information [Feretti et al. 2006]. In the work by (Feretti et al. 2007), DeLP is used to determine the activities of a cleaning service robot. In a more recent and common robotic application, DeLP is also offered for resolving potential collisions between unmanned autonomous vehicles (UAVs) via communication (Lam and Governatori 2012).

As defeasible logic forms the basis of argumentative reasoning, it has been incorporated in various approaches lately that introduce learning and reasoning capability in smart machines and devices. A similar approach that formulates argumentative reasoning to offer machine coaching is described in (Michael 2019), where all the inputs are given in the form of rules or literals in first-order language. A fair implementation of this machine coaching formulation is seen in PRUDENS (Personalized User-Deliberation Support), which is a software tool that has been developed by the research group of the

Computational Cognition Lab of the Open University of Cyprus led by Dr. Loizos Michael. Through an interaction interface, it makes it easier for both humans and machines to learn new things.

3.3 Argumentation based Communication Theory

Argumentation is an important part of communication, and it has been around for generations in our civilization. This approach finds its roots in the philosophical theory of justification and reasoning, which was initially based on oratory and reasoning proposed by Aristotle. However, with time, Aristotle's views were rejected and questioned by scholars, and a premise for argument that was larger than that proposed by Aristotle was discovered. Several scientists attempted to create ways for people to get support for their thoughts and views between the 60s and 70s. Many others have developed reasoning in other ways as well.

Communication has played an important part in our evolution as human beings convey their thoughts through speech. The speaker would provide knowledge while conversing, and the listener would listen. The listener must be able to distinguish between legitimate information and falsehoods and treachery in this situation. As per Dan Sperber, the listener must possess a mechanism that would distinguish between the legitimacy of the received information. For instance, we believe what is taught in class as we trust the teacher and the school. Among the various scholars that have proposed different approaches to argumentation, Stephen Toulmin's argumentation theory has achieved wide recognition in this domain. Being an English philosopher and logician, in his work he has described how an argument is built up through its elements (Toulmin 2003). A brief explanation of these elements can be seen below.

- *Claim*
  It is a statement presented by the speaker or listener for the acceptance of the information conveyed through it as true. For instance, one will not perform an action when asked by someone unless inquiring and understanding the requirements associated with the action. Therefore, one will ask for the support of their claim, which would require the grounds associated with the claim. For instance, *Sam is a British national*.

- *Ground*

  It is the foundation of the claim which might be made-up data used to influence the audience. More precisely, it is the foundation upon which an argument is built, and it may also provide proof for reasoning. Therefore, the information plays a vital role in the persuasion mechanism. For instance, *Sam was born in Bermuda.*

- *Warrant*

  The justification of the claim relies on the warrant which determines that the ground to the claim is proper. It might be a simple statement or a lengthy argument which might be correct, implied, or unstated. For instance, *someone born in Bermuda is usually of British nationality.*

- *Backing*

  Backing is directly proportional to the warrant associated to a claim as the warrant receives additional support when the argument receives backing. For instance, *the rights granted by the law.*

- *Qualifier*

  Terms like 'most,' 'generally,' 'always,' and 'sometimes' that limit the comprehensiveness of the claim are referred to as the qualifiers. For instance, *probably.*

- *Rebuttal*

  It refers to circumstances that are not covered by the warrant. Simply, rebuttals are the statements that explain circumstances in which the argument will be invalid. The denier serves as the annulter and allows for the demonstration of revocable logic. For instance, *both parents might be of foreign nationality or hold American citizenship.*

3.3.1 Use of Argumentation theory in HRI

Under standardized cases, it has been observed that the human-robot relationship is merely a master-slave relationship where the slave (robot) works as per the guidelines defined by the master (human). This methodology creates a barrier in the communication where the slave (robot) could also propose options through its analysis which could be utilized by the master (human) for the optimum workflow of the task. However, the functioning of the robot is only limited to reporting errors besides discussing the reasons for the failure. The robot cannot anticipate for better opportunities or stop its current task to suggest alternative actions. In order to achieve a more dynamic HRI where both the humans and the robots could engage in a dialogue exchange, the use of argumentation theory comes into recognition to be utilized in this domain. (Sklar et al. 2013) propose an approach where a human and a robot are engaged in a dialogue-based game. Here they use an argumentation-based dialogue protocol to exchange inflictions to obtain an agreement on goals and plans. The dialog protocols between the human and the robot which include information (advice) seeking, inquiry, persuasion, negotiation and delibration are modeled for a dynamic setting implementation. Using a similar approach (Sklar & Azhar 2015) demonstrate how argumentation-based dialog system enables a dynamic HRI under a gameplay environment where they focus on a Treasure Hunt game. (Black & Sklar 2016) have explored the addressing of the issues pertaining to trust, privacy and ethics when it comes to sharing information and modeling others' beliefs through computational argumentation strategies. In another study by (Azhar & Sklar 2017), objective and subjective performance analysis has been studied for a shared decision making in a human-robot team. Positive results were achieved under human-robot collaboration and argumentation based collective decision making whereas subjective results varied when it comes to the preference of choosing a robot over a human as a teammate. As argumentation-based communication facilitates a mutual explanation and understanding based outcome generation between two parties, therefore scholars have proposed studies that utilize argumentation framework to coach a machine or a cognitive agent. As per (Michael 2019), a calling assistant could be coached through a similar approach to accept or reject a call based on the user's location, time frame, caller priority, emotion etc.

3.3.2 Argumentation in Machine Learning

As machine learning is the process of learning from data and improving over time, its application has become increasingly essential in recent years since it is used in almost every domain around us. Argumentation and Machine Learning (ML) are brought together in a variety of contexts, such as to enhance ML or to aid in the extraction of arguments (Lippi & Torroni 2016, Grosse et al. 2015). The existing methods of machine learning that involve argumentation differ in their machine learning approach and strategy. For the supervised learning domain, the CN2 rule induction algorithm (Clark & Niblett 1989) has been improvised in the Argumentation-Based Machine Learning (ABML) approach by (Možina et al. 2007). (Bratko et al. 2009) propose the Argument-Based Inductive Logic Programming (ABILP) approach that finds its roots in Inductive Logic Programming (ILP). Another study by (Amgoud & Serrurier 2007) emphasizes on the version space learning framework (Hierons 1999) in their concept learning technique that is completely based on argumentation. Other studies by (Ontañón et al. 2012) which describe multi-agent inductive concept learning, and by (Ontañón & Plaza 2014) which describe the computational implementation of (Ontañón et al. 2012)'s work use concept learning (Hierons 1999) for supervised learning. In their works, (Carstens & Toni 2015, Carstens & Toni 2016) describe the Classification Enhanced with Argumentation (CleAr) technique that has been tested using Naive Bayes classifiers (John & Langley 1995), Support Vector Machines (SVMs) (Cortes & Vapnik 1995), and Random Forests (Breiman 2001). Therefore, it works as a global technique for any supervised learning methodology. For the unsupervised learning domain, (Gómez & Chesnevar 2004) use the Fuzzy Adaptive Resonance Theory (ART) model (Carpenter et al. 1991) for the hybrid approach they propose in their work. For the reinforcement learning domain, the Argumentation Accelerated Reinforcement Learning (AARL) described by (Gao & Toni 2013, Gao & Toni 2014, Gao & Toni 2015) finds its roots in SARSA (Rummery & Niranjan 1995). We can say that, depending upon the reasoning approach and the argumentation technique employed, the existing machine learning techniques differ from each other, which is evident from their varied outcomes, which are not limited to performance improvement and transparency enhancement through improved explanation.

Chapter 4
Methodology

As argumentation facilitates a bilateral reasoning framework under which knowledge could be developed through arguments or explanations, we utilize this approach through Prudens in order to develop a Human-Robot Interaction mechanism that would offer information flow between Prudens and the industrial robot.

4.1 Implementation

An approach to interact, control, and coach an industrial robotic arm through argumentative inferential deductions has been proposed below. PRUDENS has been used as a tool which facilitates the argumentative behavioral machine coaching paradigm through inference generation, which is based on the rules present in its knowledge base under the contextual information that has been sent by the robotic arm during its operation. The system has been implemented using the Web Interface (HTML) version of PRUDENS, a product of the Computational Cognition Lab of the Open University of Cyprus led by Dr. Loizos Michael, which can be downloaded from the link https://github.com/VMarkos/prudens-js and Staubli Robotics Suite (SRS) 2019.9.0, which is a licensed software development and simulation program for Staubli Robotic Arms. The robotic arm model used for the system integration is the Staubli Tx2-40, which consists of six degrees of freedom, each with a specific joint limit. The arm has a maximum extension of 515mm and a load capacity of 2Kg.

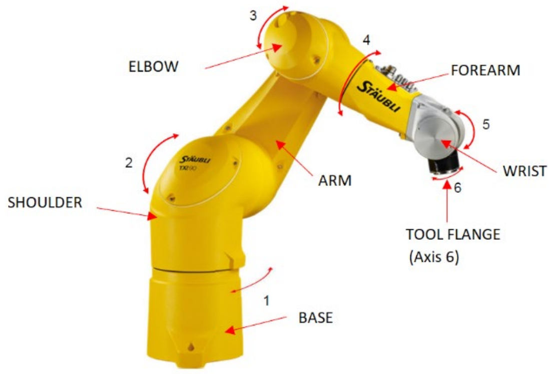


Figure 2. Robotic Arm

The arm periodically sends its status information to PRUDENS, which, based on the rules present in its knowledge base, builds an inference which is sent back to the robotic arm. Based on the inference, the robotic arm performs an action to pick a part, place the picked part, move to the home position or stop working. The coaching of the arm is facilitated by the modification or addition of the rules in the knowledge base of PRUDENS. The implementation has been tested in the simulation environment in SRS as well as on the real robotic arm.

4.1.1 Architecture

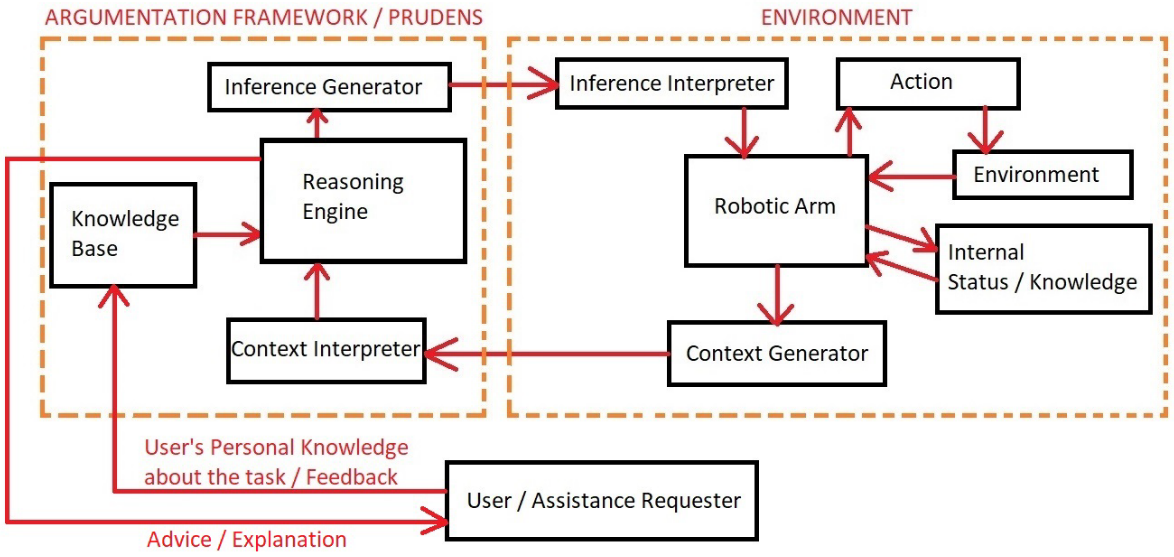


Figure 3. System Architecture

At a high level, the proposed system is a cognitive agent that models the human argumentation capability in order to interact with the environment where a robotic arm or simply a machine is present. Therefore, as a whole, the architecture is comprised of an Argumentation Framework, which has been facilitated by the PRUDENS software interface, the Environment, which consists of the Staubli Robotic Arm, and Feedback or Knowledge Input by the User or the Assistance Requester about a specific task.

4.1.1.1 Argumentation Framework

The Argumentation Framework consists of four main blocks, which are the Knowledge Base, the Context Interpreter, the Reasoning Engine, and the Inference Generator. The knowledge base consists of some predefined rules related to the ecosystem in which the interface is established. The various types of knowledge in this module are the expert knowledge, which is hard coded in the form of predefined rules; the commonsense knowledge, which is machine learned based on the robot's status and environmental data; and the knowledge based on personal biases through user feedback. The context is built up of the internal knowledge based on the arm's status and its environment, which arrives cyclically to the context interpreter. The job of the context interpreter is to prepare the incoming context information for the deductive interpretation required for reasoning. The argumentative reasoning is facilitated by the Reasoning Engine, which provides an inference based on the rules in the knowledge base under the respective context (Michael 2019). This resulting inference (advice or explanation) could be evaluated by the user or assistance requester. The Inference Generator filters the output of the Reasoning Engine which could be sent to the machine, which in our case is the robotic arm.

4.1.1.2 Environment

The environment is composed of an industrial robotic arm which has an independent CPU (Central Processing Unit) that hosts the execution of the sequential program that drives the arm to perform a task of picking a part from a point A, placing it at a point B, and then going to point C, which is the home position of the arm.

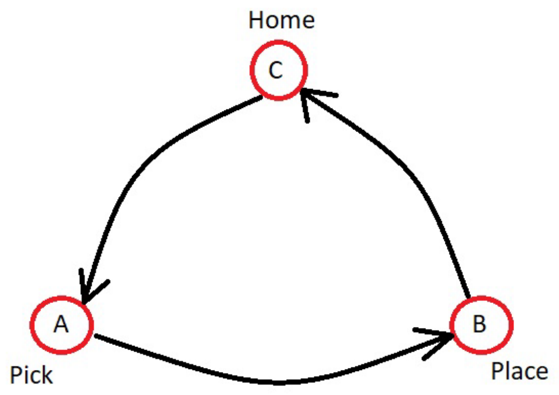


Figure 4. Arm's Motion Task

The signal determining the availability of the part is received by the arm from its environment via an optic or vision sensor. The internal status or knowledge block cyclically updates the arm's status information such as the arm's power status, arm's motion status, arm's gripping status, alarm status, etc. Besides containing the application specific status information, this block also holds some dynamic knowledge in the form of the current day of the week, current hour etc. Based on the environmental and status information, the context generator generates several contexts which are cyclically sent to the argumentation framework. The inferences generated by the argumentation framework are sent to the Inference Interpreter, which interprets the received inferences into the robot's recognizable instructions. Based on these inferences, the robot performs its Pick-Place routine, which affects its immediate environment.

The user or assistance requester could thereby coach and control the robotic arm by modifying the existing rules or appending new rules in the knowledge base of the argumentation framework (Michael 2019).

4.2 The Language of Staubli Robotic Arm – VAL3

Variable Assembly Language (VAL) is a computer-based control system and language designed specifically for programming Unimation Industrial Robots. The instruction sets used in VAL are easy to understand and self-explanatory in nature, which makes the syntax easy to read and interpret (Shimano 1979). VAL3 finds its roots in VAL and is a high-level programming language designed specifically to program and drive Staubli robots. It facilitates the combination of the basic features of a standard real-time high-level computer language with the specific functionalities required to control an industrial robot cell, such as tools for robot control, tools for geometrical modeling, tools for input/output, etc. (Akdogan 2019).

4.2.1 VAL3 Application

A VAL3 application is a complete software package that includes programs, global and local data, libraries, user data types, HMI user pages, and a multi-tasking option for concurrent program execution. A new application upon creation is generated with a start*()* program which is called when the application is started and a stop*()* program which is called when the application is stopped (Akdogan 2019).

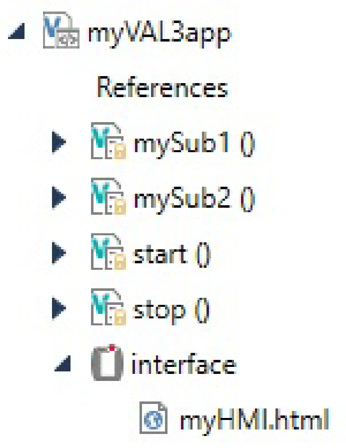


Figure 5. Sample VAL3 Application

4.3 Communication over Socket TCP IP

One of the prominent protocols of the Internet protocol suite is the Transmission Control Protocol (TCP). The data packets or streams of octets (bytes) delivered by TCP between applications running on hosts over an IP network are reliable, ordered, and error-checked (Kurose & Ross 2013).

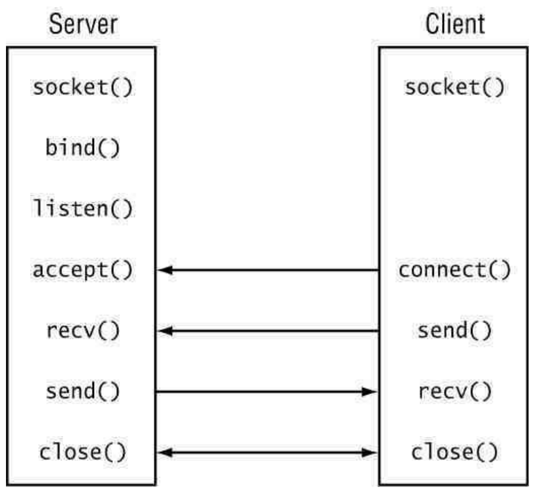


Figure 6. Socket Communication

The use of TCP for communication is so prevalent that we can easily get examples from our daily interaction with computers and the internet. Our emails, the World Wide Web, etc. are some of the most common examples that rely on TCP. Another protocol that resembles TCP but lacks reliability is the User Datagram Protocol (UDP), which does not perform error checking on the data stream and emphasizes reduced latency. Online gaming, video streaming, etc. rely on UDP.

4.3.1 Client & Server in TCP IP

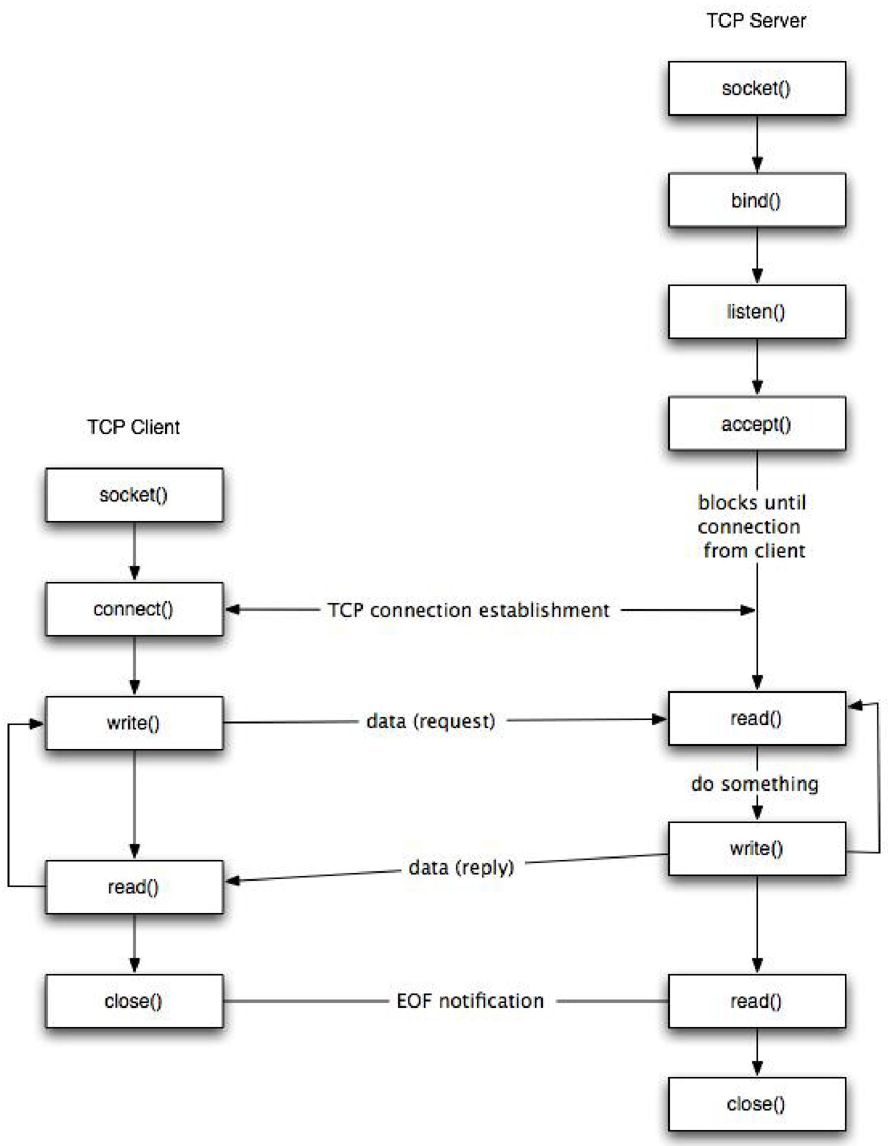


Figure 7. Client-Server Connection

The working principle of TCP/IP connections resembles that of a telephone call where a connection is required to be initiated by a person by dialing the phone number. Once connection is established, the person at the receiver's side must be listening to the call and pick up the line. In TCP/IP, the IP address is like the phone number, and the port number is like the extension code. In the TCP/IP connection, the device that dials the phone number is the Client, and the device that listens to the incoming calls is the Server. Therefore, in a TCP/IP connection, the IP address and the port number of the server are required to be known by the client.

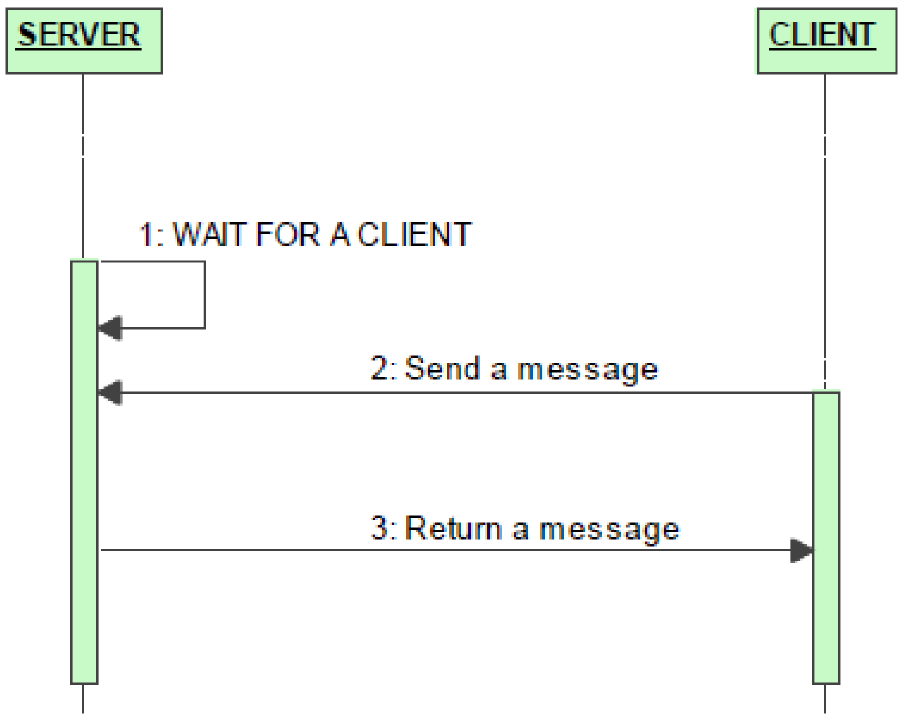


Figure 8. Communication Flow

4.4 The PRUDENS Tool

PRUDENS is a Java application that has been developed by the research group at the Computational Cognition Lab of the Open University of Cyprus led by Dr. Loizos Michael. When it is given a knowledge base about a certain task and context-encoding information about a specific scenario pertaining to that task, it gives the requestor some piece of advice along with an explanation as anticipated by the appropriate machine coaching theory.

4.4.1 The Language of PRUDENS Tool

As PRUDENS permits predicates to be used to represent relationships between universal entities, we can say that it resembles Prolog to some extent in that sense. Each predicate is made up of two parts, such as its name and a list of arguments. Anything starting with lower case Latin alphabets (a to z) and following a limited sequence of letters, digits, or underscore might be used as the predicate name. The arguments are in the form of a list, with each argument separated by a comma and enclosed by left and right parenthesis. For example, *predicateName*($X_1, X_2, X_3, \ldots, X_n$)

where, $X_1, X_2, X_3, \ldots, X_n$ are the arguments.

In the above example, the length of the arguments appearing in the argument list is *n.* Therefore, we may say that the predicate's *arity* is *n.*

The arguments are bound to have the following properties:

- Arguments could be constants that appear as strings of lower-case letters (a-z) which might follow a limited sequence of letters, digits, and underscores. Constants are distinct universal entities. E.g., *animal(tiger),* where *tiger* is a distinct universal entity.

- Arguments could be variables that appear as strings of upper-case letters (A-Z) which might follow a limited sequence of letters, digits, and underscores. The variables are used as placeholders for the constants. E.g., *herbivorous(X),* where *X* is the placeholder for a constant.

It is possible for a predicate to contain both constants and variables as the elements of the argument list. E.g. *marriedTo(X, sam)* which is interpreted as *''X is married to some universal entity Sam''*. Besides the standard user defined predicates, Prudens also provide built-in predicates for equality and inequality which are represented as *?=(something, something)* and *?<(something, something)*. In some cases, these predicates can also be used as a condition to allow a baseless variable to be united with a constant. Simply, *?=(X,Y)* is read as *''X is equal to Y''* and *?<(X,Y)* is read as *"X is less than Y".*

The rules play a key role in the language of Prudens as through them one can capture desirable behaviors or establish new relationships between the elements of the universe to create a knowledge base. The rules are built up of predicates, variables and constants or simply literals. A literal could be a predicate itself or its negation which is represented with a minus (-) sign which follows no space between the minus sign and the name of the predicate. For instance, *herbivorous(X), -herbivorous(X)* or *marriedTo(X, sam), -marriedTo(X, sam)*. In Prudens, the negation is regarded as classical negation due to the absence of the Closed World Assumption theory where it is possible to draw inferences under a similar context about something which is not declared in the knowledge base. For instance, in Closed World Assumption (CWA) it is possible to infer *-brotherOf(sam, tom)* from *brotherOf(sam, rose)* and *brotherOf(jack, sam)* as *brotherOf(sam, tom)* does not exist in the knowledge base. However, in Prudens, it is not possible to infer the above due to the absence of any CWA.

In PRUDENS, rules follow a specific structure which includes the rule's name, rule's body and rule's head where either of the three should not be null or empty. The rule's name is separated from the body and the head through a combination of two colons (::). The rule's head is referred to as a single literal whereas the rule's body is referred to as the literal list separated by commas. The head and the body together form the main part of the rule. An entire rule ends with a semicolon (;). For instance, *Rule_1 :: eatsGrass(X), animal(X) implies herbivorous(X);*. An ordered list of these rules together forms the knowledge base. The order of appearance of the rules in the list adds a priority relation to them with the rule appearing first in the list having the highest priority i.e. it is liable to be triggered first in comparison to the ones appearing below.

In Prudens language, there are two ways by which knowledge could be encoded namely the *knowledge base* and the *contexts*. Contexts are the pairwise non-conflicting literals that contain only constants as their argument and are separated from each other using semicolon (;). They act as solid facts that describe a certain situation whereas knowledge bases are built up of rules that depict a certain pattern of behavior in a variety of settings through a priority order which are provoked by the facts that appear as context. For example,

KB1
Rule_3 :: holiday(X), tired(X) implies perform(takeRest);
Rule_2 :: holiday(X), tired(X) implies -perform(longDrive);
Rule_1 :: holiday(X) implies perform(longDrive);

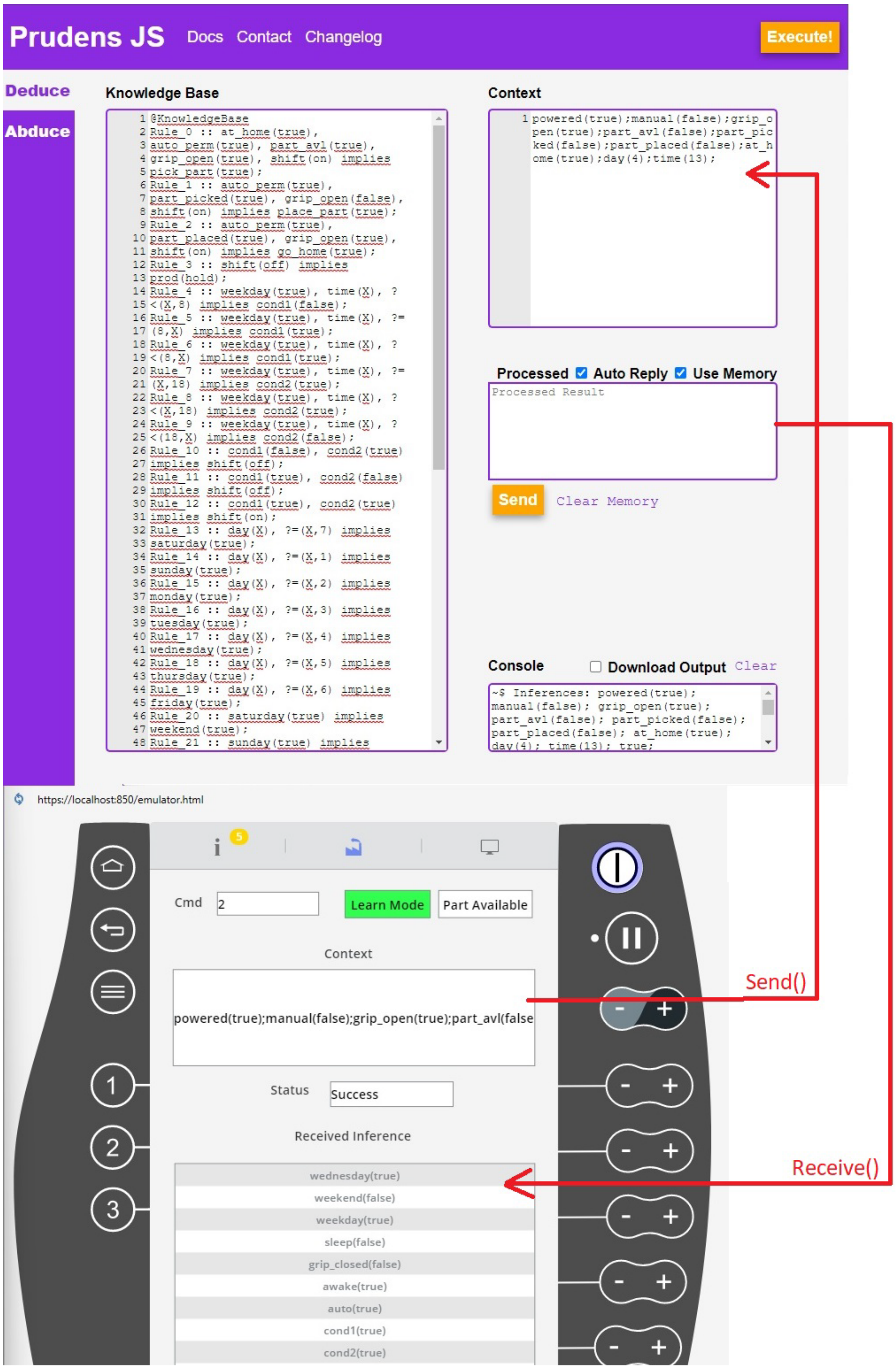
From the rules above, it is evident that a policy about what is to be performed on a holiday is encoded in the knowledge base. Rule_1 depicts that on a holiday we must go on a long drive in a general manner. Rule_2 depicts an exception to Rule_1 that if we are tired on a holiday we must not go on a long drive. Rule_3 depicts that if we are tired on a holiday then we must take rest. Hence, the knowledge base encodes that on a holiday we must go on a long drive and on a holiday if we are tired then we must take rest. These two scenarios could be encoded through two separate contexts namely S1 *holiday(today);* which depicts that today is a holiday and we are not tired and S2 *holiday(today); tired(today);* which depicts that today it's a holiday and we are tired. Hence, for the context S2 from KB1 we can say that Rule_3 and Rule_2 are triggered and as the priority of Rule_3 is higher than Rule_2 therefore Rule_2 is ignored for the inference.

If our knowledge base is somewhat similar to the one below as KB2, the outcomes achieved would have differed as compared to the above.

KB2
Rule_1 :: holiday(X) implies perform(longDrive);
Rule_3 :: holiday(X), tired(X) implies perform(takeRest);
Rule_2 :: holiday(X), tired(X) implies -perform(longDrive);

Under both the contexts S1 and S2, as the priority of Rule_1 is the highest as compared to Rule_2 and Rule_3, the inference generated is based on Rule_1 (Markos nd).

4.5 Linking PRUDENS and the Robotic Arm



Knowledge Base

```
@KnowledgeBase
Rule_0 :: at_home(true),
auto_perm(true), part_avl(true),
grip_open(true), shift(on) implies
pick_part(true);
Rule_1 :: auto_perm(true),
part_picked(true), grip_open(false),
shift(on) implies place_part(true);
Rule_2 :: auto_perm(true),
part_placed(true), grip_open(true),
shift(on) implies go_home(true);
Rule_3 :: shift(off) implies
prod(hold);
Rule_4 :: weekday(true), time(X), ?
<(X,8) implies cond1(false);
Rule_5 :: weekday(true), time(X), ?=
(8,X) implies cond1(true);
Rule_6 :: weekday(true), time(X), ?
<(8,X) implies cond1(true);
Rule_7 :: weekday(true), time(X), ?=
(X,18) implies cond2(true);
Rule_8 :: weekday(true), time(X), ?
<(X,18) implies cond2(true);
Rule_9 :: weekday(true), time(X), ?
<(18,X) implies cond2(false);
Rule_10 :: cond1(false), cond2(true)
implies shift(off);
Rule_11 :: cond1(true), cond2(false)
implies shift(off);
Rule_12 :: cond1(true), cond2(true)
implies shift(on);
Rule_13 :: day(X), ?=(X,7) implies
saturday(true);
Rule_14 :: day(X), ?=(X,1) implies
sunday(true);
Rule_15 :: day(X), ?=(X,2) implies
monday(true);
Rule_16 :: day(X), ?=(X,3) implies
tuesday(true);
Rule_17 :: day(X), ?=(X,4) implies
wednesday(true);
Rule_18 :: day(X), ?=(X,5) implies
thursday(true);
Rule_19 :: day(X), ?=(X,6) implies
friday(true);
Rule_20 :: saturday(true) implies
weekend(true);
Rule_21 :: sunday(true) implies
```

Context

```
powered(true);manual(false);grip_o
pen(true);part_avl(false);part_pic
ked(false);part_placed(false);at_h
ome(true);day(4);time(13);
```

Processed ☑ Auto Reply ☑ Use Memory

Processed Result

Console ☐ Download Output

```
~$ Inferences: powered(true);
manual(false); grip_open(true);
part_avl(false); part_picked(false);
part_placed(false); at_home(true);
day(4); time(13); true;
```

Cmd: 2

Context: powered(true);manual(false);grip_open(true);part_avl(false

Status: Success

Received Inference

wednesday(true)
weekend(false)
weekday(true)
sleep(false)
grip_closed(false)
awake(true)
auto(true)
cond1(true)
cond2(true)

Figure 9. Robotic Arm Coaching Interface

The TCP/IP socket communication protocol is used in order to establish the link between the PRUDENS software interface and the robotic arm. The PRUDENS software interface acts as a server that sends responses to the client's requests, which is the Robotic Arm. To achieve this, a JavaScript application *index.js* is developed on the PRUDENS side. This application opens one particular port on the server, and the robotic arm communicates with this server's IP at the specified port. The robotic arm will send requests (context) to the server in a predefined format. The incoming request is then processed to generate the necessary inferences through the argumentation functionality, and the output is then parsed to generate a message in a format suitable for the robotic arm to understand. If Automatic Reply functionality is not used on the front end, the parsed response could be seen populated in the Processed Result column. The Automatic Reply functionality enables the server to send the processed response automatically to the client without the need for human intervention. By selecting the Use Memory option at the front end, the server starts operating in cache mode where it generates a cache memory and stores the responses against the incoming context such that when the same context arrives at the server next time, its inference is directly sent to the client from the cache memory without waiting for the deductive reasoning process. This enables the server to generate quick responses at a low processing time.

For the backend services, Node.js is used as a scripting language, and for the front end, the prudens-js project is used from the github repository to add the Auto Reply and Use Memory checkboxes as well as the Clear Memory button. In order to start the application, we need to navigate to the root directory and start the application using the command NPM START in the command prompt window. This will start the application on the localhost:3000 url and, by default, it will also open 1258 as a telnet port for client-server communication. In order to terminate the application, ctrl+C keyboard input could be used in the command prompt window.

4.5.1 Code Description - PRUDENS

Index.js: This routine is used to implement socket communication.

```
const app = require('express')();
const http = require('http').Server(app);
const io = require('socket.io')(http);
const net = require("net");
```

To achieve this functionality, at the PRUDENS side we used the *require()* function. As part of the basic functionality of this function, reading of a JavaScript file, executing it, and returning the output as an exports object is achieved.

The above code displays the basic module imports that are required to implement socket connection. Here by the use of *express()* function, an Express application is created denoted by *app.* The *app* returned by *express()* is originally a JavaScript function that is designed to be passed to the Node's HTTP servers as a callback to handle requests. The import of the *http* module enables Node.js for data transfer over Hyper Text Transfer Protocol (HTTP). The *socket.io* library import allows real time, two-way and event driven communication between the client and the server. The net module supplies an asynchronous or half-duplex network binding such that the data flows only in one direction at a time.

```
const port = process.env.PORT || 3000;
```

Here we create a default port 3000. It tells the application to use port 3000 unless there exists a preconfigured port in the environment.

```
var express = require('express');
var cache = new Object();
var lastReq;
```

Here we create an express application and a cache object which creates a cache memory functionality to map incoming requests to their responses.

```
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});
```

Here we map the index.html as our default html page.

```
app.use(express.static(__dirname + '/'));
```

Here we map the static resources like CSS using the express.static(root, [options]) function. This function enables us to use static files such as images, css files, Javascript files etc. The root argument specifies the root directory where the static files are available and options argument which in our case is the "/". When the path name is a directory, this argument redirects to the trailing "/".

```
var globalSocket;
var useMemory;
const server = net.createServer((socket)=>{
  globalSocket = socket;
 globalSocket.write("Hello From Server!")
  var input = "";
```

Here we create a global socket server and send a greeting message to the client. In order to know that a communication has been established with the client, the server open the port and starts listening to the request and when a ping request from the client is received, it sends back the greeting response once at initialization of the communication phase.

```
socket.on("data",(data)=>{
     input +=data.toString()
```

Post initialization of the handshake between the server and the client this function is called on each receipt of the data from the client i.e. the robotic arm. One we receive a data packet from the client we iterate it inside a for loop until we find the end character which is 13 in ASCII format. As soon as the end character is found, the received request or rather context is sent for processing. Here we also check the status of the Memory function. If it is enabled and the processed response is already available in the cache memory then the response is sent to the client directly without processing.

```
for (let i = 0; i < input.length; i++) {
    if(data.toString().charCodeAt(i)== 13){
     lastReq = input;
   if(useMemory && cache[lastReq]){
    globalSocket.write(cache[lastReq])
   }else{
    io.emit('context', input);
   }
   input =""
   }
  };
```

In order to close the socket the below function is called.

```
socket.on("close",()=>{
   console.log("Connection closed.!!!")
 })
});
server.listen(1258);
```

Here we determine the port number on which we want the server to listen. Once the connection is established, the below function is called.

```
io.on('connection', (socket) => {
```

This below function will be called on receiving response from server which includes the processed result.

```
 socket.on('ServerResponse', msg => {
```

Here in order to make the response interpretable by the client or the robotic arm, extra curly braces are appended in the response.

```
    msg="{"+msg+"}"
```

In order to Send the response back to robot, we use the socket.write() command.

```
    globalSocket.write(msg)
```

Here we check that if the memory function is enabled then we cache the incoming request for future use.

```
    if(useMemory)
      cache[lastReq]=msg;
  });
```

This function is called on the clear memory event and will clear the stored responses in the memory.

```
  socket.on('clearMemory', msg => {
    cache = new Object();
  });
```

This function will activate or deactivate the memory function based on front end checkbox selection.

```
  socket.on('useMemory', msg => {
    console.log(msg)
    useMemory = msg;
  });
});
```

This is the main function to start the application on the given port.

```
http.listen(port, () => {
  console.log(`Socket.IO server running at http://localhost:${port}/`);
});
```

Index.html: The front end interface of the PRUDENS application has been designed using the index.html code. In order to catch several events from the objects that are added on the front end for their execution by the backend program, several commands are added in the original index.html code.

```
<script src="/socket.io/socket.io.js"></script>
```

In the html head, the *socket.io.js* script is added to load the socket.io client which enables the front end to communicate with the backend for data exchange over Socket.IO.

```
<script>
```

Socket is established at client side (html page) to send data to backend server.

```
var socket = io();
```

Once data is received the context will be set to the processed column on the html page.

```
socket.on('context', function(msg) {
document.getElementById('deduce-tab context').value=msg.toString().trim();
consoleOutput();
```

Here we execute the auto send function to inform the server to automatically send the processed response back to robot without manual intervention.

```
if( document.getElementById('autoSend-checkbox').checked)
sendResponse()
});
```

This below is the function that sends response back to server on the serverResponse event.

```
function sendResponse(){
console.log("Emitting:"+document.getElementById("processedResult").value )
socket.emit('ServerResponse',  document.getElementById("processedResult").value);
document.getElementById("processedResult").value='';
}
```

This function is used to clear the server's memory using the clear memory event.

```
function clearMemory(){
socket.emit('clearMemory', '');
}
```

This is the function that enables or disables the use of memory function of the server using the useMemory event.

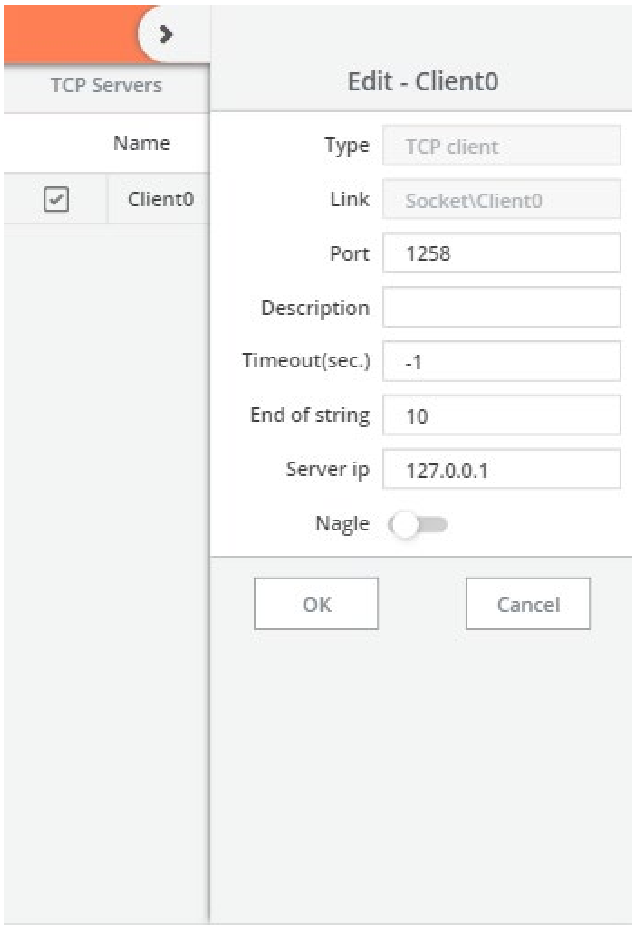
```
function useMemory(){
socket.emit('useMemory', document.getElementById('memory-checkbox').checked);
 }
</script>
```

Utils.js: The parsing of the processed result prior sending to the client is performed in this program. This following block of code is for filtering the processed output until *"; true;"* so that it results only the inferences deduced after processing.

```
var filter ="; true; ";
try{
	let processed =  contextToString(inferences);
	processed =
processed.substring(processed.indexOf(filter)+filter.length,processed.length-1)
	document.getElementById("processedResult").value = processed;
}catch(err){
	console.error(err);
```

}
 return outputString + "Inferences: " + contextToString(inferences) + "\nGraph: " +
graphToString(graph);

4.5.2 Code Description – ROBOTIC ARM



Type	TCP client
Link	Socket\Client0
Port	1258
Description	
Timeout(sec.)	-1
End of string	10
Server ip	127.0.0.1
Nagle	

Figure 10. Manual Socket Implementation

The VAL3 language enables the user to define sockets which could either be Server or Client manually from the HMI or dynamically from within the code.

A Client socket is created at the Robotic Arm's side manually from the HMI as shown in Figure 9 A client socket at the Staubli Robotic Arm requires four typical parameters which are the Port Number, Timeout in seconds, End of String character and the Server IP address.

The port number at both the server and the client side should be identical. The Server IP should contain the IP address of the system on which the server application is running. Once the client socket is implemented from the HMI, we create two programs namely *sockets* and *test01* in the robot's controller. The sockets program acts as a library to the test01 program and consists of a variable named *siocam* of data type SIO (socket input output). This variable is linked to the physical client socket which we implemented manually from the HMI in order to be used dynamically inside the program.

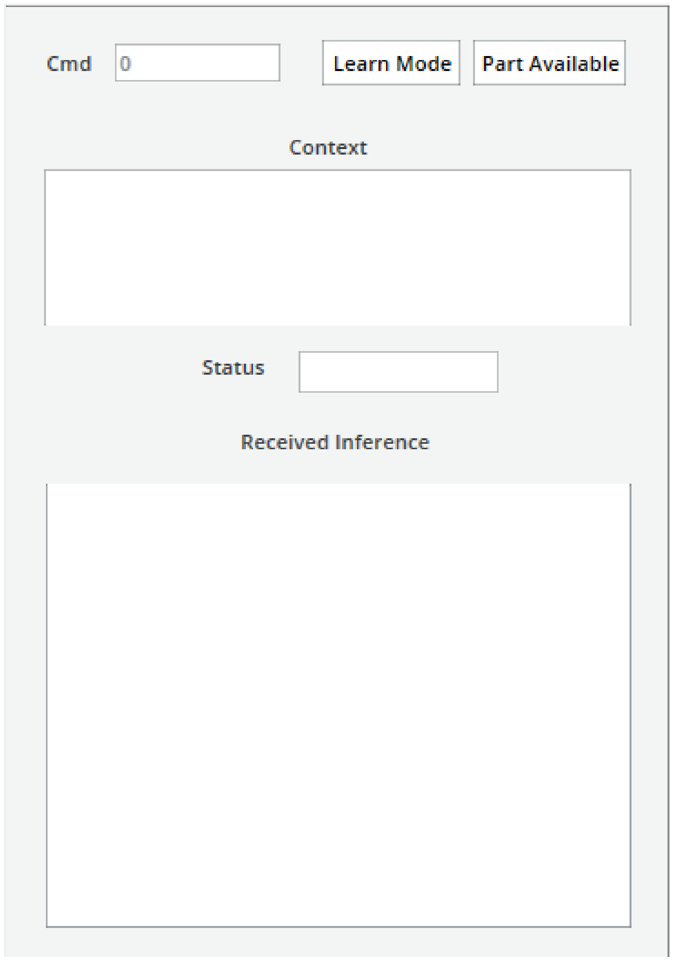


Figure 11. Robot's HMI Interface

A user interface in the form of an HMI page is created which enables the user to start the communication between the robotic arm and the PRUDENS server, a text field that displays the context message generated by the code to be sent to the server, a text field that displays the status of the send request to server function, a list box that displays the received inference from the server, a learn mode activation button which increases the cyclic delay of the send-receive process to compensate the processing delay at the server side, and a part available button that simulates the availability of a part that is required to be grabbed by the robotic arm.

On execution of the *test01* application, six parallel tasks are created as follows.

- **taskCreate** "hmi",10,hmi()

This task catches the button press events on the HMI page and manages their color on activation and deactivation.

- **taskCreate** "calender",10,getCalender()

This task returns the day of the week based on the current date as a number between 1 and 7 where 1 represents Monday.

- **taskCreate** "time",10,getTime(sHour)

This task returns the current hour in 24Hr format.

- **taskCreate** "stat",10,getStat()

This task cyclically gets the arm's status information and concatenates them as a string of contexts to be sent to the PRUDENS server. These contexts include the arm power status (powered/-powered), the mode of operation (manual/-manual), end effector tooling status (open/-open), part availability (available/-available), part picked status (picked/-picked), part placed status (placed/-placed), home position status (at home/-at home), current day of the week and current hour.

- **taskCreate** "refList",10,refBinding()

This task cyclically refreshes the list box on the HMI to display the latest inferences received from the server as contents of the list.

- **taskCreate** "prod",10,production()

This task consists of three motion commands namely Pick Part, Place Part and Go Home which are executed on parsing the received inferences from the PRUDENS server.

Post creation of the above parallel tasks, the program enters an endless loop where it waits for the user to switch on the Learning Mode and provide a numeric start command to initiate the communication with the PRUDENS server.

```
while true
  //
  wait(nCmd==1)
  //
  while !bLearning
    popUpMsg("Switch on Learning Mode")
    delay(0.1)
  endWhile
  //
```

When the user enters 1 at the numeric command, the initialize communication procedure starts where the robot's application sends a message "*Client Calling*" to the server. On successful delivery of the message to the server, "*Pinged Server*" message is printed in the status field of the robot's HMI.

```
call sendRequestInit(0,"Client Calling",bErr)
  if !bErr
    sStatus="Pinged Server"
    call getResponse(0,l_sResp)
    sRequest=l_sResp
    if sRequest!=""
      call resetSocket(0,-1,bErr)
      if !bErr
        nCmd=2
      delay(0.5)
      sRequest=""
     else
      nCmd=0
      sStatus="Reinitialize Communication!"
     endIf
    endIf
   else
    sStatus="Ping Failed"
    call resetSocket(0,1,bErr)
    if bErr
     nCmd=0
     sStatus="Reinitialize Communication!"
    endIf
   endIf
```

The server sends the greeting message subsequently, which on reception at the robot's side the program enters the main cyclic loop where the processed contexts are sent to the PRUDENS server, and the respective received messages are parsed and filtered to be populated as elements of a list of inferences which are filtered for the motion commands by the "*Production*" task running in parallel.

```
while nCmd==2
   //
   sSendMsg=sSendData
   call sendRequest(0,sSendMsg,bErr)
   if !bErr
    sStatus="Success"
    //
    call getResponse1(0,l_sResp)
    //
    call parsing(nLengthServ,nStreamServ,sData,false,bErr)
    if !bErr
```

```
        call filterData(sData,sFilter)
        call getInference(sFilter,sInf,bErr,sRequest)
        //
       endIf
      else
       sStatus="Failed"
       call resetSocket(0,1,bErr)
       if !bErr
        nCmd=1
       else
        nCmd=0
        sStatus="Reinitialize Communication!"
       endIf
      endIf

      delay(nDelay)
     endWhile
     delay(0)
    endWhile
```

Data Parsing: The received data from the PRUDENS server is parsed using the *parsing()* function. This function looks for the initial character, the separator, and the end character for data parsing. The characters starting after the initial character up to the separator are combined to form a message chunk and stored in a data array. This process is repeated until the appearance of the end character.

Data Filtration: If the parsed message chunks include white spaces, they are required to be filtered through the *filterData()* function. This function checks each element of the data array to filter the white spaces if present and stores the filtered message chunks into the filtered data array.

Inference acquisition: The inferences are acquired from the filtered data array using the *getInference()* function. This function outputs the inferences as text keys of a collection data type. These keys are checked for the motion commands in the “Production” task running in parallel. For example, if the inference collection has a key named *pick_part(true)* then the robotic arm moves to the pick position to grab the part. Similarly, if the key named *place_part(true)* is available then the robotic arm moves to the place point in order to release the part and if the key named *go_home(true)* is available then the robotic arm moves to the home position.

Error Handling: The error handing in the communication is handled in a very sophisticated manner through reporting the user with the status information on the HMI. This enables the user to understand about the communication status that whether it is successful in each cycle or failed.

In case of communication failure at any instance, the program automatically goes into communication initialization by using the *resetSocket()* function. With the help of this function, the *timeout* parameter of the socket is modified dynamically to 1 second which is otherwise –1. This enables the socket to wait 1 second at each send and receive instance. In case the *resetSocket()* function returns error, the program informs the user to *reinitialize the communication* manually. Upon successful automatic restoration of the client-server communication the timeout parameter of the socket is restored to -1 which means that the send and receive cycles process whatever message is available at the port without any delay.

Chapter 5
Evaluation

In order to evaluate our proposed system in the previous chapter we proceed with two real time experiments where the robotic arm is tested in two separate production environments by ten volunteers belonging to distinct industrial domains and expertise. Fifty percent of the volunteers participated in experiment 1 and the remaining participated in experiment 2. The volunteers remained engaged in the actual coaching of the robotic arm for ten minutes each where they controlled the robotic arm through PRUDENS interface and tried to improve the robot's performance through various policies shown below. However, we discuss in the text the most suitable policies that attained maximum performance in terms of the robot's operational tasks under various constraints. In these experiments we monitor the system's performance based on intrinsic factors. We also evaluate the system performance extrinsically based on the collective feedback of the ten volunteers.

Experiment 1: Part Handling between two points

In this experiment the robot's task is to Pick a part from the pick point and place it to the place point. After placing the part the arm has to return to the home position and loop in a similar fashion. All the three Pick, Place and Home positions are static locations which have already been taught manually to the robotic arm. The robotic arm is interfaced with the PRUDENS server which tells it to pick, place or go to home depending upon the deductive reasoning utilizing the rules in the knowledge base under the incoming contexts as status information from the robotic arm.

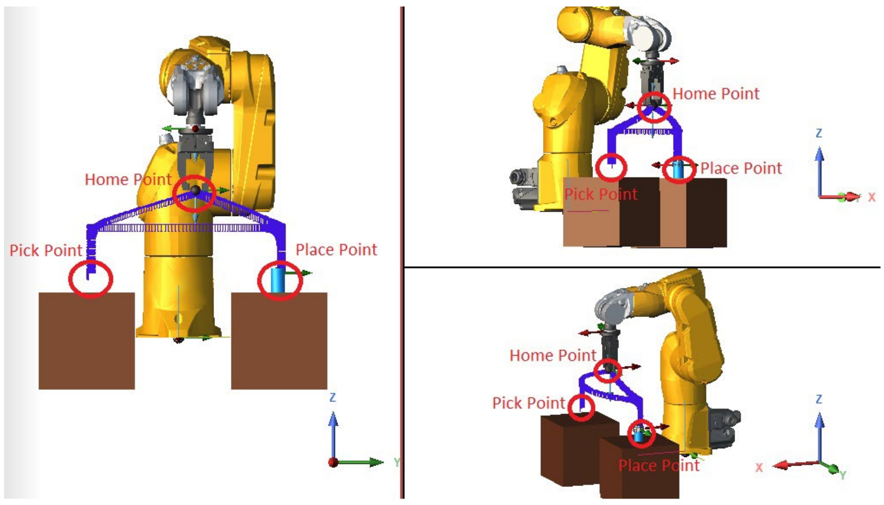


Figure 12. Arm's Pick-Place Cycle

In the initial phase the robot follows a standard policy with the following rules in the knowledge base of the PRUDENS server that drive the robot based on the context.

Rule_0 :: auto_perm, part_avl, grip_open implies pick_part;
Rule_1 :: auto_perm, part_picked implies place_part;
Rule_2 :: auto_perm, part_placed implies go_home;
Rule_3 :: auto_perm, -part_avl, grip_open implies wait_part;
Rule_4 :: -sleep, auto implies auto_perm;
Rule_5 :: -powered implies sleep;
Rule_6 :: powered implies -sleep;
Rule_7 :: manual implies -auto;
Rule_8 :: -manual implies auto;

At an instant the context received from the robotic arm is as under.

powered; -manual; grip_open; -part_avl; -part_picked; -part_placed; at_home; day(5); time(2);

The inferences received by the robotic arm are *-sleep; auto; auto_perm; wait_part.*

As human could be directly involved with the robotic arm in collaborative working scenarios or as the production is directly affected by this arm under the non-collaborative scenarios it becomes necessary for this system to explain each of its unexpected move.

The explainability method of knowledge extraction is a close fit for such a system to increase its interpretability (Adadi & Berrada 2018). As the building block of this system is argumentation theory through hypothesis generation based on the rules present in its knowledge base as well as the rules learnt and generated through counter argumentation, this system explains about what it is supposed to do when asked to perform a new task or modify its current task or learn new challenges.

It becomes quite a challenge for the machine operators or blue-collar workers to establish communication with complex machines such as the robotic arms. Our proposed system enhances the ability of the non-technical staff in a production facility to interact with the machine through argumentation. In the above case when the operator seeks guidance to understand the cause of robot's stalling, the console section on the PRUDENS interface describes its inferences in the following manner.

-sleep: [Rule_6 :: powered implies -sleep;]

Rule 6 suggests that when the robot is powered on it is not in sleep mode.

auto: [Rule_8 :: -manual implies auto;]

Rule 8 suggests that when the robot is not in manual mode, it is in automatic mode.

auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]

Rule 4 suggests that when the robot is not in sleep mode and automatic mode is available then robot is ready to work in automatic cycle.

wait_part: [Rule_3 :: auto_perm, -part_avl, grip_open implies wait_part;]

Rule 3 suggests that when part is not available and other conditions are true then it is waiting for part.

This functionality of self-explanation makes the Human Machine Interaction easy to everyone. The operator thus understands that he needs to make the parts available so that the robot could proceed with its regular operation. In this first case, the operator creates an exception in the form of his counter argument to coach the robot by modifying *Rule 0* and deleting *Rule 3.*

The knowledge base now looks as below.

Rule_0 :: auto_perm, grip_open implies pick_part;
Rule_1 :: auto_perm, part_picked implies place_part;
Rule_2 :: auto_perm, part_placed implies go_home;
Rule_4 :: -sleep, auto implies auto_perm;
Rule_5 :: -powered implies sleep;
Rule_6 :: powered implies -sleep;
Rule_7 :: manual implies -auto;
Rule_8 :: -manual implies auto;

Now under the similar context as above the inferences generated are *-sleep; auto; auto_perm; pick_part.*

The description of the generated inferences is as under.

-sleep: [Rule_6 :: powered implies -sleep;]
auto: [Rule_8 :: -manual implies auto;]
auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]
pick_part: [Rule_0 :: auto_perm, grip_open implies pick_part;]

Here the robot moves to pick the part because the conditions of *Rule 0* are satisfied and iterates in a loop to place the part and moving to the home position subsequently.

In the second case, the operator seeks guidance to understand why the robot works non-stop without caring about the weekends. As evident from the above robot's standard policy and the relevant explanations which does not include the time constraint even though the context contains it, the operator understands the need to add exceptions in order to coach the robotic arm such that it should only work during the weekdays. The knowledge base is then modified with the following exceptions.

Rule_0 now contains an additional predicate named *at_home* as the operator desires that the robot should only go to pick the part once it arrives to the home position.

Rule_0 :: at_home, auto_perm, part_avl, grip_open, weekday implies pick_part;
Rule_1 :: auto_perm, part_picked, -grip_open, weekday implies place_part;
Rule_2 :: auto_perm, part_placed, grip_open, weekday implies go_home;
Rule_3 :: day(X), ?=(X,1) implies monday;
Rule_4 :: day(X), ?=(X,2) implies tuesday);
Rule_5 :: day(X), ?=(X,3) implies wednesday;
Rule_6 :: day(X), ?=(X,4) implies thursday;
Rule_7 :: day(X), ?=(X,5) implies friday;
Rule_8 :: day(X), ?=(X,6) implies saturday;
Rule_9 :: day(X), ?=(X,7) implies sunday;
Rule_10 :: saturday implies weekend;
Rule_11 :: sunday implies weekend;
Rule_12 :: monday implies -weekend;
Rule_13 :: tuesday implies -weekend;
Rule_14 :: wednesday implies -weekend;
Rule_15 :: thursday implies -weekend;
Rule_16 :: friday implies -weekend;
Rule_17 :: -weekend implies weekday;
Rule_18 :: weekend implies -weekday;
Rule_19 :: -sleep, auto implies auto_perm;
Rule_20 :: -powered implies sleep;
Rule_21 :: powered implies -sleep;
Rule_22 :: manual implies -auto;
Rule_23 :: -manual implies auto;

The operator therefore adds exceptional rules (Rule_3 to Rule 18) apart from the standard policy rules to incorporate the feature of weekend off for the robot.

In the third case, the operator inquires about the status of the Gripper and the permission to manually jog the robot. As the knowledge base does not contain any fact or exception regarding the operator's query therefore the operator defines further facts and exceptions regarding the same. Now the knowledge base looks as below.

Rule_0 :: at_home, auto_perm, part_avl, grip_open, weekday implies pick_part;
Rule_1 :: auto_perm, part_picked, -grip_open, weekday implies place_part;
Rule_2 :: auto_perm, part_placed, grip_open, weekday implies go_home;
Rule_3 :: day(X), ?=(X,1) implies monday;
Rule_4 :: day(X), ?=(X,2) implies tuesday);
Rule_5 :: day(X), ?=(X,3) implies wednesday;
Rule_6 :: day(X), ?=(X,4) implies thursday;
Rule_7 :: day(X), ?=(X,5) implies friday;
Rule_8 :: day(X), ?=(X,6) implies saturday;
Rule_9 :: day(X), ?=(X,7) implies sunday;
Rule_10 :: saturday implies weekend;
Rule_11 :: sunday implies weekend;
Rule_12 :: monday implies -weekend;
Rule_13 :: tuesday implies -weekend;
Rule_14 :: wednesday implies -weekend;
Rule_15 :: thursday implies -weekend;
Rule_16 :: friday implies -weekend;
Rule_17 :: -weekend implies weekday;
Rule_18 :: weekend implies -weekday;
Rule_19 :: awake, auto implies auto_perm;
Rule_20 :: -sleep, manual implies jog_perm;
Rule_21 :: -powered implies sleep;
Rule_22 :: powered implies -sleep;
Rule_23 :: grip_open implies -grip_closed;
Rule_24 :: -grip_open implies grip_closed;

Rule_25 :: jog_perm implies -auto_perm;

Rule_26 :: sleep implies -awake;

Rule_27 :: -sleep implies awake;

Rule_28 :: manual implies -auto;

Rule_29 :: -manual implies auto;

Under the context which says that, *powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(8);*

The PRUDENS server now generates the inferences as *monday; -weekend; weekday; -sleep; -grip_closed; awake; auto; auto_perm; pick_part* which are explained in the following way.

monday: [Rule_3 :: day(1), ?=(1, 1) implies monday;]

Rule 3 suggests that it is Monday.

-weekend: [Rule_12 :: monday implies -weekend;]

Rule 12 suggests that it is not weekend.

weekday: [Rule_17 :: -weekend implies weekday;]

Rule 17 suggests that it is a weekday.

-sleep: [Rule_22 :: powered implies -sleep;]

Rule 22 suggests that when the robot is powered on it is not in sleep mode.

-grip_closed: [Rule_23 :: grip_open implies -grip_closed;]

Rule 23 suggests that gripper is not closed.

awake: [Rule_27 :: -sleep implies awake;]

Rule 27 suggests that as the robot is not in sleep mode, it is awake.

auto: [Rule_29 :: -manual implies auto;]

Rule 29 suggests that the robot is in automatic mode.

auto_perm: [Rule_19 :: awake, auto implies auto_perm;]

Rule 19 suggests that as the robot is awake and is in automatic mode that means it has permission to work automatically.

pick_part: [Rule_0 :: at_home, auto_perm, part_avl, grip_open, weekday implies pick_part;]

Rule 0 suggests that the robot must pick the part as all of its conditions are satisfied.

If the day is a weekend for example, then the context becomes *powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(6); time(2);*

Then based on the above rules in the knowledge base the following inferences are generated, *saturday; weekend; -weekday; -sleep; -grip_closed; awake; auto; auto_perm;* whose explanations are inferred from the following rules.

saturday: [Rule_8 :: day(6), ?=(6, 6) implies saturday;]
weekend: [Rule_10 :: saturday implies weekend;]
-weekday: [Rule_18 :: weekend implies -weekday;]
-sleep: [Rule_22 :: powered implies -sleep;]
-grip_closed: [Rule_23 :: grip_open implies -grip_closed;]
awake: [Rule_27 :: -sleep implies awake;]
auto: [Rule_29 :: -manual implies auto;]
auto_perm: [Rule_19 :: awake, auto implies auto_perm;]

From rules 8, 10 and 18, it is evident that the current day is a weekend and therefore the inference to pick the part is not generated. In this manner the robotic arm is coached for not working on weekends.

In the fourth case, the operator enquires about the robot working beyond the standard production hours of 8:00 and 18:00. The operator does not receive a satisfactory explanation due to the evidence that no such exceptions exist in the robot's standard policy as well as the modified knowledge base. Therefore, the operator further adds the relevant exceptions (*Rule_4 to Rule_12*) which modifies the knowledge base in the following manner.

Rule_0 :: at_home, auto_perm, part_avl, grip_open, shift_active implies pick_part;
Rule_1 :: auto_perm, part_picked, -grip_open, shift_active implies place_part;
Rule_2 :: auto_perm, part_placed, grip_open, shift_active implies go_home;
Rule_3 :: -shift_active implies prod_hold;
Rule_4 :: time(X), ?<(X,8), weekday implies -cond1;
Rule_5 :: time(X), ?=(8,X), weekday implies cond1;
Rule_6 :: time(X), ?<(8,X), weekday implies cond1;
Rule_7 :: time(X), ?=(X,18), weekday implies cond2;
Rule_8 :: time(X), ?<(X,18), weekday implies cond2;
Rule_9 :: time(X), ?<(18,X), weekday implies -cond2;
Rule_10 :: -cond1, cond2 implies -shift_active;
Rule_11 :: cond1, -cond2 implies -shift_active;
Rule_12 :: cond1, cond2 implies shift_active;
Rule_13 :: day(X), ?=(X,1) implies monday;
Rule_14 :: day(X), ?=(X,2) implies tuesday;
Rule_15 :: day(X), ?=(X,3) implies wednesday;
Rule_16 :: day(X), ?=(X,4) implies thursday;
Rule_17 :: day(X), ?=(X,5) implies friday;
Rule_18 :: day(X), ?=(X,6) implies saturday;
Rule_19 :: day(X), ?=(X,7) implies sunday;
Rule_20 :: saturday implies weekend;

Rule_21 :: sunday implies weekend;

Rule_22 :: monday implies -weekend;

Rule_23 :: tuesday implies -weekend;

Rule_24 :: wednesday implies -weekend;

Rule_25 :: thursday implies -weekend;

Rule_26 :: friday implies -weekend;

Rule_27 :: -weekend implies weekday;

Rule_28 :: weekend implies -weekday;

Rule_29 :: awake, auto implies auto_perm;

Rule_30 :: -sleep, manual implies jog_perm;

Rule_31 :: -powered implies sleep;

Rule_32 :: powered implies -sleep;

Rule_33 :: grip_open implies -grip_closed;

Rule_34 :: -grip_open implies grip_closed;

Rule_35 :: jog_perm implies -auto_perm;

Rule_36 :: sleep implies -awake;

Rule_37 :: -sleep implies awake;

Rule_38 :: manual implies -auto;

Rule_39 :: -manual implies auto;

Then under the context which contains the following information, *powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(8);*

The following explanations are generated by the system.

cond1: [Rule_5 :: time(8), ?=(8, 8), weekday implies cond1;]

cond2: [Rule_8 :: time(8), ?<(8, 18), weekday implies cond2;]

shift_active: [Rule_12 :: cond1, cond2 implies shift_active;]

monday: [Rule_13 :: day(1), ?=(1, 1) implies monday;]

-weekend: [Rule_22 :: monday implies -weekend;]

weekday: [Rule_27 :: -weekend implies weekday;]

-sleep: [Rule_32 :: powered implies -sleep;]

-grip_closed: [Rule_33 :: grip_open implies -grip_closed;]

awake: [Rule_37 :: -sleep implies awake;]
auto: [Rule_39 :: -manual implies auto;]
auto_perm: [Rule_29 :: awake, auto implies auto_perm;]
pick_part: [Rule_0 :: at_home, auto_perm, part_avl, grip_open, shift_active implies pick_part;]

The above inferences suggest that from rules 13, 22 and 27 that it is a weekday and from rules 5, 8 and 12 that it is a standard shift hour and from rules 32, 33, 37, 39, 29 and 0 that it should pick the part.

When it is a weekday and the time is beyond the standard production hour, the robot now operates as desired.

Under the context, *powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(19);*

The following explanations are generated.

cond1: [Rule_6 :: time(19), ?<(8, 19), weekday implies cond1;]
-cond2: [Rule_9 :: time(19), ?<(18, 19), weekday implies -cond2;]
-shift_active: [Rule_11 :: cond1, -cond2 implies -shift_active;]
monday: [Rule_13 :: day(1), ?=(1, 1) implies monday;]
-weekend: [Rule_22 :: monday implies -weekend;]
weekday: [Rule_27 :: -weekend implies weekday;]
-sleep: [Rule_32 :: powered implies -sleep;]
-grip_closed: [Rule_33 :: grip_open implies -grip_closed;]
awake: [Rule_37 :: -sleep implies awake;]
auto: [Rule_39 :: -manual implies auto;]
prod_hold: [Rule_3 :: -shift_active implies prod_hold;]
auto_perm: [Rule_29 :: awake, auto implies auto_perm;]

From these explanations it becomes evident that from rules 13, 22 and 27 that it is a weekday and from rules 6, 9 and 11that it is a no production hour and from rules 32, 33, 37, 39, 3, and 29 that the robot should hold the production.

In this way the operators managed to coach the robotic arm based on various scenario demands to achieve the desired performance.

The various policies provided by the remaining four volunteers could be seen in the following table for experiment 1.

Volunteer	Policy	Context	Constraint	Result

1	Rule_0 :: auto_perm, -part_avl, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_picked implies place_part;
	Rule_2 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		-part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(5);
		time(2);
			Part Availability (Robot should work even though part is available or not)	Pass

	Rule_0 :: at_home, auto_perm, part_avl, grip_open, weekday implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, weekday implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, weekday implies go_home;
	Rule_3 :: awake, auto implies auto_perm;
	Rule_4 :: -sleep, manual implies jog_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: grip_open implies -grip_closed;
	Rule_8 :: -grip_open implies grip_closed;
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(1);
		time(8);
			Weekend (Robot should work only on weekdays)	Failed

	No response	powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(8);	Time Period (Robot should work only between standard production hours of 8:00 and 18:00)	Failed

2	Rule_0 :: auto_perm, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_picked implies place_part;
	Rule_2 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		powered; -manual; grip_open; -part_avl; -part_picked; -part_placed; at_home; day(5); time(2);	Part Availability (Robot should work even though part is available or not)	Pass

	Rule_0 :: auto_perm, part_avl, grip_open, weekday implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, weekday implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, weekday implies go_home;
	Rule_3 :: day(X), ?<(X,6) implies weekday;
	Rule_4 :: awake, auto implies auto_perm;
	Rule_5 :: -sleep, manual implies jog_perm;
	Rule_6 :: -powered implies sleep;
	Rule_7 :: powered implies -sleep;
	Rule_8 :: jog_perm implies -auto_perm;
	Rule_9 :: sleep implies -awake;
	Rule_10 :: -sleep implies awake;
	Rule_11 :: manual implies -auto;
	Rule_12 :: -manual implies auto;
		powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(8);	Weekend (Robot should work only on weekdays)	Pass

	Rule_0 :: auto_perm, part_avl, grip_open, weekday, start_work implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, weekday, start_work implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, weekday, start_work implies go_home;
	Rule_3 :: time(X), ?<(X,18) implies start_work;
	Rule_4 :: day(X), ?<(X,6) implies weekday;
	Rule_5 :: awake, auto implies auto_perm;
	Rule_6 :: -sleep, manual implies jog_perm;
	Rule_7 :: -powered implies sleep;
	Rule_8 :: powered implies -sleep;
	Rule_9 :: jog_perm implies -auto_perm;
	Rule_10 :: sleep implies -awake;
	Rule_11 :: -sleep implies awake;
	Rule_12 :: manual implies -auto;
	Rule_13 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(1);
		time(8);
			Time Period (Robot should work only between standard production hours of 8:00 and 18:00)	Failed

3	Rule_0 :: auto_perm, part_avl, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_picked implies place_part;
	Rule_2 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		*powered;
		-manual;
		grip_open;
		-part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(5);
		time(2);*
			Part Availability (Robot should work even though part is available or not)	Failed

	No Response	*powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(1);
		time(8);*
			Weekend (Robot should work only on weekdays)	Failed

	No Response	powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(1);
		time(8);
			Time Period (Robot should work only between standard production hours of 8:00 and 18:00)	Failed

4	Rule_0 :: auto_perm, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_picked implies place_part;
	Rule_2 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		-part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(5);
		time(2);
			Part Availability (Robot should work even though part is available or not)	Pass

	Rule_0 :: auto_perm, grip_open, weekday implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, weekday implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, weekday implies go_home;
	Rule_3 :: day(X), ?<(X,6) implies weekday;
	Rule_4 :: awake, auto implies auto_perm;
	Rule_5 :: -sleep, manual implies jog_perm;
	Rule_6 :: -powered implies sleep;
	Rule_7 :: powered implies -sleep;
	Rule_8 :: jog_perm implies -auto_perm;
	Rule_9 :: sleep implies -awake;
	Rule_10 :: -sleep implies awake;
	Rule_11 :: manual implies -auto;
	Rule_12 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		at_home;
		day(1);
		time(8);
			Weekend (Robot should work only on weekdays)	Pass

	Rule_0 :: auto_perm, part_avl, grip_open, weekday, start_work, start_work1 implies pick_part; Rule_1 :: auto_perm, part_picked, -grip_open, weekday, start_work, start_work1 implies place_part; Rule_2 :: auto_perm, part_placed, grip_open, weekday, start_work, start_work1 implies go_home; Rule_3 :: time(X), ?<(8,X) implies start_work; Rule_4 :: time(X), ?<(X,18) implies start_work1; Rule_5 :: day(X), ?<(X,6) implies weekday; Rule_6 :: awake, auto implies auto_perm; Rule_7 :: -sleep, manual implies jog_perm; Rule_8 :: -powered implies sleep; Rule_9 :: powered implies -sleep; Rule_10 :: jog_perm implies -auto_perm; Rule_11 :: sleep implies -awake; Rule_12 :: -sleep implies awake; Rule_13 :: manual implies -auto; Rule_14 :: -manual implies auto;	powered; -manual; grip_open; part_avl; -part_picked; -part_placed; at_home; day(1); time(8);	Time Period (Robot should work only between standard production hours of 8:00 and 18:00)	Failed (The robot works between 9:00 and 17:00 though)

Table 3. Policies provided by volunteers under various constraints for Experiment 1

Experiment 2: Part handling between multiple points.

In the second experiment the robot's task is to Pick a part from the pick point on the infeed conveyor and place it to the place point on the outfeed conveyor. After placing the part, the arm has to return to the home position and loop in a similar fashion. As an additional complexity, the parts that arrive on the conveyor could be faulty or non-faulty which are sensed through the vision system dynamically on the conveyor and the respective part status is sent to the robot as an input. The peripheral devices connected to the robot also suggest any human presence in the vicinity of the robot. All the three Pick, Place and Home positions are static locations which have already been taught manually to the robotic arm. In order to enhance the user's comprehension of the system's explanations, an additional Download button is added on the console of the PRUDENS interface.

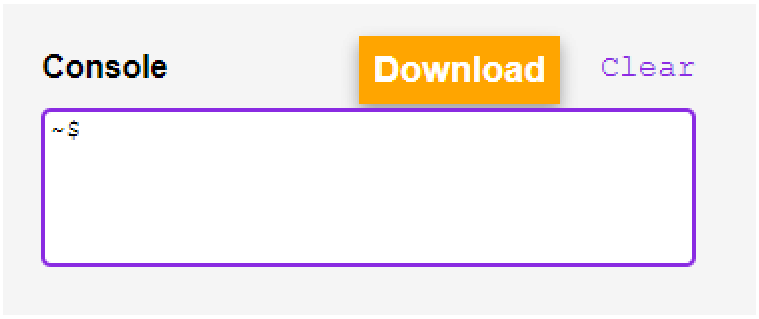
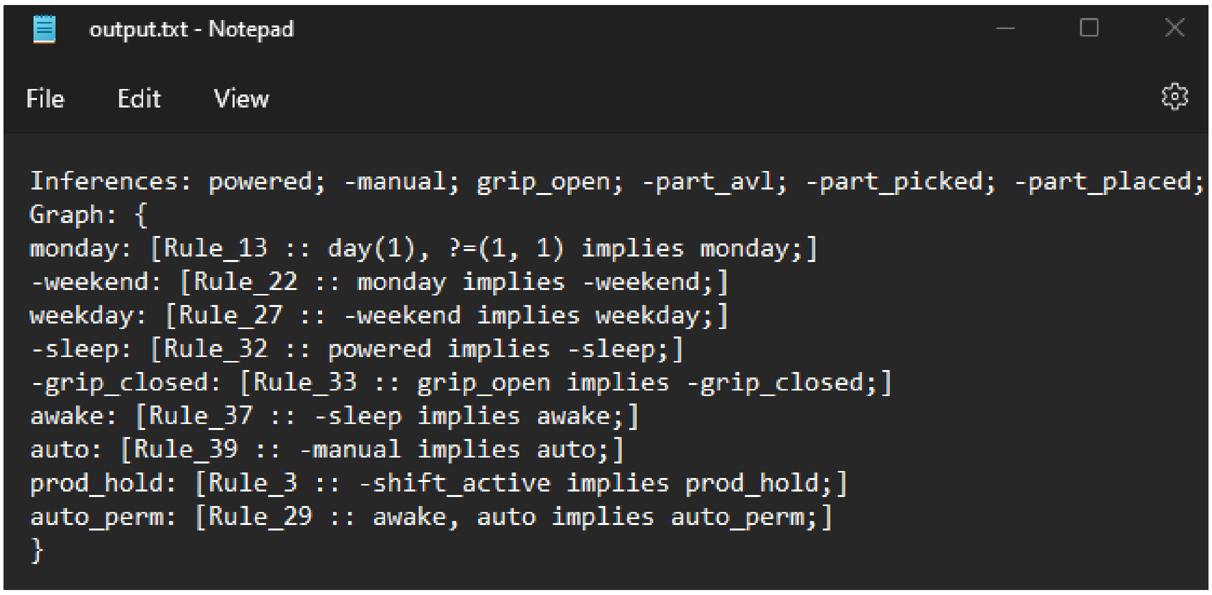


Figure 13. PRUDENS Console Download Button

This download button enables the user to download the explanations generated by the system in the form of text file as below. This further enhances the user to keep a record of the agent's arguments as part of the coaching process.



```
Inferences: powered; -manual; grip_open; -part_avl; -part_picked; -part_placed;
Graph: {
monday: [Rule_13 :: day(1), ?=(1, 1) implies monday;]
-weekend: [Rule_22 :: monday implies -weekend;]
weekday: [Rule_27 :: -weekend implies weekday;]
-sleep: [Rule_32 :: powered implies -sleep;]
-grip_closed: [Rule_33 :: grip_open implies -grip_closed;]
awake: [Rule_37 :: -sleep implies awake;]
auto: [Rule_39 :: -manual implies auto;]
prod_hold: [Rule_3 :: -shift_active implies prod_hold;]
auto_perm: [Rule_29 :: awake, auto implies auto_perm;]
}
```

Figure 14. Example Text file on Download

In the initial phase the robot follows a standard policy with the following rules in the knowledge base of the PRUDENS server that drive the robot based on the context.

Rule_0 :: auto_perm, grip_open implies pick_part;

Rule_1 :: auto_perm, part_picked implies place_part;

Rule_2 :: auto_perm, part_placed implies go_home;

Rule_3 :: -sleep, auto implies auto_perm;

Rule_4 :: -powered implies sleep;

Rule_5 :: powered implies -sleep;

Rule_6 :: manual implies -auto;
Rule_7 :: -manual implies auto;

At an instant the context received from the robotic arm is as under.

powered; -manual; grip_open; -part_avl; -part_picked; -part_placed; -human_det; -part_def; -rob_run; at_home; day(1); time(8);

The inferences received by the robotic arm are *-sleep; auto; auto_perm; pick_part.* When the robot picks the part then it moves to place it through the inferences *-sleep; auto; auto_perm; place_part.* On placing the part, the robot goes to its home position through -*sleep; auto; auto_perm; pick_part; go_home.*

In another instance, the operator observes the arrival of the defected part which the robot picks and places at the non-defected part position in the generic manner. This is observed as below.

Context under which Picking of the part takes place.

powered; -manual; grip_open; part_avl; -part_picked; -part_placed; -human_det; ***part_def;*** *-rob_run; at_home; day(1); time(8);*

Explanations generated as *-sleep; auto; auto_perm; pick_part.*

Context under which Placing of the part takes place.

powered; -manual; -grip_open; part_avl; part_picked; -part_placed; -human_det; ***part_def;*** *rob_run; -at_home; day(1); time(8);*

Explanations generated as *-sleep; auto; auto_perm; place_part.*

The robot then moves to its home position as part of its standard policy.

On enquiring about this behavior of placing the defected part to the non-defected part's position, the operator gets the following explanation.

-sleep: [Rule_5 :: powered implies -sleep;]
auto: [Rule_7 :: -manual implies auto;]
auto_perm: [Rule_3 :: -sleep, auto implies auto_perm;]
place_part: [Rule_1 :: auto_perm, part_picked implies place_part;]

The operator agrees with the system's explanation and decides to add exception in the knowledge base in order to place the defected part at its respective location which has already been taught to the robot. The modified knowledge base now looks as below.

Rule_0 :: auto_perm, grip_open implies pick_part;
Rule_1 :: auto_perm, part_picked, part_def implies place_def;
Rule_2 :: auto_perm, part_picked, **-part_def** implies place_part;
Rule_3 :: auto_perm, part_placed implies go_home;
Rule_4 :: -sleep, auto implies auto_perm;
Rule_5 :: -powered implies sleep;
Rule_6 :: powered implies -sleep;
Rule_7 :: manual implies -auto;
Rule_8 :: -manual implies auto;

The operator has added an exception as *Rule_1* and added an additional predicate in *Rule_2* in order to place the defected part to its respective location.

Now under the similar context as above, the following explanations are generated,
-sleep; auto; auto_perm; place_def.

-sleep: [Rule_6 :: powered implies -sleep;]
auto: [Rule_8 :: -manual implies auto;]
auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]
place_def: [Rule_1 :: auto_perm, part_picked, part_def implies place_def;]

In the third instance, the operator observes human presence in the vicinity of the robot but the robot continues to operate without stopping. This is observed as below.

Context under which Picking of the part takes place.

powered; -manual; grip_open; part_avl; -part_picked; -part_placed; ***human_det;*** *-part_def; -rob_run; at_home; day(1); time(8);*

Explanations generated as *-sleep; auto; auto_perm; pick_part.*

Context under which Placing of the part takes place.

powered; -manual; -grip_open; part_avl; part_picked; -part_placed; ***human_det;*** *-part_def; rob_run; -at_home; day(1); time(8);*

Explanations generated as *-sleep; auto; auto_perm; place_part.*

The robot then moves to its home position as part of its standard policy.

On enquiring about this behavior of continued operation even under human presence in the vicinity, the operator gets the following explanation.

While Picking the part.

-sleep: [Rule_6 :: powered implies -sleep;]
auto: [Rule_8 :: -manual implies auto;]
auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]
pick_part: [Rule_0 :: auto_perm, grip_open implies pick_part;]

While Placing the part.

-sleep: [Rule_6 :: powered implies -sleep;]
auto: [Rule_8 :: -manual implies auto;]

auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]

place_part: [Rule_2 :: auto_perm, part_picked, -part_def implies place_part;]

While Going to home position.

-sleep: [Rule_6 :: powered implies -sleep;]

auto: [Rule_8 :: -manual implies auto;]

auto_perm: [Rule_4 :: -sleep, auto implies auto_perm;]

pick_part: [Rule_0 :: auto_perm, grip_open implies pick_part;]

go_home: [Rule_3 :: auto_perm, part_placed implies go_home;]

The operator presents its counter argument in the form of several exceptions pertaining to human safety to achieve the required behavior such that the robot stops working on the detection of human in its vicinity. The knowledge base now looks as under.

Rule_0 :: auto_perm, rob_run, human_det implies rob_stop;

Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;

Rule_2 :: at_home, auto_perm, grip_open, **-rob_stop** implies pick_part;

Rule_3 :: auto_perm, part_picked, -grip_open, **-rob_stop**, part_def implies place_def;

Rule_4 :: auto_perm, part_picked, -grip_open, **-rob_stop**, -part_def implies place_part;

Rule_5 :: auto_perm, part_placed, grip_open, **-rob_stop** implies go_home;

Rule_6 :: -sleep, auto implies auto_perm;

Rule_7 :: -powered implies sleep;

Rule_8 :: powered implies -sleep;

Rule_9 :: manual implies -auto;

Rule_10 :: -manual implies auto;

Rule_11 :: rob_run, -human_det implies -rob_stopped;

Rule_12 :: -rob_run, human_det implies rob_stopped;

Now under the similar context as above, the following explanations are generated,
-sleep; auto; rob_stopped; auto_perm

-sleep: [Rule_8 :: powered implies -sleep;]

auto: [Rule_10 :: -manual implies auto;]
rob_stopped: [Rule_12 :: -rob_run, human_det implies rob_stopped;]
auto_perm: [Rule_6 :: -sleep, auto implies auto_perm;]

In the fourth instance, the operator enquires about the robot not switching off its power during the lunch interval which takes places between 12:00 – 13:00 hours everyday.

Context under which Picking of the part takes place.

powered; -manual; grip_open; part_avl; -part_picked; -part_placed; -human_det; -part_def; -rob_run; at_home; day(1); ***time(12);***

The following explanations are generated.

-sleep: [Rule_8 :: powered implies -sleep;]
auto: [Rule_10 :: -manual implies auto;]
auto_perm: [Rule_6 :: -sleep, auto implies auto_perm;]
-rob_stop: [Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;]
pick_part: [Rule_2 :: at_home, auto_perm, grip_open, -rob_stop implies pick_part;]

Context under which Placing of the part takes place.

powered; -manual; -grip_open; part_avl; part_picked; -part_placed; -human_det; -part_def; rob_run; -at_home; day(1); ***time(12);***

The following explanations are generated.

-sleep: [Rule_8 :: powered implies -sleep;]
auto: [Rule_10 :: -manual implies auto;]
auto_perm: [Rule_6 :: -sleep, auto implies auto_perm;]
-rob_stop: [Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;]
place_part: [Rule_4 :: auto_perm, part_picked, -grip_open, -rob_stop, -part_def implies place_part;]

Context under which Home movement takes place.

powered; -manual; grip_open; part_avl; -part_picked; part_placed; -human_det; -part_def; -rob_run; -at_home; day(1); ***time(12);***

The following explanations are generated.

-sleep: [Rule_8 :: powered implies -sleep;]
auto: [Rule_10 :: -manual implies auto;]
auto_perm: [Rule_6 :: -sleep, auto implies auto_perm;]
-rob_stop: [Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;]
go_home: [Rule_5 :: auto_perm, part_placed, grip_open, -rob_stop implies go_home;]

The operator presents its counter argument in the form of several exceptions and facts such that the robot switches off its power during the lunch interval. The knowledge base now looks as under.

Rule_0 :: auto_perm, rob_run, human_det implies rob_stop;
Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;
Rule_2 :: at_home, auto_perm, grip_open, **-power_off**, -rob_stop implies pick_part;
Rule_3 :: auto_perm, part_picked, -grip_open, **-power_off**, -rob_stop, part_def implies place_def;
Rule_4 :: auto_perm, part_picked, -grip_open, **-power_off**, -rob_stop, -part_def implies place_part;
Rule_5 :: auto_perm, part_placed, grip_open, **-power_off**, -rob_stop implies go_home;
Rule_6 :: lunch_time implies power_off;
Rule_7 :: -lunch_time implies -power_off;
Rule_8 :: time(X), ?<(X,12) implies -cond1;
Rule_9 :: time(X), ?=(12,X) implies cond1;
Rule_10 :: time(X), ?<(12,X) implies cond1;
Rule_11 :: time(X), ?=(X,13) implies cond2;

Rule_12 :: time(X), ?<(X,13) implies cond2;

Rule_13 :: time(X), ?<(13,X) implies -cond2;

Rule_14 :: -cond1, cond2 implies -lunch_time;

Rule_15 :: cond1, -cond2 implies -lunch_time;

Rule_16 :: cond1, cond2 implies lunch_time;

Rule_17 :: -sleep, auto implies auto_perm;

Rule_18 :: -powered implies sleep;

Rule_19 :: powered implies -sleep;

Rule_20 :: manual implies -auto;

Rule_21 :: -manual implies auto;

Rule_22 :: rob_run, -human_det implies -rob_stopped;

Rule_23 :: -rob_run, human_det implies rob_stopped;

Now under the similar contexts as above, the following explanations are generated,
cond1; cond2; lunch_time; -sleep; auto; power_off; auto_perm; -rob_stop

cond1: [Rule_9 :: time(12), ?=(12, 12) implies cond1;]

cond2: [Rule_12 :: time(12), ?<(12, 13) implies cond2;]

lunch_time: [Rule_16 :: cond1, cond2 implies lunch_time;]

-sleep: [Rule_19 :: powered implies -sleep;]

auto: [Rule_21 :: -manual implies auto;]

power_off: [Rule_6 :: lunch_time implies power_off;]

auto_perm: [Rule_17 :: -sleep, auto implies auto_perm;]

-rob_stop: [Rule_1 :: auto_perm, -rob_run, -human_det implies -rob_stop;]

In this way, the robot switches off its power during the lunch interval at the production facility thereby reducing energy consumption. Therefore, the operators have managed to coach the robotic arm based on various scenario demands to achieve the desired performance.

The various policies provided by the remaining four volunteers could be seen in the following table for experiment 2.

Volunteer	**Policy**	**Context**	**Constraint**	**Result**
1	Rule_0 :: auto_perm, part_avl, grip_open implies pick_part; Rule_1 :: auto_perm, part_picked implies place_part_def; Rule_2 :: auto_perm, part_placed implies go_home; Rule_4 :: -sleep, auto implies auto_perm; Rule_5 :: -powered implies sleep; Rule_6 :: powered implies -sleep; Rule_7 :: manual implies -auto; Rule_8 :: -manual implies auto;	*powered; -manual; grip_open; part_avl; -part_picked; -part_placed; -human_det; part_def; -rob_run; at_home; day(1); time(8);*	Faulty Parts (Robot should place the faulty parts faulty part position & non-faulty parts at their respective location)	Failed
	Rule_0 :: auto_perm, rob_run, -human_det implies robo_work; Rule_1 :: at_home, auto_perm, part_avl, grip_open, robo_work implies pick_part; Rule_2 :: auto_perm, part_picked, -grip_open, robo_work implies place_part; Rule_3 :: auto_perm, part_placed, grip_open, robo_work implies go_home; Rule_4 :: -sleep, auto implies auto_perm; Rule_5 :: -powered implies sleep; Rule_6 :: powered implies -sleep; Rule_7 :: manual implies -auto; Rule_8 :: -manual implies auto;	*powered; -manual; grip_open; part_avl; -part_picked; -part_placed; human_det; -part_def; rob_run; at_home; day(1); time(8);*	Human Presence (Robot should stop on detection of human nearby)	Pass

	Rule_0 :: auto_perm, rob_run, -human_det implies robo_work;
	Rule_1 :: at_home, auto_perm, part_avl, grip_open, robo_work, halt, halt1 implies pick_part;
	Rule_2 :: auto_perm, part_picked, -grip_open, robo_work, halt, halt1 implies place_part;
	Rule_3 :: auto_perm, part_placed, grip_open, robo_work, halt, halt1 implies go_home;
	Rule_4 :: time(X), ?=(X,12) implies halt;
	Rule_5 :: time(X), ?=(X,13) implies halt1;
	Rule_6 :: -sleep, auto implies auto_perm;
	Rule_7 :: -powered implies sleep;
	Rule_8 :: powered implies -sleep;
	Rule_9 :: manual implies -auto;
	Rule_10 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		part_placed;
		-human_det;
		-part_def;
		-rob_run;
		-at_home;
		day(1);
		time(12);
			Power Consumption (Robot must switch off its power during lunch hour of 12:00 - 13:00)	Failed

2	No response	*powered;*
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		-human_det;
		part_def;
		-rob_run;
		at_home;
		day(1);
		time(9);
			Faulty Parts (Robot should place the faulty parts faulty part position & non-faulty parts at their respective location)	Failed

	No response	*powered; -manual; grip_open; part_avl; -part_picked; -part_placed; human_det; -part_def; rob_run; at_home; day(1); time(9);*	Human Presence (Robot should stop on detection of human nearby)	Failed

	Rule_4 :: time(X), ?<(X,12) implies alarm;
	Rule_5 :: -sleep, auto implies auto_perm;
	Rule_6 :: -powered implies sleep;
	Rule_7 :: powered implies -sleep;
	Rule_8 :: manual implies -auto;
	Rule_9 :: -manual implies auto;
		powered; -manual; grip_open; part_avl; -part_picked; part_placed; -human_det; -part_def; -rob_run; -at_home; day(1); time(12);	Power Consumption (Robot must switch off its power during lunch hour of 12:00 - 13:00)	Failed

3	Rule_0 :: auto_perm, part_avl, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_picked, part_def implies place_part_def;
	Rule_2 :: auto_perm, part_picked, -part_def implies place_part;
	Rule_3 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		powered; -manual; grip_open; part_avl; -part_picked; -part_placed; -human_det; part_def; -rob_run; at_home; day(2); time(8);	Faulty Parts (Robot should place the faulty parts faulty part position & non-faulty parts at their respective location)	Pass

	Rule_0 :: at_home, auto_perm, part_avl, grip_open, robo_continue implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, robo_continue implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, robo_work implies go_home;
	Rule_3 :: auto_perm, rob_run, -human_det implies robo_continue;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		human_det;
		-part_def;
		rob_run;
		at_home;
		day(2);
		time(8);
			Human Presence (Robot should stop on detection of human nearby)	Pass

	No response	powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		part_placed;
		-human_det;
		-part_def;
		-rob_run;
		-at_home;
		day(1);
		time(12);
			Power Consumption (Robot must switch off its power during lunch hour of 12:00 - 13:00)	Failed

4	Rule_0 :: auto_perm, part_avl, grip_open implies pick_part;
	Rule_1 :: auto_perm, part_def, part_picked implies place_defected;
	Rule_2 :: auto_perm, -part_def, part_picked implies place_part;
	Rule_3 :: auto_perm, part_placed implies go_home;
	Rule_4 :: -sleep, auto implies auto_perm;
	Rule_5 :: -powered implies sleep;
	Rule_6 :: powered implies -sleep;
	Rule_7 :: manual implies -auto;
	Rule_8 :: -manual implies auto;
		*powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		-human_det;
		part_def;
		-rob_run;
		at_home;
		day(2);
		time(8);*
			Faulty Parts (Robot should place the faulty parts faulty part position & non-faulty parts at their respective location)	Pass

	Rule_0 :: at_home, auto_perm, part_avl, grip_open, -human_det implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, -human_det implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, -human_det implies go_home;
	Rule_3 :: -sleep, auto implies auto_perm;
	Rule_4 :: -powered implies sleep;
	Rule_5 :: powered implies -sleep;
	Rule_6 :: manual implies -auto;
	Rule_7 :: -manual implies auto;
		*powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		-part_placed;
		human_det;
		-part_def;
		rob_run;
		at_home;
		day(2);
		time(8);*
			Human Presence (Robot should stop on detection of human nearby)	Pass

	Rule_0 :: at_home, auto_perm, part_avl, grip_open, -human_det implies pick_part;
	Rule_1 :: auto_perm, part_picked, -grip_open, -human_det implies place_part;
	Rule_2 :: auto_perm, part_placed, grip_open, -human_det implies go_home;
	Rule_3 :: time(Time), ?<(Time,12) implies no_halt;
	Rule_4 :: time(Time), ?<(13,Time) implies no_halt1;
	Rule_5 :: -sleep, auto implies auto_perm;
	Rule_6 :: -powered implies
		powered;
		-manual;
		grip_open;
		part_avl;
		-part_picked;
		part_placed;
		-human_det;
		-part_def;
		-rob_run;
		-at_home;
		day(1);
		time(12);
			Power Consumption (Robot must switch off its power during lunch hour of 12:00 - 13:00)	Failed

	sleep;
	Rule_7 :: powered implies -sleep;
	Rule_8 :: manual implies -auto;
	Rule_9 :: -manual implies auto;			

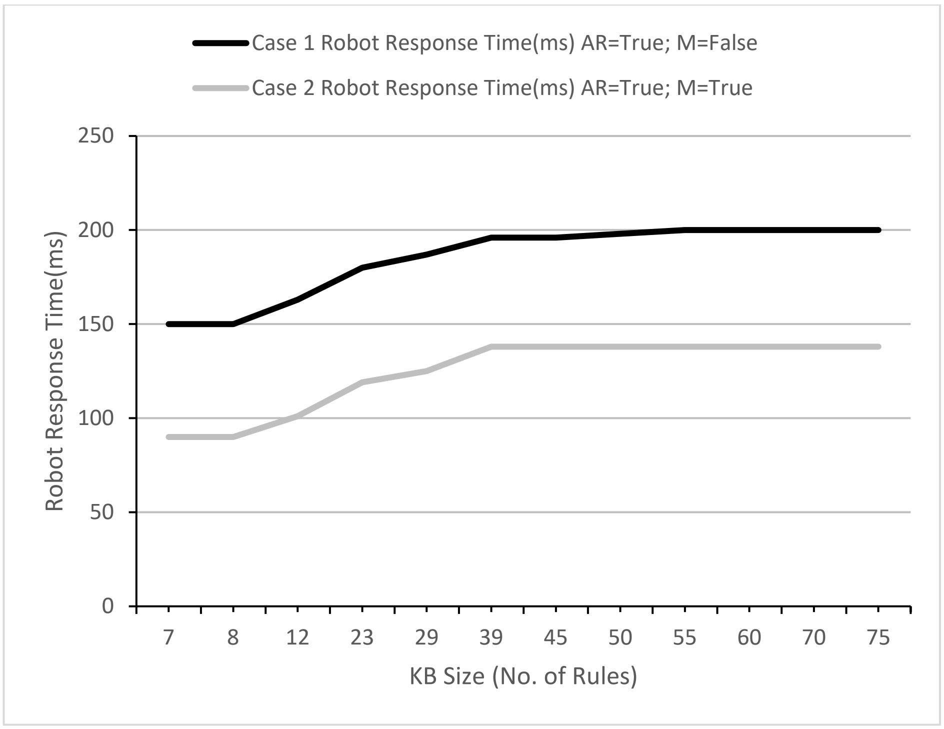
Table 4. Policies provided by volunteers under various constraints for Experiment 2

5.1 Intrinsic System Performance

As our proposed system is a combination of software applications running on distinct platforms that involve communication between the two nodes, the PC (PRUDENS) and the robot, we found it crucial to determine the performance of the combined system based on the internal software parameters. We observed the robot's response time in the processing of an information exchange cycle between the two nodes at varied sizes of the knowledge base where the robot was responsive enough to execute correct motion under two separate cases. One information exchange cycle consists of the compilation and transfer of contexts from the robot to PRUDENS, processing of the context, inference generation based on the rules and transfer of the output to the robot from PRUDENS, and finally processing of the received information and execution of motion by the robot. This was achieved by observing the robot's activity at cyclic delays ranging from 4ms to 200ms. For each cyclic delay, 20 robotic operation cycles were performed. For each KB size, the cyclic delays at which the robot successfully performed 20 operation cycles were recorded as the robot's response time, which is shown in Table 5 below. In the first case, we kept the Automatic Reply (AR) option on the PRUDENS server active and the Memory (M) option deactivated. However, in the second case, we kept both the options active. This test was performed in a standard production environment where the robot controller was directly connected to the PC through a CAT6 ethernet cable. The PC on which the PRUDENS server was running consists of an Intel i7-11800H @ 2.30GHz processor, 16.0 GB of RAM, and Windows 11 Enterprise as the operating system. Google Chrome was used as the standard browser for the PRUDENS server. The Robot controller on the other hand consists of Atom @ 1.9Ghz processor and a 2.0 GB RAM. The internal communication speed of the CPU is 4 ms. The control pendant of the robotic arm, which displays the robot's interface, is a 7-inch touchscreen that displays user pages in HTML format.

S. No.	KB Size (No. of Rules)	Case 1 Robot Response Time(ms) AR=True; M=False	Case 2 Robot Response Time(ms) AR=True; M=True
1.	**7**	**150**	**90**
2.	**8**	**150**	**90**
3.	**12**	**163**	**101**
4.	**23**	**180**	**119**
5.	**29**	**187**	**125**
6.	**39**	**196**	**138**
7.	**45**	**196**	**138**
8.	**50**	**198**	**138**
9.	**55**	**200**	**138**
10.	**60**	**200**	**138**
11.	**70**	**200**	**138**
12.	**75**	**200**	**138**

Table 5. Robot response time data at distinct KB size with Case 1 and Case 2 functionalities



Graph 2. KB Size (No. of Rules) vs Robot Response Times (ms) in Cases 1 & 2

From Graph 2 we observe that with the increase in KB size in Case 1, the Robot's response time in one communication cycle shows a sharp increase up to 200ms after which it gets stabilized irrespective of the increase in the number of rules in the KB. However, it is observed that in Case 2, the robot response time attains stability much before as compared to Case 1. Therefore, the cache memory functionality offered an increased performance at large KB size.

	KB Size (No. of Rules)	Case 1 Robot Response Time(ms)	Case 2 Robot Response Time(ms)
KB Size (No. of Rules)	**1**		
Case 1 Robot Response Time(ms)	**0.900575074**	**1**	
Case 2 Robot Response Time(ms)	**0.884384466**	**0.996925001**	**1**

Table 6. Correlation b/w KB Size and Robot Response Times in Case 1 & 2

From Table 6, we observe that the robot response times in both the cases exhibit high correlation with the size of the KB as well as with each other. It is however observed that the correlation between the Case 2 response time and the KB is lower as compared to that in Case 1. This again suggests that after a particular value of robot's response time in Case 2 the influence of the size of KB is decreased.

5.2 Extrinsic System Performance

In order to have an extrinsic evaluation of the system's performance from the cognitive point of view, we prepared several tasks pertaining to the system's use and functionality in a real time industrial environment that each participant needs to perform. To achieve this an initial training regarding PRUDENS and Staubli Robot was provided to the participants. Subsequent to the task phase, the participants were asked to record their feedback through a questionnaire. Each participant was informed about the motive of the evaluation and was asked to give their consent as part of the data collection and data handling policy before beginning with the survey. Ten volunteers (eight male, two female) from the industry having ages ranging from twenty-two to fifty-five voluntarily participated in the evaluation process. All the participants possessed an intermediate to fluent level of English language. In the initial phase of the evaluation, we proceed with a demographic survey of the participants as below.

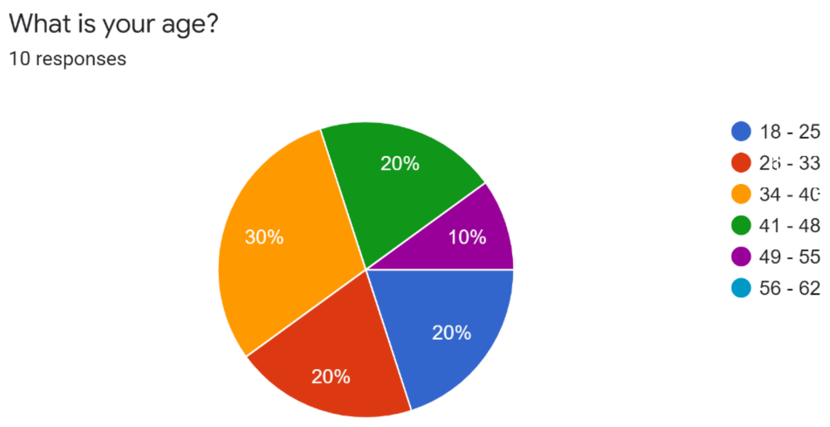


Figure 15

30% of the participants fall under the age group of 34 – 40 years. This suggests that evaluation would be done by people who are greatly acquainted to advanced industrial technologies. We also observe that there is an equal distribution of participants that lie in the age groups of 26 – 33 years, 41- 48 years and 18 - 25 years. It suggests that we will receive equal opinions about the system from distinct age groups. We also found that 10% of the participants fall under the age group of 49 – 55 years which determines that evaluation will also contain feedback from people who possess a more manual approach towards industrial solutions.

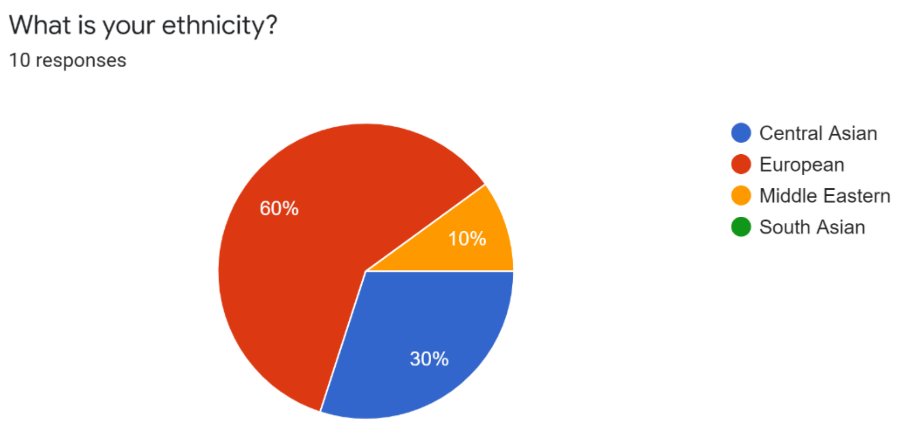


Figure 16

From figure 16, it is evident that people from three different ethnicities will be evaluating the system. They would enable us to understand their approach towards the system's usage according to which further development of the system could be customized.

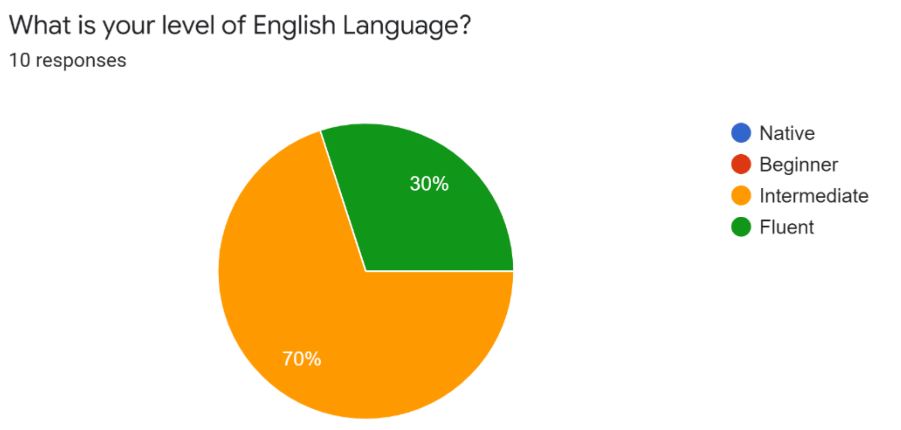


Figure 17

Majority of the participants possess Intermediate English level whereas the remaining classify themselves as fluent English users. This has enabled us to test the system on English comprehension-based factors.

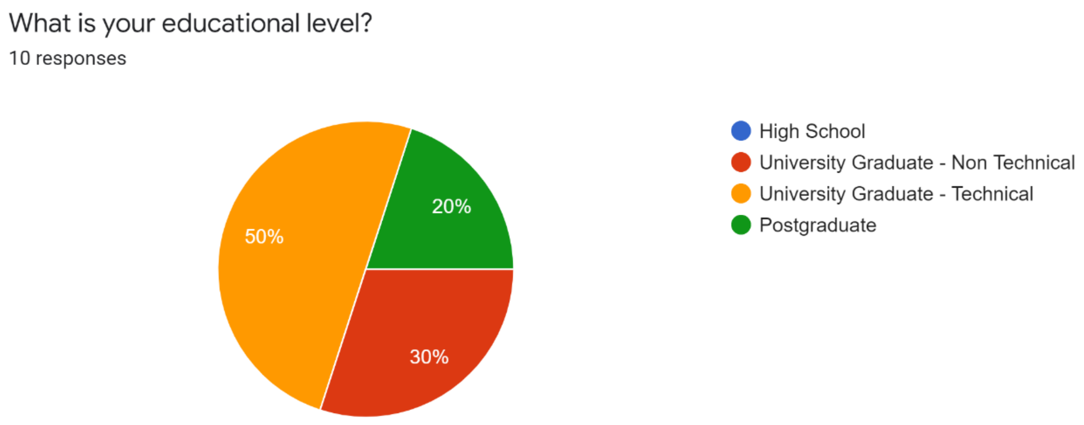


Figure 18

Most of the participants are from a technical background. This implies for a detailed technology evaluation.

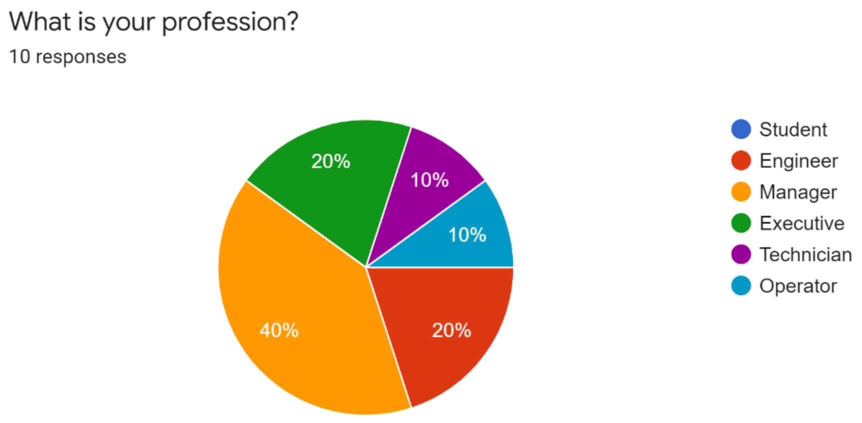


Figure 19

40% of the participants work as Managers whereas the remaining show a mixed distribution of Engineers, Executives, Technicians and Operators. This would enable an evaluation pertaining to the user's approach towards the system from various professions.

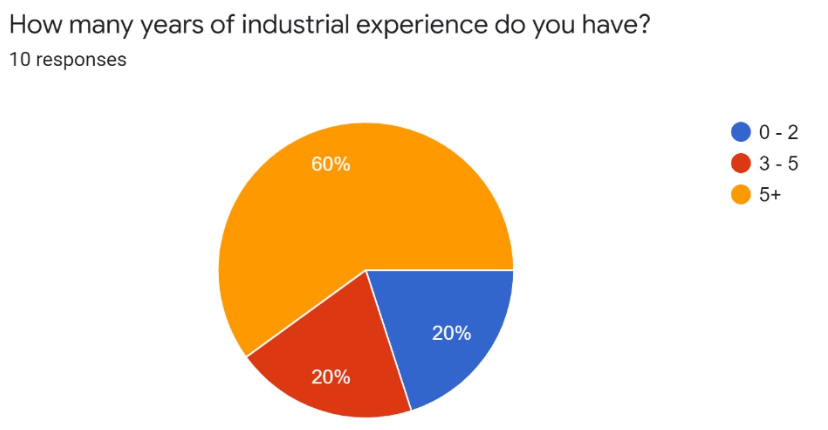


Figure 20

Most of the participants possess an industrial experience of over five years. This has enabled us to get the opinion about the system from industry experts with strong technical know-how.

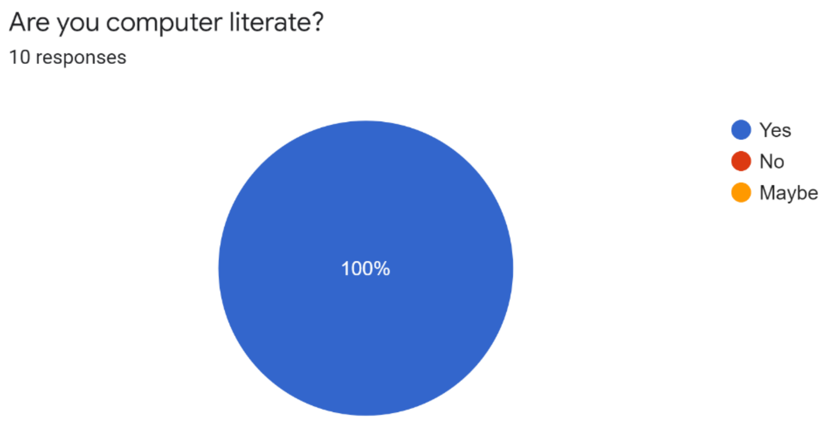


Figure 21

All the participants are familiar with working on computers. As our system is a computer-based application therefore computer proficiency was a major requirement from the usage and evaluation point of view.

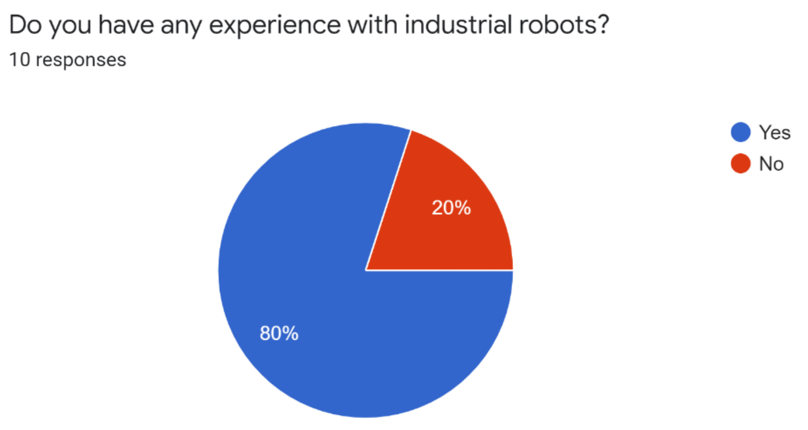


Figure 22

80% of the participants know how to use industrial robots. This shows that most of the participants are acquainted with the usage of industrial robots. Due to this fact they would evaluate the system keeping in mind the complexities and the challenges they face while dealing with the standardized robotic applications.

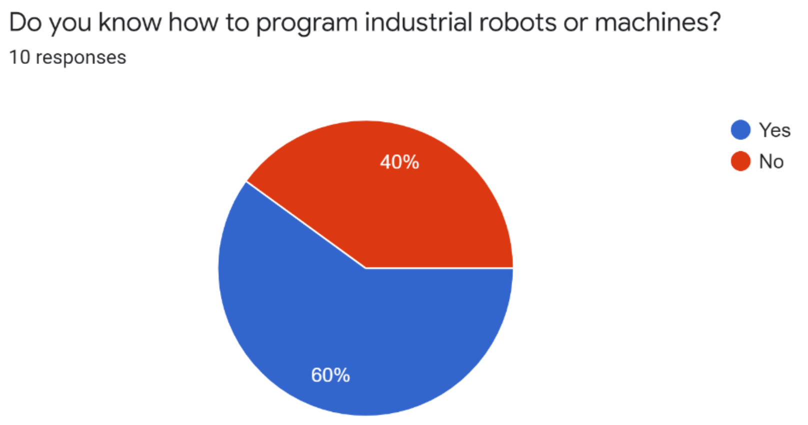


Figure 23

60% of the participants are familiar with industrial robot or machine programming. This shows that most of the participants are proficient with the programming of the industrial systems. This would enable them to distinguish between the software-based challenges and production loss when they need to modify something pertaining to the robot's or machine's action and how they approach towards the same through our system.

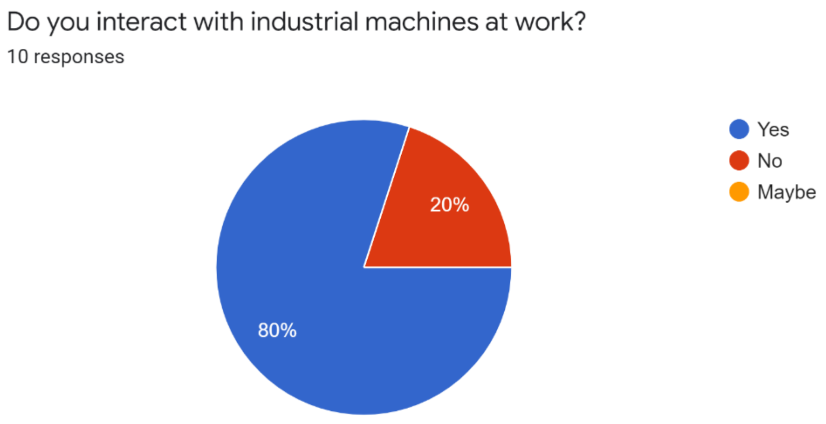


Figure 24

Most of the participants are engaged with some kind of Human-Machine Interaction. This determines that they are aware of the challenges they face during the interaction process. This would in turn help us to get our system evaluated from HMI experienced professionals.

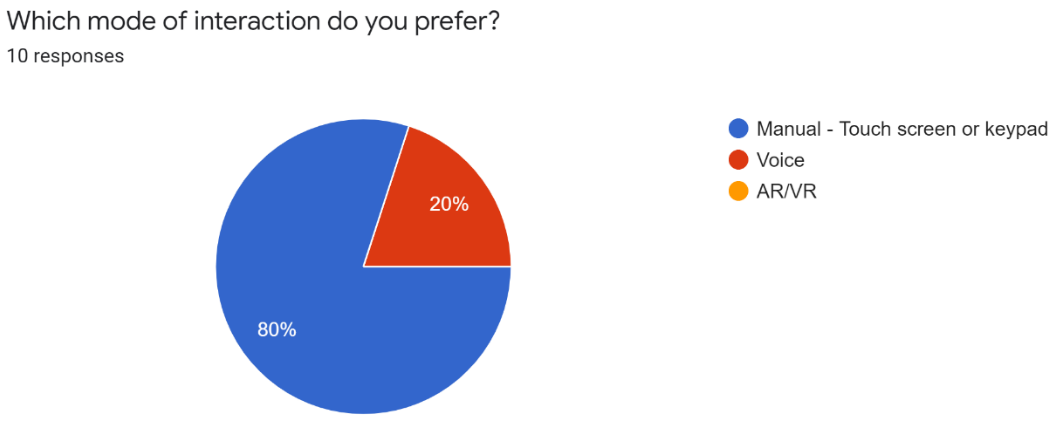


Figure 25

Much to a surprise, in the era of industry 4.0, most of the participants prefer Manual HMI methods through touch and visualization. This determines the general human outlook when it comes to adapting new technology. Humans generally feel comfortable with using tools that are common among masses.

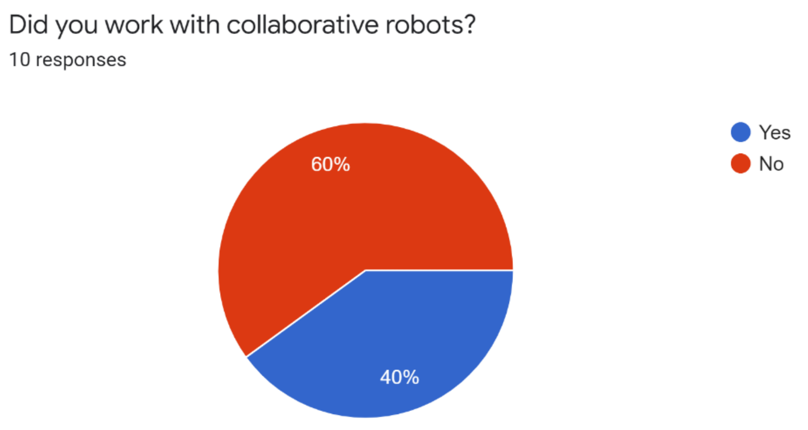


Figure 26

40% of the participants have the experience to work with collaborative robots in the industry. Collaborative robots are the industrial robots that are designed to work with humans. They are programmed to work at a speed that would produce minimum energy such that it should cause no damage in case it strikes with any human body part. In today's industries, collaborative robots have become a necessity to increase production and offer safety. However, when it comes to establishing communication between human and the robot classical approaches are followed. Low collaborative experience of the participants would create some discrepancy in the evaluation results.

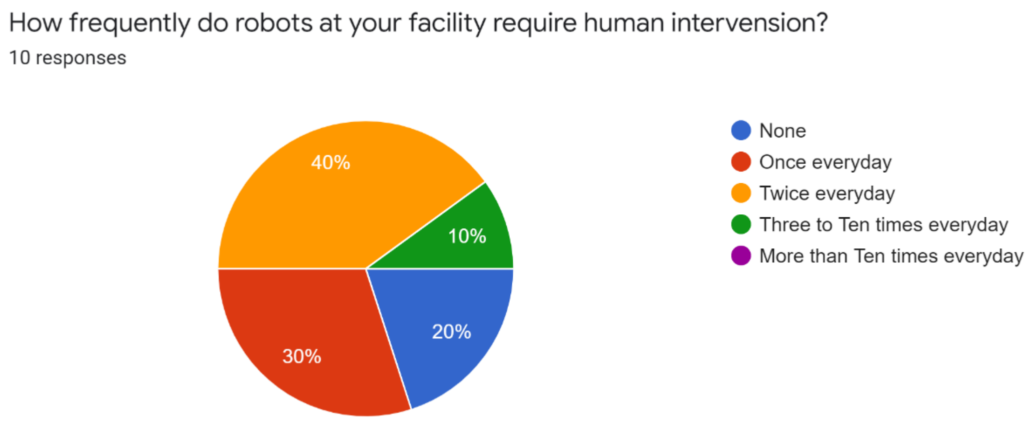


Figure 27

Only 20% of the participants suggest that their industrial robots do not require any kind of human intervention. This data might not be so correct as almost all the machine operators or engineers intervene with the robots at least once every day in order to check for logs or acknowledge alarms.

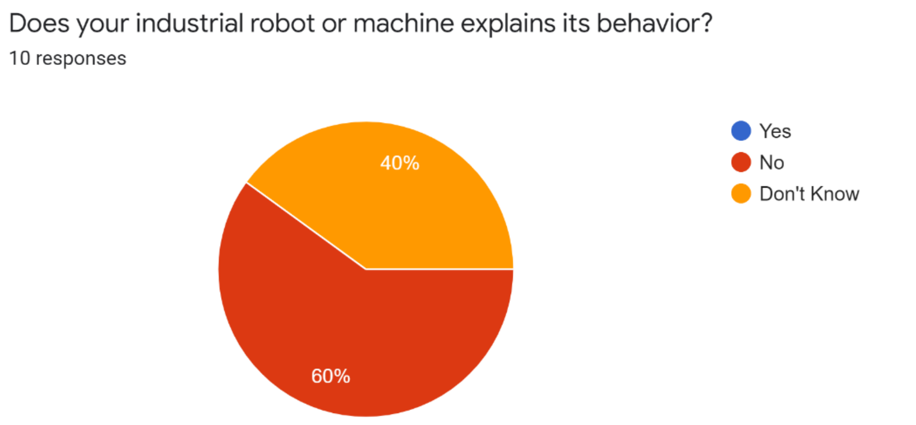


Figure 28

This suggests that most of the participants still have no experience working with smart machines in the industry. This determines that they might find our system a bit complex to comprehend in the initial phase.

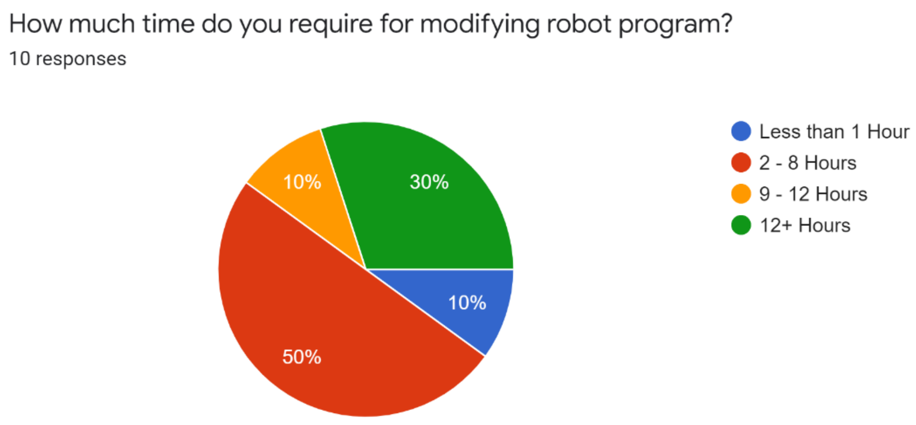


Figure 29

Generally, participants require 2 – 8 work hours in order to modify the robot program. This suggests a minimum of 2 – 8 hours of production down time.

As part of the second phase of the evaluation, we proceed with the cognition-based task execution. The observations are below.

1. Download the Robot Application into the Robot Controller.

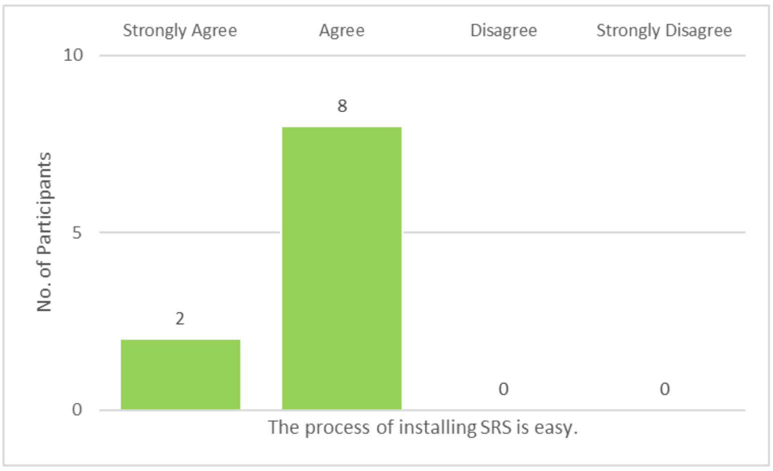


Figure 30

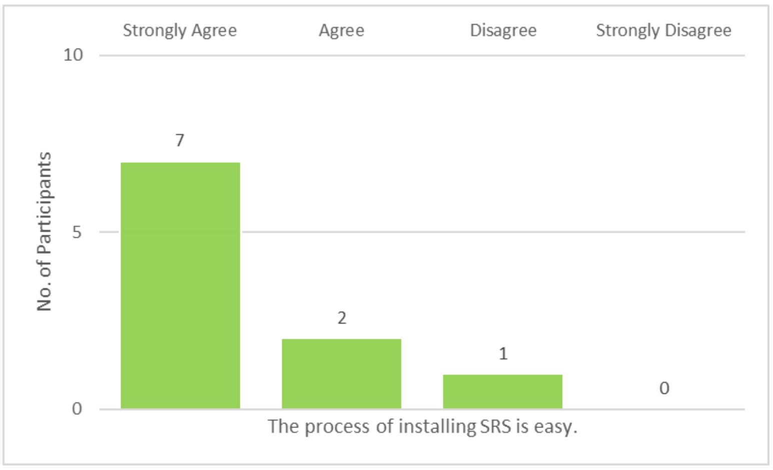


Figure 31

The process of downloading the robot application into the robot controller is an important step towards using our proposed system. It requires the installation of the Staubli Robotics Suits (SRS) into the PC. Once SRS is installed, the user has to send the API to the robot through FTP that is facilitated by the Transfer Manager tool integrated in SRS. It was observed that one of the participants experienced an issue with the installation of SRS due to which he did not find this process to be easy.

2. Create TCP IP Socket on the robot

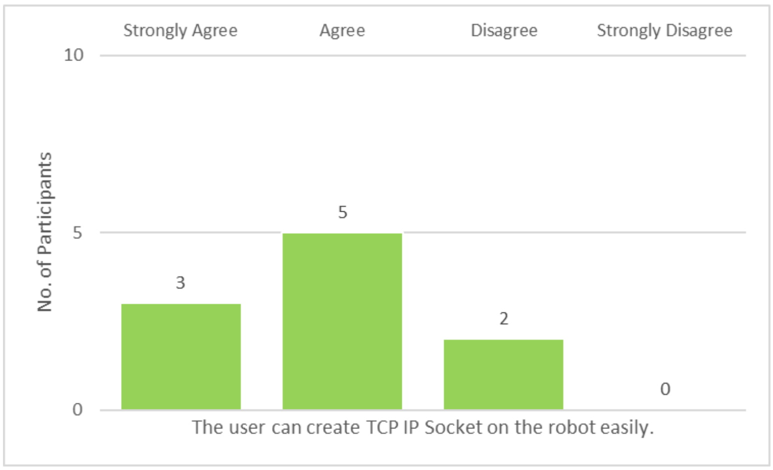


Figure 32

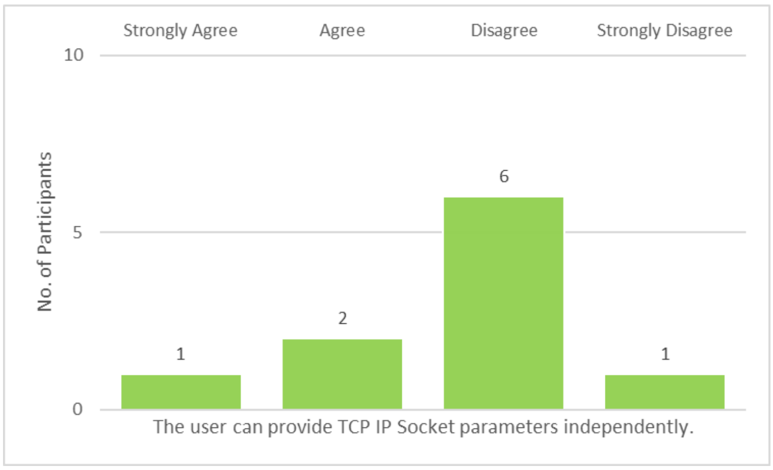


Figure 33

It was observed that 50% of the participants could easily create the TCP-IP socket on the robot whereas 60% of them found it difficult to insert the communication parameters.

3. Open, Run and Interact with the downloaded robot API from the HMI pendant

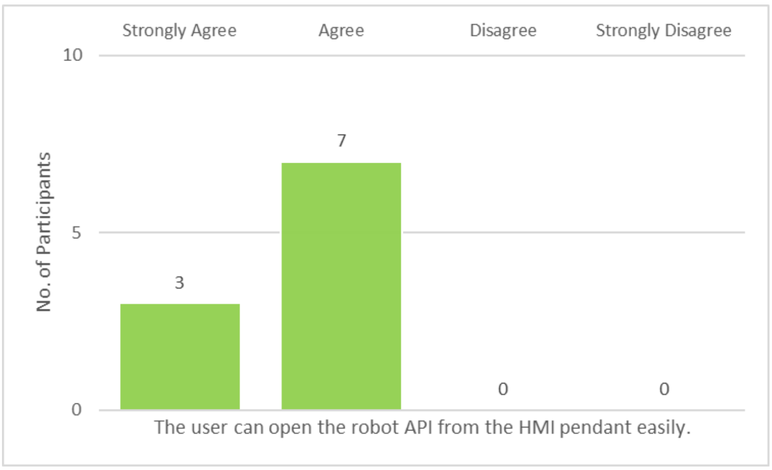


Figure 34

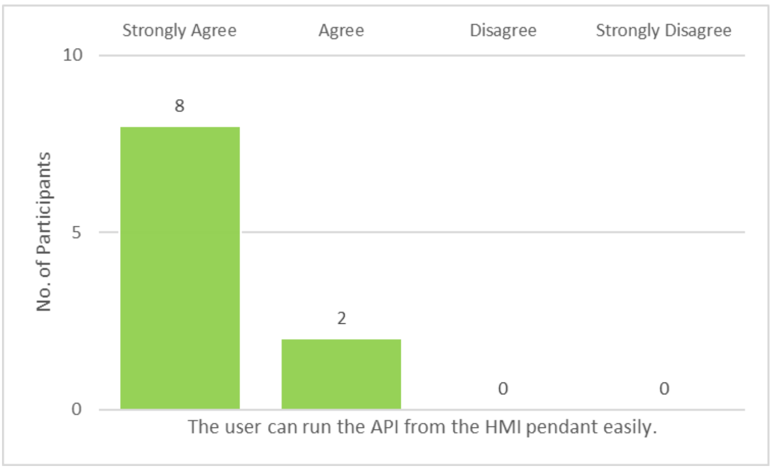


Figure 35

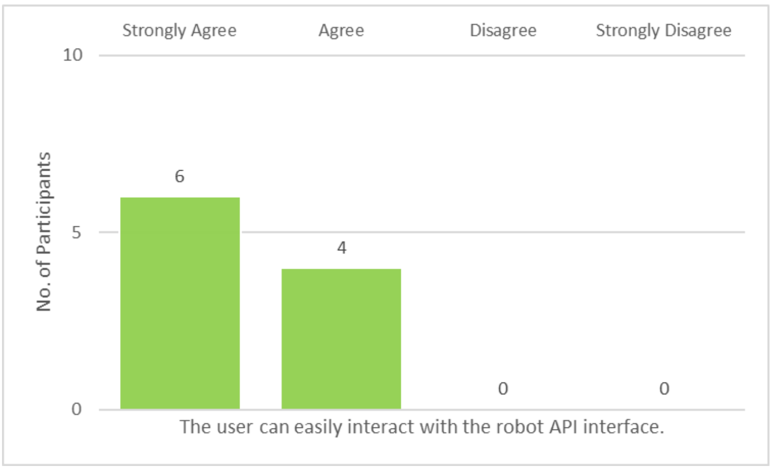


Figure 36

It was observed that 30% of the participants did not find it too easy to open and load the robot API into the RAM from the HMI pendant as it involves multiple operations to access the robot controller's disk from the HMI menu and choose the respective API from the application list. However, once the application is opened and loaded into the robot controller's RAM around 70% of the participants found is very easy to RUN it and interact with the API interface. The main reason behind this is that Running an application from the HMI pendant is a one button operation if the application is already loaded into the robot controller's RAM. The application's interface is simple to interact with as the user just needs to enter 1 as a numeric input in order to begin the communication process with the PRUDENS server.

4. Run PRUDENS server on the PC

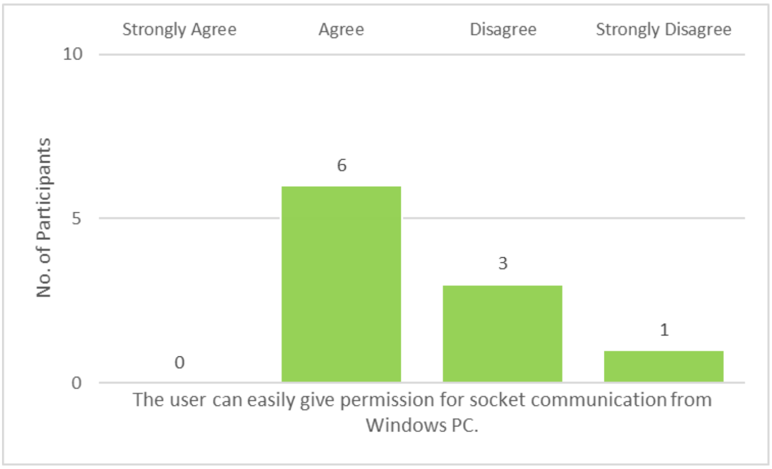


Figure 37

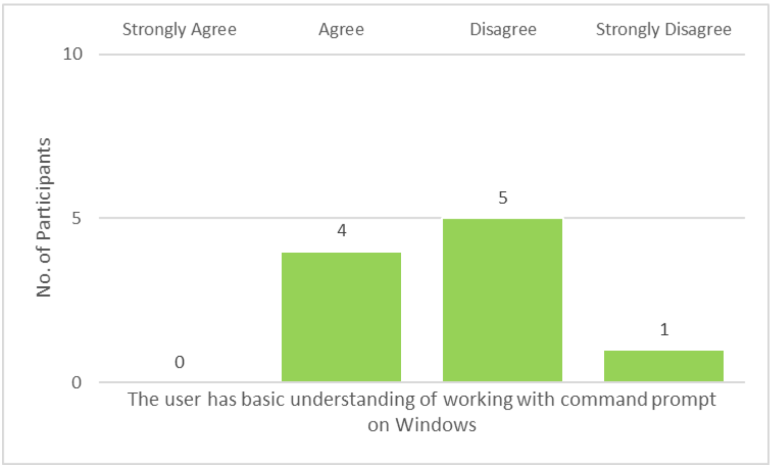


Figure 38

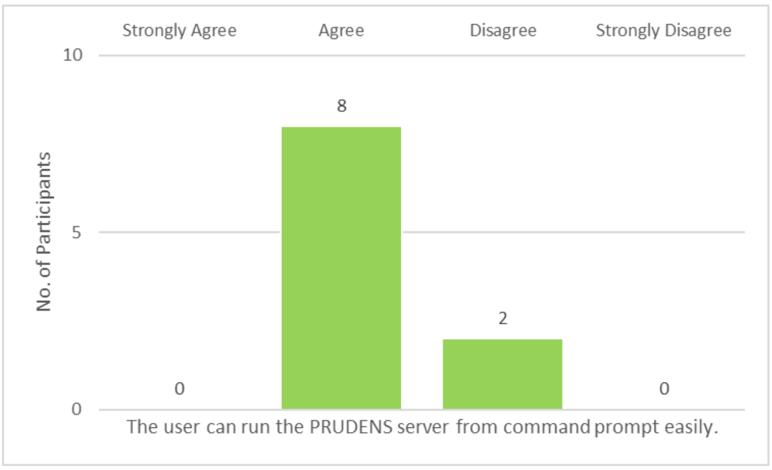


Figure 39

We observed that 30% of the participants found it difficult to give access rights from Windows for their PC to enable data exchange over TCP IP. One of the participants failed to do so due to some stringent access right policy in his company's laptop. This participant however successfully configured the permissions in his personal laptop. The remaining 60% participants were easily able to do so by following the respective cues on their Windows operating system. When it comes to working with the command prompt it was found that 50% of the participants had negligible or little understanding of this tool. However, 80% of the participants found it easy to run PRUDENS server from the command prompt by executing the "*npm start*" command. It was observer that 20% of the participants found it difficult to run PRUDENS server as they could not navigate to the source folder from the command prompt.

5. Workaround with the PRUDENS interface

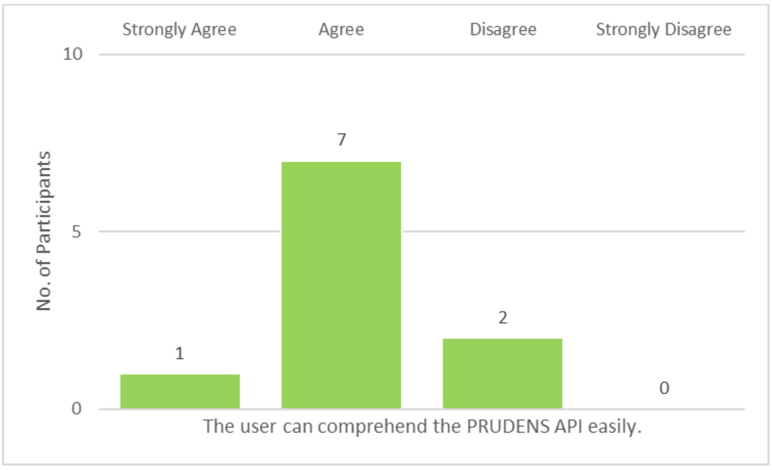


Figure 40

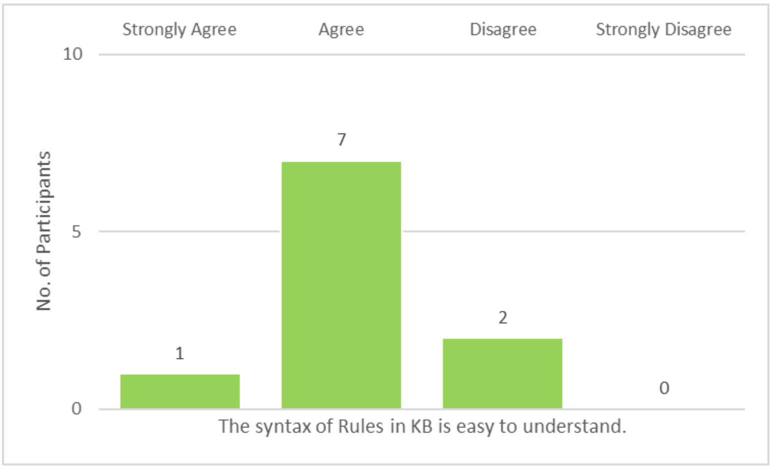


Figure 41

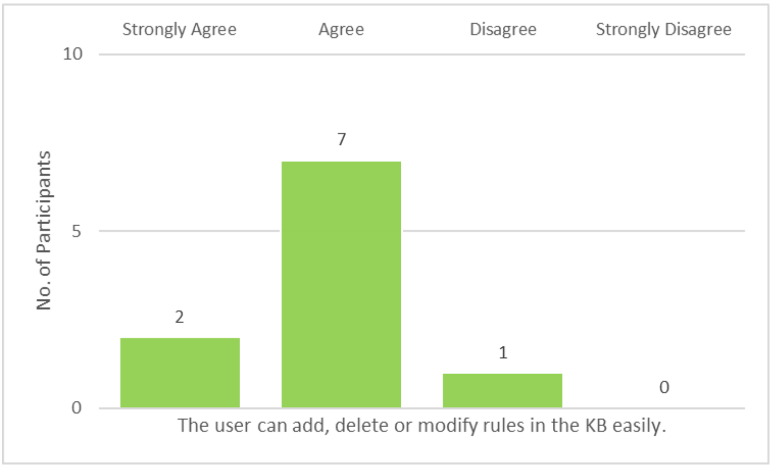


Figure 42

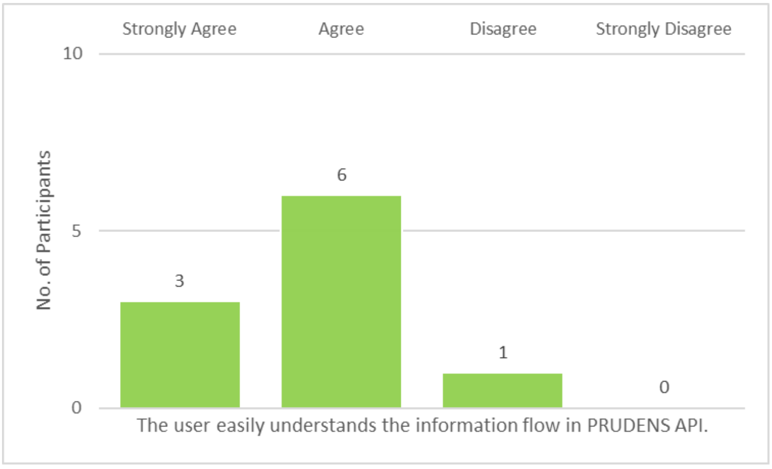


Figure 43

It was observed that 80% of the participants found it easy to comprehend the PRUDENS interface understand the syntax of Rules in the knowledge base. 20% of the participants however found it difficult to grasp the basics of the PRUDENS concept. It was also found that 90% of the participants were comfortable with the logical concept behind the rules and could alter the knowledge base easily.

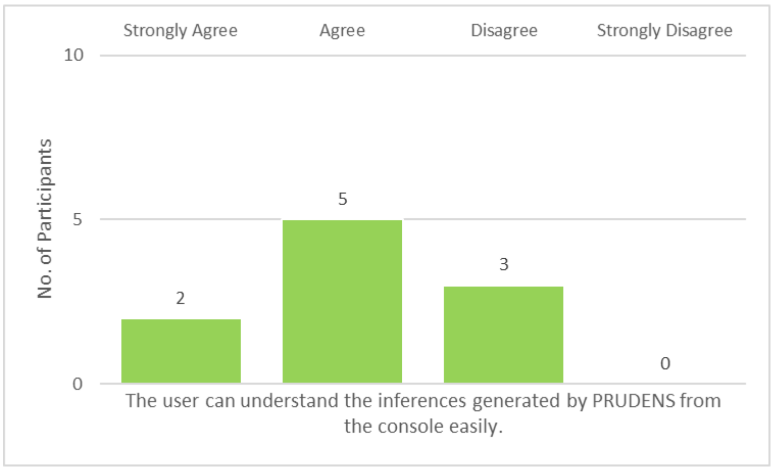


Figure 44

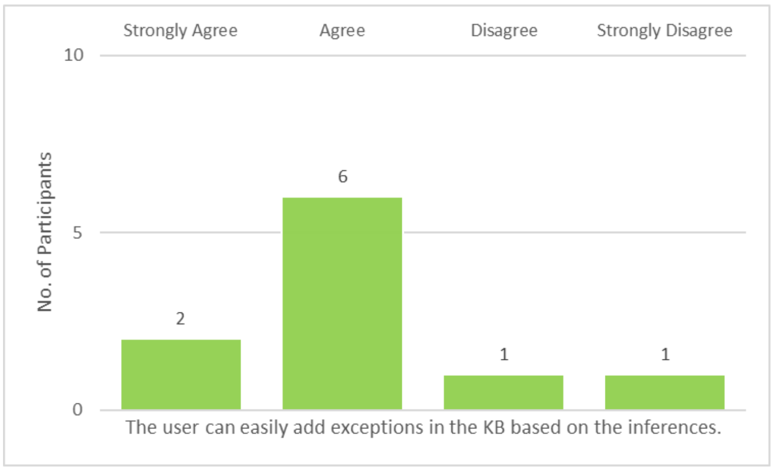


Figure 45

When it comes to understanding of the inferences generated by PRUDENS in the console, we found that 70% of the participants could easily understand them. However, 30% found it difficult initially to build an understanding of the inferences appearing on the console. We also found that 80% of the participants could successfully and easily create exceptions in the knowledge base. The remaining 20% of the participants found it difficult to create exceptions, if necessary, based on the inferences generated out of which one participant found it extremely difficult to work around with this.

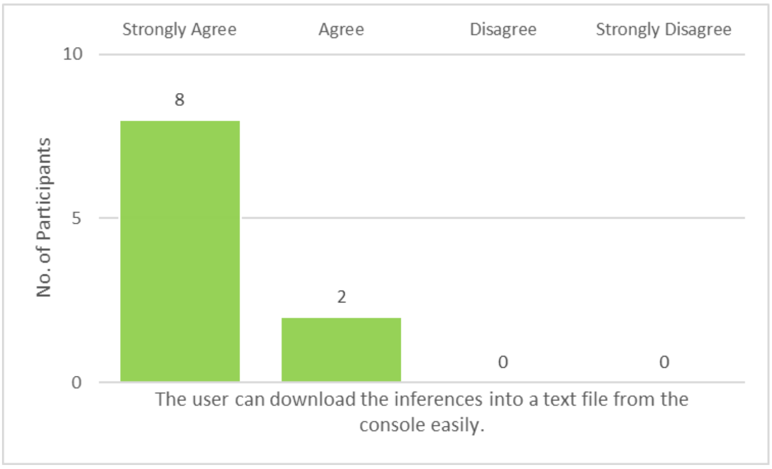


Figure 46

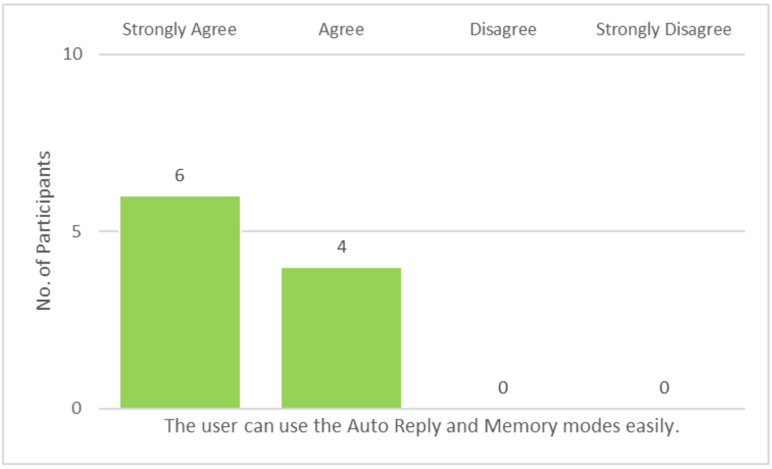


Figure 47

All of the participants could easily download the inferences in the form of a text file by using the Download button on the console and use the Auto Reply and Memory functionalities.

6. Control and Coach the Robot through PRUDENS

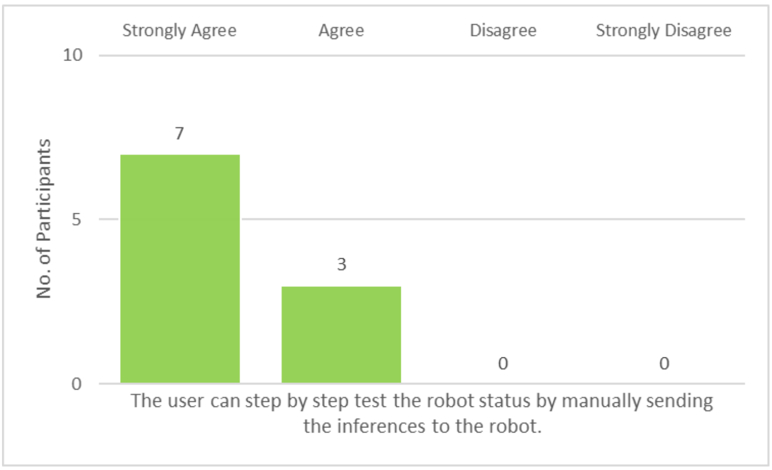


Figure 48

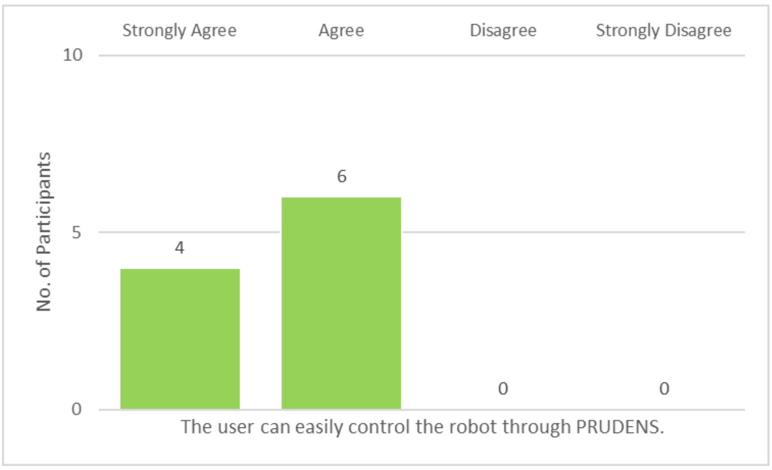


Figure 49

We observed that all the participants were comfortable in testing the robot by manually sending the outcomes generated by PRUDENS as part of the coaching scenario. 70% of the participants found this operation to be very easy to perform. Similar results were observed when the participants tried to control the robot's motion through PRUDENS

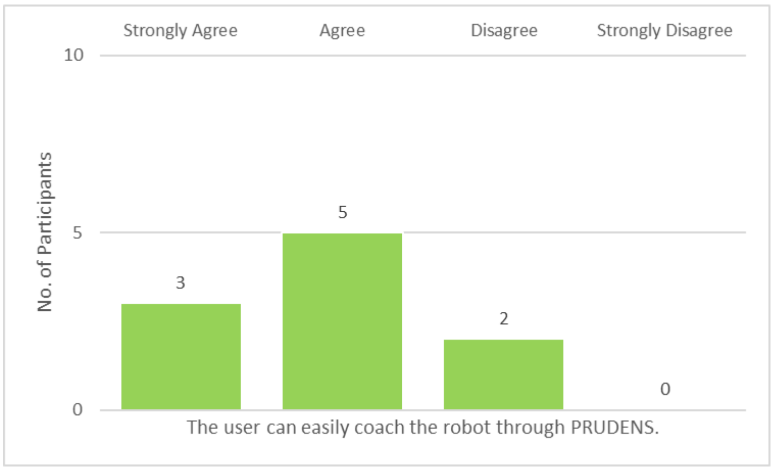


Figure 50

When it comes to a complete coaching scenario of the robotic arm through PRUDENS, it was found that 80% of the participants could easily perform this operation. Two of the participants were however got confused at various stages of coaching and therefore found the process to be difficult.

As the final evaluation criteria, the participants were asked to complete the questionnaire that determines the System Usability Scale (SUS) as part of their individual feedback. The questionnaire consists of 10 questions where the participants are required to record their feedback on 1 – 5 scale.

From Table 7 below, we observe the outlook of the participants on the questionnaire we created for SUS evaluation.

S. No.	User Feedback	Strongly Agree	Agree	Neither Agree Nor Disagree	Disagree	Strongly Disagree
1	**The proposed system was easy to use.**	**0**	**3**	**4**	**2**	**1**
2	**The proposed system was difficult to use.**	**0**	**2**	**6**	**1**	**1**
3	**I would prefer to use this system to coach**	**1**	**2**	**4**	**3**	**0**

	the robot frequently.					
4	**I cannot use this system independently.**	**1**	**4**	**2**	**2**	**1**
5	**This proposed system is not complete.**	**0**	**1**	**8**	**2**	**0**
6	**I need training to use this system.**	**2**	**5**	**2**	**1**	**0**
7	**The proposed system could have a better user interface.**	**0**	**3**	**5**	**2**	**0**
8	**I don't prefer working with TCP IP.**	**0**	**3**	**2**	**4**	**1**
9	**The proposed system cannot be implemented in complex robotic processes.**	**2**	**4**	**3**	**1**	**0**
10	**The proposed system is awful.**	**0**	**0**	**1**	**3**	**6**

Table 7. System Usability Scale Questionnaire

The first question was based on the ease of use of the system where out of ten participants four did not find it to be easy nor difficult whereas three of the participants found the system to be easy to use. Two participants found the system to be difficult probably due to their low industrial know-how whereas one participant found the system to be very difficult. The second question asked the participants about the system's difficulty level and it was found that two out of ten participants found the system to be difficult to use. However, six participants were not sure about the same. The remaining participants were comfortable with the system and found it not difficult at all. Based on their feedback on the third question we found that three participants would like to use such a system frequently to coach their robot out of which one participant looked very much promising in doing so. We also observe that another three participants were skeptical about using a similar system for robot coaching purposes. It might be possible that they belong to the category of the users who feel comfortable in using the standard technology rather than adopting an out of the box approach. The remaining four participants were not sure about using or not using a similar system for robot coaching purposes in their jobs. Regarding the fourth question about using a similar system independently, we observed that five out of ten participants were able to do so out of which one participant was highly confident regarding the same. Out of the remaining fifty percent participants two could not decide that whether they could use a similar system independently or not. However, three participants did not find it possible to use such a system independently with one being completely reluctant. This gave us an idea that the system might not target the users who are reluctant towards trying to get accustomed with new technology. This would further require us to add several features to the system that would gain the confidence of such users. Eight out of ten participants could not decide that whether the proposed system was actually ready to be adopted industrially or not. Although two participants were sure enough that the system needs to be developed further for a complete industrial integration one participant found it to be ready. We also agree that a complete industrial integration requires many factors to be considered. These factors might be related to several features for alarm and scenario management, reporting, safety etc. which could be adopted into our proposed system through a multi-dimensional research and analysis. Seventy percent of the participants found that they would require an initial training prior using such a system based on question six. Although the user interface of the system was quite simple, we observed that three out of ten participants desired for a better user interface which might be adaptable to the users of distinct cognitive traits with a possibility of personalization. However, fifty percent of the participants were not sure about the user interface. We also found that the remaining two participants found the user interface to be acceptable. As TCP -IP communication was one of the main features of the proposed system, we observed that according to the user feedback on question eight, fifty percent of the participants were comfortable with working with TCP-IP and thirty percent were against using this communication protocol. It might be due to their narrow experience in using TCP-IP communication within the industrial domain. However, the remaining twenty percent of the participants remained unsure about this. According to the user feedback on question nine, we found that sixty percent of the participants agreed that the proposed system cannot be implemented in complex robotic processes which might go beyond picking and placing applications such as laser cutting, welding, conveyor tracking etc. One participant however disagreed with the outlook of the majority. We on the other hand also disagree with the majority's feedback as laser cutting and welding applications are trajectory specific application where real time trajectory optimization is required. Trajectory optimization coaching could be easily facilitated with some modifications in the system. Also, concepts like product feed balancing, area distribution, conveyor strategy etc. that are widely used in robotic conveyor tracking applications could also be managed through our proposed system with some modifications and optimizations. From the user feedback related to the tenth question, we found that the majority of the participants did not find our system to be awful. This gives us motivation and strength to work on future developments related to a more sophisticated industrial integration and application specific optimization.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
P1	**3**	**5**	**3**	**3**	**5**	**4**	**3**	**2**	**3**	**2**	**52.5**
P2	**2**	**2**	**2**	**4**	**2**	**3**	**2**	**4**	**4**	**1**	**45**
P3	**4**	**5**	**3**	**4**	**2**	**4**	**2**	**3**	**2**	**1**	**40**
P4	**1**	**3**	**2**	**3**	**3**	**3**	**4**	**5**	**3**	**2**	**42.5**
P5	**3**	**2**	**5**	**1**	**3**	**4**	**4**	**2**	**2**	**2**	**65**
P6	**4**	**3**	**3**	**5**	**2**	**1**	**3**	**4**	**2**	**3**	**45**
P7	**5**	**2**	**5**	**3**	**4**	**3**	**5**	**3**	**4**	**2**	**75**
P8	**4**	**4**	**4**	**3**	**3**	**4**	**4**	**3**	**3**	**1**	**57.5**
P9	**5**	**3**	**4**	**3**	**3**	**3**	**3**	**4**	**5**	**2**	**62.5**
P10	**4**	**4**	**5**	**5**	**3**	**5**	**3**	**5**	**3**	**2**	**42.5**

Table 8. System Usability Scale Matrix

Based on the SUS questionnaire, we created a SUS matrix as seen in Table 8 above in order to calculate the system usability score of the proposed system. Here the ten participants are represented as Pn (n = 1, 2, 3…10) and the ten questions are represented as Qn(n = 1,2,3….10). We then asked the participants to grade the system based on the ten questions on a scale of 1-5. Based on the SUS score calculation, we found the maximum score to be 75, the minimum score to be 40 and the average score to be 52.75 out of 100.

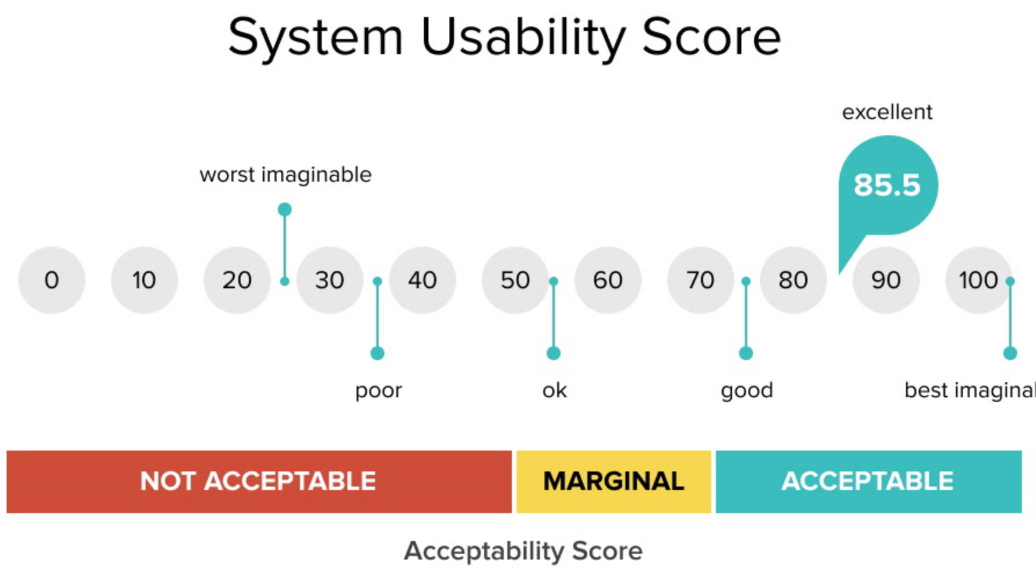


Figure 51. SUS Acceptability Score (Adobe 2021)

As per (Adobe 2021), we found that on an average, our proposed system lies on the marginal band which determines that there is a scope to research and develop the system further in terms of effectiveness, efficiency, and user satisfaction.

Chapter 6
Conclusion

6.1 Conclusion and Future Work

Indulging in an argument with a machine sounds fascinating, but it brings complexities that need to be handled at a broader level. During our integration of the PRUDENS argumentation framework with an industrial robotic arm, the main challenge was to design a communication framework that is machine manufacturer independent. Although we chose to integrate the system with a Staubli Robotic Arm, the framework that we created could also be integrated with any available industrial robot on the market that allows TCPIP socket communication. The communication framework adopted is the cheapest in the market as it is not brand specific, as is the case today where industrial automation brands want to stick with customized industrial communication protocols (Pereira & Neumann 2009), whose scope remains limited within their brand specific equipment.

Based on system trials with participants and analyzing the results, we were able to gain a detailed understanding of implementing such a system in an industrial setting and build a framework that would reduce programming complexities. Based on our findings and the discussion with the participants, we believe that we managed to explore the domain which requires such smart systems where the user could seek information from the machine and guide it through exceptions, thereby enhancing its performance. We also found that technological limitations and the bias that humans possess when it comes to using new technology play a major role in the design and implementation of such systems. Communication speeds are vital in a real-time system, therefore the protocol that verifies the data transferred and received in a communication cycle at high frequency is of utmost importance. TCP/IP assures an error-free data transfer between two parties where the data size determines the communication speed. Although TCP IP serves best for this purpose of information exchange, there are better technologies that could be adapted for the development of such systems. For instance, the entire system could be developed through the Robot Operation System (ROS), which would rule out the use of information exchange between two separate parties. This would have provided better performance and optimization capabilities. However, as the industrial robots are available with their inbuilt programming language that is widely preferred to program simple industrial applications without any extra programming cost, therefore, utilizing ROS technology would be expensive for our proposed system due to the requirement of several licenses and permissions from the robot manufacturer. But this technology would have made it easier to set up several automatic programming routines that aren't possible in the programming language that our robot uses by default.

Determining the naming style of the predicates that are used to create rules in the knowledge base of the argumentation framework was also found to be of critical importance. Therefore, we chose a style that is easy to interpret for the robot as well as for the human. We proceeded with the introduction of Boolean characteristics such as true and false in the predicate names, which makes it easy for the robotic application to interpret. In the initial phase of the development, we decided to proceed with bitwise operations such that the robot would send the contexts in the form of bits, which would be decoded at the PRUDENS server prior to processing for inference generation and subsequently convert the resultant into bits prior to transfer to the robot. Inference generation from a set of 30–40 rules is also time-consuming at the server side. Therefore, incorporating two additional parsers for bitwise operations would add to the processing time that would be further increased with the processing and decoding delays at the robot side, thereby increasing the overall delay in one send-receive cycle. It was also observed that using Booleans in the predicates made the entire naming criteria weird, which was also different from what PRUDENS suggests. We therefore moved to the standard naming criteria as suggested by PRUDENS and switched from bitwise operation to string operation at a later stage to keep the overall processing and comprehension times low. We also found that performing too many string operations also slows down the overall system performance. Therefore, we kept ourselves limited to minimal string operations on both the PRUDENS and the robot side. On the robot side, we developed contexts from its knowledge as strings that match the literals in the rules. This enabled a fast transfer of data by the robot, which was comprehended as context by the PRUDENS server without the need for any additional parser. We found that generating inferences is time-consuming as the size of the rules increases. Therefore, we created a cache memory function on the server side that mapped the incoming contexts to the generated inferences and stored the outcome in the cache memory. This enabled us to transfer the inferences as per the context from the cache memory without the need for reprocessing. In order to process the inferences received and generate an action at the robotic arm's side, we developed an inference parser. It was found that a complex parser's efficiency depends on the CPU and RAM capacity as it incorporates string operations which are time-consuming on slower CPUs. We experienced a delay in the parsing and execution operations on the robot side due to the low-capacity CPU and small RAM. In an overall scenario, we managed to work a complete cycle of context transfer and inference comprehension for non-cached outcomes to be as slow as 200 ms and for cached outcomes to be as slow as 138 ms. We also found that there was a requirement for a dynamic time synchronization setting for a transfer cycle at the robot side, which enables it to start in learning mode where the server does the respective mapping in the cache memory. After several cycles, the robot could be switched to non-learning mode where it now enables fast transfer cycles. From here we also found why Working Memory plays an important role in the entire Human Argumentation Model, which involves the complex cognitive operations of learning and reasoning, intelligence, etc.

This research might be extended into numerous areas, such as Human-Machine Interaction, Robotics, Automation, Learning through Argumentative Reasoning, and so on. There is a lot of room for research in areas like automated programming and program induction (Ellis and Gulwani, 2017) to achieve procedural knowledge, autonomous learning of arguments in a natural language environment, and improving machine-learning approaches to argumentation, among others. As we stand today on the verge of industry 5.0, we expect that the human workforce will be indulged more in machine knowledge enhancement rather than being physically involved in the tedious jobs which could be handled by machines or robots more efficiently. Focus would be on the human-machine interaction paradigm where the conveyance of knowledge, commands, and feedback would match that of human-human conversation. Great progress has already been made in the natural language interpretation modules of the voice-controlled home automation systems like Siri, Alexa, and Google Assistant, etc. Our idea of coaching an industrial robotic arm through argumentation is the initial step of bringing forward the use of argumentation theory as an interaction interface with industrial machines in an industrial setting, where we also bring forward the concept of machine programming through argumentation at a very superficial level. We believe that further developments in our approach to machine coaching would make industries ready for the upcoming industrial revolutions.

Appendix A
Code

Robot Coaching through PRUDENS

The code for our proposed system is available at the following GitHub repository.

https://github.com/khan4search/Robot-Coaching-through-PRUDENS.git

References

Adadi, A., AND Berrada, M. (2018) Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. https://doi.org/10.1109/access.2018.2870052

Adobe. (2021). The System Usability Scale & How it's Used in UX | Adobe XD. Ideas. https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/

Akdogan, K.E. (2019) Introduction to Industrial Robot Programming. https://mece104.cankaya.edu.tr/uploads/files/Introduction%20to%20Industrial%20Robot%20Programming.pdf.

Amgoud, L., & Serrurier, M. (2007). Agents that argue and explain classifications. Autonomous Agents and Multi-Agent Systems, 16(2), 187–209. https://doi.org/10.1007/s10458-007-9025-6

Arrieta, A., B., et al. (2020), Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward responsible AI, Information Fusion (58), 82-115.

Aroyo, A. M., Rea, F., Sandini, G., & Sciutti, A. (2018). Trust and Social Engineering in Human Robot Interaction: Will a Robot Make You Disclose Sensitive Information, Conform to Its Recommendations or Gamble? IEEE Robotics and Automation Letters, 3(4), 3701–3708. https://doi.org/10.1109/lra.2018.2856272

Azhar, M. Q., & Sklar, E. I. (2017). A study measuring the impact of shared decision making in a human-robot team. The International Journal of Robotics Research, 36(5–7), 461–482. https://doi.org/10.1177/0278364917710540

Bergamini, L., Sposato, M., Pellicciari, M., Peruzzini, M., Calderara, S., & Schmidt, J. (2020). Deep learning-based method for vision-guided robotic grasping of unknown objects. Advanced Engineering Informatics, 44, 101052. https://doi.org/10.1016/j.aei.2020.101052

Bernardo, B., Alves-Oliveira, P., Santos, M. G., Melo, F. S., & Paiva, A. (2016). An Interactive Tangram Game for Children with Autism. Intelligent Virtual Agents, 500–504. https://doi.org/10.1007/978-3-319-47665-0_63

Black, E., & Sklar, E. I. (2016). Computational argumentation to support multi-party human-robot interaction: Challenges and advantages. In *Proceedings of the Groups in Human-Robot Interaction Workshop: A workshop at the IEEE International Symposium on Robot and Human Interactive Communication.*

Bratko, I., Žabkar, J., Možina, M. (2009). Argument-Based Machine Learning. In: Simari, G., Rahwan, I. (eds) Argumentation in Artificial Intelligence. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-98197-0_23

Breiman, L. (2001). Random forests. Machine Learning 45(1), 5–32

Buizza Avanzini, G., Ceriani, N. M., Zanchettin, A. M., Rocco, P., & Bascetta, L. (2014). Safety Control of Industrial Robots Based on a Distributed Distance Sensor. IEEE Transactions on Control Systems Technology, 22(6), 2127–2140. https://doi.org/10.1109/tcst.2014.2300696

Canal, G., Escalera, S., & Angulo, C. (2016). A real-time Human-Robot Interaction system based on gestures for assistive scenarios. Computer Vision and Image Understanding, 149, 65–77. https://doi.org/10.1016/j.cviu.2016.03.004

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, 4(6), 759–771. https://doi.org/10.1016/0893-6080(91)90056-b

Carstens, L., Toni, F. (2015) Improving out-of-domain sentiment polarity classification using argumentation. In: IEEE International Conference on Data Mining Workshop, ICDMW. pp. 1294–1301.

Chen, F., Lv, H., Pang, Z., Zhang, J., Hou, Y., Gu, Y., Yang, H., & Yang, G. (2019). WristCam: A Wearable Sensor for Hand Trajectory Gesture Recognition and Intelligent Human–Robot Interaction. IEEE Sensors Journal, 19(19), 8441–8451. https://doi.org/10.1109/jsen.2018.2877978

Chen, L., Su, W., Feng, Y., Wu, M., She, J., & Hirota, K. (2020). Two-layer fuzzy multiple random forest for speech emotion recognition in human-robot interaction. Information Sciences, 509, 150–163. https://doi.org/10.1016/j.ins.2019.09.005

Chen, X., Wang, N., Cheng, H., & Yang, C. (2020). Neural Learning Enhanced Variable Admittance Control for Human–Robot Collaboration. IEEE Access, 8, 25727–25737. https://doi.org/10.1109/access.2020.2969085

Cherubini, A., Passama, R., Fraisse, P., & Crosnier, A. (2015). A unified multimodal control framework for human–robot interaction. Robotics and Autonomous Systems, 70, 106–115. https://doi.org/10.1016/j.robot.2015.03.002

Cid, F., Prado, J. A., Bustos, P., & Nunez, P. (2013). A real time and robust facial expression recognition and imitation approach for affective human-robot interaction using Gabor filtering. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. https://doi.org/10.1109/iros.2013.6696662

Clark, P., Niblett, T. (1989) The CN2 induction algorithm, Machine Learning Journal 4 (3) 261–283.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. https://doi.org/10.1007/bf00994018

Craven, R., & Toni, F. (2016). Argument graphs and assumption-based argumentation. Artificial Intelligence, 233, 1–59. https://doi.org/10.1016/j.artint.2015.12.004

Ding, I. J., & Shi, J. Y. (2017). Kinect microphone array-based speech and speaker recognition for the exhibition control of humanoid robots. Computers & Electrical Engineering, 62, 719–729. https://doi.org/10.1016/j.compeleceng.2015.12.010

Doering, M., Glas, D. F., & Ishiguro, H. (2019). Modeling Interaction Structure for Robot Imitation Learning of Human Social Behavior. IEEE Transactions on Human-Machine Systems, 49(3), 219–231. https://doi.org/10.1109/thms.2019.2895753

Du, G., Chen, M., Liu, C., Zhang, B., & Zhang, P. (2018). Online Robot Teaching With Natural Human–Robot Interaction. IEEE Transactions on Industrial Electronics, 65(12), 9571–9581. https://doi.org/10.1109/tie.2018.2823667

Dung, P., M. (1995), On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games, Artificial Intelligence 77, 321-357.

Ellis, K., & Gulwani, S. (2017) Learning to learn programs from examples: Going beyond program structure. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. https://doi.org/10.24963/ijcai.2017/227

Erden, M. S., & Billard, A. (2014). End-point impedance measurements at human hand during interactive manual welding with robot. 2014 IEEE International Conference on Robotics and Automation (ICRA). https://doi.org/10.1109/icra.2014.6906599

Erden, M. S., & Billard, A. (2015a). End-Point Impedance Measurements Across Dominant and Nondominant Hands and Robotic Assistance with Directional Damping. IEEE Transactions on Cybernetics, 45(6), 1146–1157. https://doi.org/10.1109/tcyb.2014.2346021

Erden, M. S., & Billard, A. (2015b). Hand Impedance Measurements During Interactive Manual Welding With a Robot. IEEE Transactions on Robotics, 31(1), 168–179. https://doi.org/10.1109/tro.2014.2385212

Fang, B., Ma, X., Wang, J., & Sun, F. (2020). Vision-based posture-consistent teleoperation of robotic arm using multi-stage deep neural network. Robotics and Autonomous Systems, 131, 103592. https://doi.org/10.1016/j.robot.2020.103592

Ferretti, E., Errecalde, M., Garcıa, A. and Simari, G. (2006). Khedelp: A framework to support defeasible logic programming for the khepera robots, ISRA06.

Ferretti, E., Errecalde, M., García, A.J. and Simari, G.R. (2007). An application of defeasible logic programming to decision making in a robotic environment, Logic Programming and Nonmonotonic Reasoning, Springer, pp.297–302.

Ficuciello, F., Villani, L., & Siciliano, B. (2015). Variable Impedance Control of Redundant Manipulators for Intuitive Human–Robot Physical Interaction. IEEE Transactions on Robotics, 31(4), 850–863. https://doi.org/10.1109/tro.2015.2430053

Fujii, K., Gras, G., Salerno, A., & Yang, G. Z. (2018). Gaze gesture based human robot interaction for laparoscopic surgery. Medical Image Analysis, 44, 196–214. https://doi.org/10.1016/j.media.2017.11.011

Gaertner & Toni, F. (2007). Computing arguments and attacks in assumption-based argumentation, IEEE Intelligent Systems 22(6) (2007), 24–33.

Gamboa-Montero, J. J., Alonso-Martín, F., Castillo, J. C., Malfaz, M., & Salichs, M. A. (2020). Detecting, locating and recognising human touches in social robots with contact microphones. Engineering Applications of Artificial Intelligence, 92, 103670. https://doi.org/10.1016/j.engappai.2020.103670

Gao, Q., Liu, J., Ju, Z., & Zhang, X. (2019). Dual-Hand Detection for Human–Robot Interaction by a Parallel Network Based on Hand Detection and Body Pose Estimation. IEEE Transactions on Industrial Electronics, 66(12), 9663–9672. https://doi.org/10.1109/tie.2019.2898624

Gao, Y., Toni, F. (2013). Argumentation accelerated reinforcement learning for robocup keepaway-takeaway. In: Theory and Applications of Formal Argumentation - Second International Workshop, TAFA. vol. 8306, pp. 79–94

Gao, Y., Toni, F. (2014). Argumentation accelerated reinforcement learning for cooperative multi-agent systems. In: ECAI 2014 - 21st European Conference on Artificial Intelligence. pp. 333–338

Gao, Y., Toni, F. (2015) Argumentation accelerated reinforcement learning. Ph.D. thesis, Imperial College London

García, A. J. and Simari, G. R. (2004). Defeasible logic programming: An argumentative approach. In Theory and practice of logic programming (Vol. 4, pp. 95–138). Cambridge University Press.

Ge, S. S., Wang, C., & Hang, C. C. (2008). Facial expression imitation in human robot interaction. RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication. https://doi.org/10.1109/roman.2008.4600668

Gómez, S.A., Chesnevar, C.I. (2004). A hybrid approach to pattern classification using neural networks and defeasible argumentation. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA. pp. 393–398

Governatori, G. (2004). Defeasible Description Logic. 10.1007/978-3-540-30504-0_8.

Governatori, G. and Terenziani, P. (2007). Temporal extensions to defeasible logic. In Proceedings of the 20th Australian joint conference on Advances in artificial intelligence (AI'07). Springer-Verlag, Berlin, Heidelberg, 476–485.

Grosse, K., González, M. P., Chesñevar, C. I., & Maguitman, A. G. (2015). Integrating argumentation and sentiment analysis for mining opinions from Twitter. AI Communications, 28(3), 387–401. https://doi.org/10.3233/aic-140627

Gunawan, A. A. S., Stevelino, A., Ngarianto, H., Budiharto, W., & Wongso, R. (2017). Implementation of Blind Speech Separation for Intelligent Humanoid Robot using DUET Method. Procedia Computer Science, 116, 87–98. https://doi.org/10.1016/j.procs.2017.10.014

Hierons, R. (1999). Machine learning. Tom M. Mitchell. Published by McGraw-Hill, Maidenhead, U.K., International Student Edition, 1997. ISBN: 0–07-115467-1, 414 pages. Price: U.K. £22.99, soft cover. Software Testing, Verification and Reliability, 9(3), 191–193.

Hong, J. H., Song, Y. S., & Cho, S. B. (2007). Mixed-Initiative Human–Robot Interaction Using Hierarchical Bayesian Networks. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 37(6), 1158–1164. https://doi.org/10.1109/tsmca.2007.906570

IFR International Federation of Robotics. (2020). IFR presents World Robotics Report 2020. https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe

Jensen, B., Tomatis, N., Mayor, L., Drygajlo, A., & Siegwart, R. (2005). Robots Meet Humans—Interaction in Public Spaces. IEEE Transactions on Industrial Electronics, 52(6), 1530–1546. https://doi.org/10.1109/tie.2005.858730

John, G.H., Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338–345. UAI'95

Kim, Y. M., Koo, S. Y., Lim, J., & Kwon, D. S. (2010). A robust online touch pattern recognition for dynamic human-robot interaction. IEEE Transactions on Consumer Electronics, 56(3), 1979–1987. https://doi.org/10.1109/tce.2010.5606355

Kronander, K., & Billard, A. (2014). Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction. IEEE Transactions on Haptics, 7(3), 367–380. https://doi.org/10.1109/toh.2013.54

Kurose, J.F., & Ross, K.W. (2013). Chapter 2: Application Layer. In Computer networking: A top-down approach 6th ed (pp. 156–168). essay, PEARSON.

Lam, H. P., & Governatori, G. (2012). Towards a model of UAVs navigation in urban canyon through defeasible logic. Journal of Logic and Computation, 23(2), 373–395. https://doi.org/10.1093/logcom/exr028

Landi, C. T., Cheng, Y., Ferraguti, F., Bonfe, M., Secchi, C., & Tomizuka, M. (2019). Prediction of Human Arm Target for Robot Reaching Movements. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). https://doi.org/10.1109/iros40897.2019.8968559. Languages for the Semantic Web, Springer, pp.98–112.

Li, X. (2020). Human–robot interaction based on gesture and movement recognition. Signal Processing: Image Communication, 81, 115686. https://doi.org/10.1016/j.image.2019.115686

Lippi, M., & Torroni, P. (2016). Argumentation Mining. ACM Transactions on Internet Technology, 16(2), 1–25. https://doi.org/10.1145/2850417

Liu, H., & Wang, L. (2021). Collision-free human-robot collaboration based on context awareness. Robotics and Computer-Integrated Manufacturing, 67, 101997. https://doi.org/10.1016/j.rcim.2020.101997

Liu, Z. T., Xie, Q., Wu, M., Cao, W. H., Mei, Y., & Mao, J. W. (2018). Speech emotion recognition based on an improved brain emotion learning model. Neurocomputing, 309, 145–156. https://doi.org/10.1016/j.neucom.2018.05.005 logic, AI 2007: Advances in Artificial Intelligence, Springer, pp.476–485.

Markos, V. (nd). Prudens JS . GitHub. Retrieved February 12, 2022, from https://vmarkos.github.io/prudens-js/docs.html#The-language-of-Prudens

McColl, D., Jiang, C., & Nejat, G. (2017). Classifying a Person's Degree of Accessibility From Natural Body Language During Social Human–Robot Interactions. IEEE Transactions on Cybernetics, 47(2), 524–538. https://doi.org/10.1109/tcyb.2016.2520367

Meattini, R., Benatti, S., Scarcia, U., de Gregorio, D., Benini, L., & Melchiorri, C. (2018). An sEMG-Based Human–Robot Interface for Robotic Hands Using Machine Learning and Synergies. IEEE Transactions on Components, Packaging and Manufacturing Technology, 8(7), 1149–1158. https://doi.org/10.1109/tcpmt.2018.2799987

Melo, F. S., Sardinha, A., Belo, D., Couto, M., Faria, M., Farias, A., Gambôa, H., Jesus, C., Kinarullathil, M., Lima, P., Luz, L., Mateus, A., Melo, I., Moreno, P., Osório, D., Paiva, A., Pimentel, J., Rodrigues, J., Sequeira, P., Ventura, R. (2019). Project INSIDE: towards autonomous semi-unstructured human–robot social interaction in autism therapy. Artificial Intelligence in Medicine, 96, 198–216. https://doi.org/10.1016/j.artmed.2018.12.003

Michael, L. (2019) "Machine Coaching". IJCAI 2019 Workshop on Explainable Artificial Intelligence (XAI @ IJCAI 2019).

Možina, M., ŽAbkar, J., & Bratko, I. (2007). Argument based machine learning. Artificial Intelligence, 171(10–15), 922–937. https://doi.org/10.1016/j.artint.2007.04.007

Nute, D. (1994). Defeasible logic. In Handbook of Logic in Artificial Intelligence and Logic Programming: Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning (Vol. 3, pp. 353–395). Clarendon Press.

Oliveira, J., Ferreira, A., & Reis, J. C. (2020). Design and experiments on an inflatable link robot with a built-in vision sensor. Mechatronics, 65, 102305. https://doi.org/10.1016/j.mechatronics.2019.102305

Ontañón, S., & Plaza, E. (2014). Coordinated inductive learning using argumentation-based communication. Autonomous Agents and Multi-Agent Systems, 29(2), 266–304. https://doi.org/10.1007/s10458-014-9256-2

Ontañón, S., Dellunde, P., Godo, L., & Plaza, E. (2012). A defeasible reasoning model of inductive concept learning from examples and communication. Artificial Intelligence, 193, 129–148. https://doi.org/10.1016/j.artint.2012.08.006

Pereira C.E., Neumann P. (2009) Industrial Communication Protocols. In: Nof S. (eds) Springer Handbook of Automation. Springer Handbooks. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78831-7_56

Perikos, I., Markos, V., and Michael, L. (2020). D3.1 Basic Social Relation Learning Module (Ref. Ares(2020)2299834-29/04/2020). WeNet Consortium.

Peternel, L., Tsagarakis, N., & Ajoudani, A. (2017). A Human–Robot Co-Manipulation Approach Based on Human Sensorimotor Information. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 25(7), 811–822. https://doi.org/10.1109/tnsre.2017.2694553

Prakken, H. (2010), An Abstract Framework for Argumentation with Structured Arguments, Argument & Computation, 1:2, 93-124.

Qureshi, A. H., Nakamura, Y., Yoshikawa, Y., & Ishiguro, H. (2018). Intrinsically motivated reinforcement learning for human–robot interaction in the real-world. Neural Networks, 107, 23–33. https://doi.org/10.1016/j.neunet.2018.03.014

Raiola, G., Cardenas, C. A., Tadele, T. S., de Vries, T., & Stramigioli, S. (2018). Development of a Safety- and Energy-Aware Impedance Controller for Collaborative Robots. IEEE Robotics and Automation Letters, 3(2), 1237–1244. https://doi.org/10.1109/lra.2018.2795639

Robla-Gomez, S., Becerra, V. M., Llata, J. R., Gonzalez-Sarabia, E., Torre-Ferrero, C., & Perez-Oria, J. (2017). Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments. IEEE Access, 5, 26754–26773. https://doi.org/10.1109/access.2017.2773127

Rozo, L., Calinon, S., Caldwell, D. G., Jimenez, P., & Torras, C. (2016). Learning Physical Collaborative Robot Behaviors From Human Demonstrations. IEEE Transactions on Robotics, 32(3), 513–527. https://doi.org/10.1109/tro.2016.2540623

Rummery, G.A., Niranjan, M. (1994). On-line Q-learning using connectionist systems. Tech. rep., Cambridge University Engineering Department

Shiomi, M., Shatani, K., Minato, T., & Ishiguro, H. (2018). How Should a Robot React Before People's Touch?: Modeling a Pre-Touch Reaction Distance for a Robot's Face. IEEE Robotics and Automation Letters, 3(4), 3773–3780. https://doi.org/10.1109/lra.2018.2856303

Sklar, E. I., & Azhar, M. Q. (2015). Argumentation-Based Dialogue Games for Shared Control in Human-Robot Systems. Journal of Human-Robot Interaction, 4(3), 120. https://doi.org/10.5898/jhri.4.3.sklar

Sklar, E., Azhar, M. Q., Flyr, T., & Parsons, S. (2013). Enabling human-robot collaboration via argumentation. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (pp. 1251-1252).

Stenning, K., & Lambalgen, V. M. (2012). Human Reasoning and Cognitive Science (A Bradford Book) (1st ed.). MIT Press.

Toulmin, S. E. (2003). The Uses of Argument (Updated ed.). Cambridge University Press, 89.

Tseng, S. H., Chao, Y., Lin, C., & Fu, L. C. (2016). Service robots: System design for tracking people through data fusion and initiating interaction with the human group by inferring social situations. Robotics and Autonomous Systems, 83, 188–202. https://doi.org/10.1016/j.robot.2016.05.004

Valiant, L., G. (1984), A Theory of the Learnable, Communications of the ACM 27 (11), 1134-1142.

Wallén, J. (2008). The history of the industrial robot (Report No. LiTH-ISY-R-2853). Department of Electrical Engineering, Linköpings universitet, Sweden. http://www.control.isy.liu.se/publications/?type=techreport&number=2853&go=Search&output=html

Xue, K., Wang, Z., Shen, J., Hu, S., Zhen, Y., Liu, J., Wu, D., & Yang, H. (2021). Robotic seam tracking system based on vision sensing and human-machine interaction for multi-pass MAG welding. Journal of Manufacturing Processes, 63, 48–59. https://doi.org/10.1016/j.jmapro.2020.02.026

Yongda, D., Fang, L., & Huang, X. (2018). Research on multimodal human-robot interaction based on speech and gesture. Computers & Electrical Engineering, 72, 443–454. https://doi.org/10.1016/j.compeleceng.2018.09.014

Zheng, X., Shiomi, M., Minato, T., & Ishiguro, H. (2020). What Kinds of Robot's Touch Will Match Expressed Emotions? IEEE Robotics and Automation Letters, 5(1), 127–134. https://doi.org/10.1109/lra.2019.2947010

Zhou, H., Yang, G., Lv, H., Huang, X., Yang, H., & Pang, Z. (2020). IoT-Enabled Dual-Arm Motion Capture and Mapping for Telerobotics in Home Care. IEEE Journal of Biomedical and Health Informatics, 24(6), 1541–1549. https://doi.org/10.1109/jbhi.2019.2953885