Solving problems using the General Environment Description Language

Nina Bąkowska [1] Krzysztof Zatwarnicki [2]

[1] *Opole University of Technology, Proszkowska 76, 45-758 Opole, Poland*
[2] *Opole University of Technology, Proszkowska 76, 45-758 Opole, Poland*

Abstract

The history of automated systems demonstrates their wide-ranging impact on industries. Artificial intelligence has potential for task automation without human oversight, but currently "narrow AI" focuses on specific tasks rather than general intelligence. Versatile systems that can learn autonomously and adapt to various environments are essential for technological progress. This can only be achieved with accurate environment description. The General Environment Description Language (GEDL) emerged as the most promising solution for conceptualizing agent and robot environments. The notation is inspired by human knowledge organization, with an emphasis on learning from errors and improving efficiency. GEDL's definition of conceptual systems, occurrences, and experiences makes it an effective problem-solving tool. The study's focus centers on developing an algorithm with the capability to solve a range of issues for general-purpose systems. A proposed solution takes the form of a universal approach to problem-solving with available resources. The outcome of implementing the algorithm to accomplish a basic task, such as navigating a board, is outlines. A procedure for improving the execution is also analyzed.

Keywords

general purpose systems, environment description language, algorithm, automatic systems, knowledge representation, conceptual system, problem solving

1. Introduction

Although it may seem strange, automated systems that can operate without human supervision have been in existence and in use for over two thousand years. These systems were surprisingly simple in design, and included items such as water clocks and animal traps [1]. Autonomous systems have had a significant impact on technological progress, completely transforming various sectors and enhancing both efficiency and productivity.

The evolution of robotics represented a significant milestone in the domain of automated systems. The first industrial robots, capable of performing repetitive tasks with programmed precision, were introduced during the 1960s and 1970s. Initially, these devices found utility in manufacturing, particularly within the domain of automotive assembly processes. Over time, robotics technology has evolved, resulting in the creation of increasingly adaptable and intricate robots designed for various industries and applications.

The significant impact that automated systems have had on technological progress is widely recognized. However, in modern times, attention has shifted towards Artificial Intelligence (AI) as a solution capable of performing tasks with minimal supervision. AI can be defined as the use of technology to automate activities that typically require human intelligence. This distinction emphasizes that AI often focuses on automating particular task categories - those which are commonly considered

Proceedings ITTAP'2023: 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, November 22–24, 2023, Ternopil, Ukraine, Opole, Poland
EMAIL: bakowskanina1704@gmail.com (A. 1); k.zatwarnicki@po.opole.pl (A. 2)
ORCID: 0009-0000-7719-2379 (A. 1); 0000-0001-6156-6030 (A. 2)

to require human intelligence during their execution. The capabilities of AI-driven systems are rapidly expanding, creating new opportunities for exploration.

Currently, intricate systems are precisely customized to support specific tasks, serving a restricted range of utilities. These systems mainly utilize deep learning techniques, resulting in outstanding achievements like accurate image identification or creating artificial visual and written materials. The most effective and widespread version is known as narrow AI, which refers to a particular type of AI that requires developing a learning algorithm to accomplish a single task, and the gained insights are not inherently practical to other tasks. This indicates that the primary focus in this area is on designing programs that manifest intelligence in a specialized area. Although "narrow AI" solutions achieve impressive results within their assigned domains, they demonstrate restricted adaptability to challenges outside their scope.

Despite current technological advancements, which have made it possible to create intelligent, adaptable, and self-guided systems, there has been no significant breakthrough in the past 20 years to push forward the growth of generalized AI.

Although progress has been made with systems that are highly specialised for particular domains, the current focus should be on general-purpose systems that are able to learn about different environments autonomously and perform tasks in different contexts. The development of general-purpose systems is still out of reach because there is no solution that can comprehensively tackle multifaceted challenges and integrate knowledge from various environments. There is an expectation that solutions based on "narrow AI" will serve as the foundation for building these systems because of their ability to accurately interpret accumulated data.

Developing algorithms that can solve complex problems is essential to making general-purpose systems efficient and practical. However, the algorithms utilised in current solutions are inadequate for this undertaking as they possess a limited focus, rendering it difficult to tackle a broad spectrum of tasks and possibilities.

2. Related work

The execution of tasks without human oversight extends beyond just general-purpose systems. Various endeavors have been made to define different types of environments, which can be quite constrained, in order to enable automation and decision-making. Nonetheless, there is increasing interest in the development of Artificial General Intelligence (AGI) that can address a broad range of tasks [2].

The development of narrow AI has not led to significant progress towards the goals of AGI. The creation of artificial general intelligence involves distinct concepts and methodologies compared to the development of more specialised, narrow AI systems. Various theoretical frameworks emphasise the significant distinctions between AGI and narrow AI; however, none of these frameworks have comprehensive empirical evidence to support them [3].

Along with the new possibilities, it was acknowledged that capturing statistical patterns is inadequate for interacting with humans. This presents a challenge for automatic systems tasked with answering questions or engaging in conversation. A multi-agent learning environment was proposed as a solution, along with corresponding learning techniques that facilitate the emergence of a foundational compositional language [4]. This language consists of sequences of abstract discrete symbols communicated by agents over time, while maintaining a coherent structure characterised by a distinct vocabulary and syntax.

There are numerous requirements for efficient utilisation of AGI, resulting in the creation of specialised systems designed for specific tasks, such as playing games. The Game Description Language (GDL) is a comprehensive knowledge representation framework specifically developed to formalise the rules of any game [5].

However, GDL is not the exclusive solution for describing the essential rules necessary for playing a game. An illustration of contrasting descriptive terminology is present in the frequently employed Unity game engine. UnityVGDL is an extended framework of Video Game Description Language (VGDL) [6]. The UnityVGDL framework presents fresh opportunities for General Video Game Playing competitions, research undertakings, and wider game production. It converts VGDL into a tool for game developers and functions as a research equipment.

There are numerous languages, each with a distinct role in defining the characteristics and behaviors of agents. For example, the JIAC Agent Description Language (JADL) provides a means of describing an agent's environment by defining its goals and constraints. Its rule framework is straightforward, consisting of a condition and two associated actions - one executed when the condition becomes true, and the other when it becomes false [7]. JADEL (Java Agent Development Framework Language) was developed to facilitate the construction of systems based on the Java Agent Development Framework (JADE). Its objective is to aid in the creation of agents, behaviors, and ontologies, with the ultimate aim of streamlining the integration of agents as essential components within larger systems [8].

The Ludi GDL [9] is a game description language that furnishes more precise tools for playing, comprehending and synthesizing names within a scope of GDL. It is anchored in a ludemic comprehension of games, which alludes to a profound insight and understanding into the mechanics, dynamics, strategies, and underlying principles of games. The language can independently assess games for their ability to captivate players, and generate innovative games with outstanding quality. This showcases a practical methodology for automated game testing.

An intriguing method of illustrating general gameplay is through the use of GDL-III - a language which incorporates imperfect information and introspection [10]. As a result, it boasts a simpler syntax for representing actions and knowledge specifically for gaming purposes. GDL-III advances upon GDL, introducing the ability to incorporate epistemic games governed by rules dependent on the players' knowledge.

The General Environment Description Language (GEDL) [11] offers a comprehensive solution for representing environments suitable for robots and agents. GEDL is inspired by the human mind's knowledge organization to construct a conceptual system and learn from errors. It consists of three components. Conceptual System, Occurrences, and Experience all have a crucial role in creating a comprehensive description of the environment.

GEDL's constructed reality encompasses both individuals and objects. Individuals are capable of acquiring knowledge and recognising objects that have varied features and relationships. The effectiveness of language in portraying an environment underscores its potential to serve as a basis for creating intelligent, adaptable general-purpose systems in the future.

Although there are significant differences in how organisms and machines work, it is possible to find connections and use them to optimize AI-based solutions. General intelligence encompasses the ability to reason contextually, adopt various perspectives, set objectives, and competently handle uncertain or ambiguous information [12]. Exploring new possibilities is closely related to the problem-solving process, but is difficult to address algorithmically.

Creating an artificial human brain is a well-established topic, but it remains challenging. The ability to store and organize knowledge and to solve previously unknown problems is what artificial general intelligence aims to achieve. However, there are two significant challenges to overcome in simulating the human brain. Until recently, humanity lacked the technological capability to construct computers that could handle thought processes due to insufficient information processing rates. Furthermore, creating intelligent machines resembling the brain requires understanding specific connectivity patterns and dynamic parameters necessary for artificial neural circuits to exhibit brain-like behaviour. Given that the human brain is potentially the most complex system known within the scope of human understanding, it is unsurprising that little progress has been made in acquiring a deep understanding of how our brains function [13].

Currently, an increasingly popular use of AI, known as ChatGPT, is gaining traction. This sophisticated system possesses the ability to comprehend and decipher human language, thereby allowing users to pose questions and obtain responses. Not only does it offer a novel method to search for information, but it also seeks to transform many facets of daily existence [14]. ChatGPT may be considered a natural language processing system, given the seemingly boundless potential of this tool [15]. It is crucial to recognize, however, that ChatGPT is much more than solely this. The latest version of GPT-4 demonstrates a remarkable level of proficiency and often surpasses previous models. With its extensive capabilities and impressive competence, it could potentially be considered as an early version of an artificial general intelligence system [16].

2.1. Motivation

The aim of the study is to explore the possibility of establishing a universal approach for finding solutions to problems using available resources. The basis for this attempt is the use of the General Environment Description Language (GEDL). This notation forms the initial premise for achieving the set goal. It will be employed to describe the current initial state of the environment, as well as the projected final state of the environment in light of the existing challenge. With this information, an individual possessing knowledge is highly likely to develop a series of necessary steps guiding towards a resolution. This expectation applies to a program written in any programming language.

Therefore, this paper proposes an algorithm that facilitates the construction of a list of actions required to move from the initial state to the final state. For effective implementation of this concept, a detailed representation of the accessible objects, their relationships and the executable actions is essential. The GEDL notation provides an opportunity to accomplish this task through a specific example.

3. General Environment Description Language

General Environment Description Language is a notation capable of describing different environments in which robots or agents can operate. It also allows to assign meaning to objects in the environment and to plan tasks based on these objects.

The concept behind this language is inspired by the way the human mind works and organizes knowledge. The process of acquiring information involves the creation of a personal conceptual system in which a collection of ideas and thoughts come together to form a meaningful whole that exceeds the significance of the individual components. A remarkable aspect is the ability to learn from mistakes, allowing the correction and adaptation of this conceptual system. In addition, intelligent beings have the ability to transfer not only ideas but entire concepts to others, with the clear advantage of saving time and resources that would otherwise be spent on individual learning. All these features have been used to develop a comprehensive language for describing the environment.

3.1. Description

The description is based on how we, as humans, perceive and interact with the world. To better understand how the model was created, a simple assumption is made: individuals and objects exist in an environment and can form relationships.

The environment is a representation of the physical world or another form of reality and comprises individuals and objects. Individuals can acquire knowledge through cognitive mechanisms and recognize objects as elements of the environment. These objects are referred to as instances in the knowledge of the individual and possess certain features. Features may be organized into feature sets and assigned to objects of identical instances. In any described world, objects can have relationships with each other (or an individual who is in fact an object). A proper relationship must follow from logical premises. A concept of a relationship is also identified.

Instances have the ability to perform actions independently or in cooperation with other instances. These actions serve as a means of changing the state of the environment by modifying properties, relationships, or creating and deleting objects. An action concept refers to a group of actions that modify objects with the ultimate goal of achieving a comparable state. It is important to note that each action is considered atomic, meaning that it cannot be further broken down into smaller steps. For example, the action of placing an object on a shelf is a complete action in itself, as it cannot be further broken down into smaller movements or gestures.

Individuals acquire knowledge and develop their own comprehension of reality. The information obtained through perception of the environment and concepts learned is referred to as individual knowledge. A systematic set of rules can be developed through observation, deduction, improvement of previously gathered facts or by obtaining information from other individuals. The latter method appears promising for increasing the speed of knowledge growth.

The individual knowledge consist of three different elements: Conceptual System, Occurrences and Experience, each of which plays an important role in composing a complete environment description.

A Conceptual System presents an individual's perception of concepts connected to entities, interactions, and activities. To comprehensively illustrate the system, it is essential to describe five components: features, feature sets, instance concepts, relationship concepts and action concepts. Features are defined as a combination of a title and a range of possible values that allow instances to be categorized. When an instance has the designated attributes and corresponding values that belong in specific intervals, personal knowledge can infer that this instance is part of the given concept. Attributes shared among objects of the same instance are included in sets of features. Features are grouped under a single label to improve understanding and streamline classification. As expected, a relationship concept defines the correlation that exists between two instances. Connections can develop: some may end due to environmental changes, while new ones can emerge. Transformations occur through the execution of actions. In this context, a solitary activity is seen as a transition from the initial state to the ultimate state. The input description is expected to be provided with any resulting changes caused by this activity.

Occurrences can be described as a limited set of instances and the relationships among them. This set includes only those parts that an individual can identify. In contrast to the conceptual structure, which provides explanations and interconnections, occurrences have unique attributes with individual values, and activities can be performed using them.

Experience is a part of an individual's knowledge that can initiate a search for solutions. When a problem is viewed in relation to its surrounding environment, an individual can begin to explore strategies that can lead to the desired outcome. The goal function is essential to facilitate the selection of the optimal solution. As a consequence, step-by-step guidelines leading to the expected state of the environment are provided.

In the GEDL notation a problem is defined as the modification of the environment using available actions with the aim of moving from the initial state to the final state. Various solutions can be identified, and the best one is selected based on specific objectives, such as minimizing time or the number of steps taken.

Every individual shapes their knowledge and understanding of reality. An individuals' knowledge consists of the information derived from observing their surroundings and acquiring concepts. Various approaches can be used to formulate organized principles, such as making observations, deductions, refining past discoveries, or acquiring insights from others. The latter approach appears promising for accelerating the process of expanding knowledge.

3.2. Example

A complete description of all the elements required for storing an individual's knowledge using GEDL can be written in any notation that fulfills its requirements. In this work, the environment is defined using JSON notation. This approach enables a universal description that is easily comprehensible to humans. JSON files are widely used in various systems and can be interpreted by most programming languages. Graphical and logical representations of collections, nested objects, etc. are provided to help visualize the dependencies that occur in the described world.

To improve understanding of GEDL, this paper provides a simple example to illustrate its construction. The example is positioned in appendix 1 and includes an individual capable of moving and a board divided into nine equal squares. For simplicity, we consider it a robot that explores the environment, makes decisions based on its knowledge and gathered information. The individual can interact with some squares, while others are inaccessible. The task is for the robot to reach a specified position; to achieve this, it must find a path on the board. We assume the individual can only move in four directions: up, down, right, or left, without any diagonal movements considered. The solution to this problem is easy for a human to find, given the small environment. To enable a robot to find the path, a suitable description is necessary. The example is presented in the Figure 1.

The Conceptual System consists of features, instance concepts, relationship concepts and action concepts. There are three defined features:

- available - informs whether it is possible to move to a particular square or not,
- isSquare - informs whether the object is a square,
- isIndividual - informs if the object is the individual moving on the board.

In this case the value can be either true or false, more advanced examples might include a wider range.

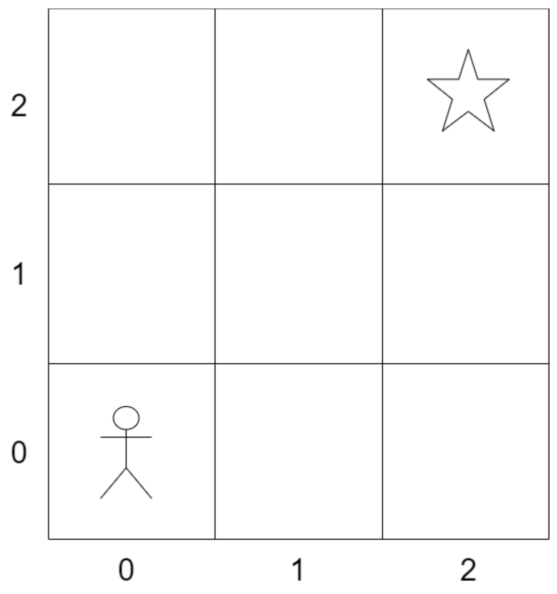


Figure 1: An example of an environment

In the following part of the Conceptual System, there is a description of instance concepts. Only two types of instances can be distinguished in this example: c_individual, which refers to an individual capable of movement, and c_square, which refers to a single square on the board. It is important to note that there will be multiple instances of that object, but the concept is defined only once. Instances have assigned features, as described above, but their values may be restricted. For instance, an object of the c_individual instance must be an individual, so the "isIndividual" property must be set to true. Similarly, an object of the c_square instance must be a square, though it can be either available or unavailable. To provide information about interaction with other instances, the concept of relationship is assigned. It is assumed that individuals cannot leave the board, so they are always placed on one of the squares. Conversely, a particular square may not always have an individual standing on it. The concept of action is defined by activities that involve objects of a particular instance interacting and modifying the existing environment. Certain instances have the ability to perform actions, either alone or with other objects, whereas others lack this capability.

In order to accurately describe the dependencies that occur in the environment, five relationship concepts have been defined. A relationship is posited between two instances, namely “role 1” and “role 2”. However, complex relationships necessitate being broken down into atomic relationships. The c_isPlacedOn concept signifies that an individual has been placed on a square. Conversely, the c_adjoinLeft, c_adjoinRight, c_adjoinTop and c_adjoinBottom concepts furnish insights into how the squares are arranged.

Providing a complete depiction of the action concept necessitates specifying the most information. As previously stated, an individual can move in four directions - up, down, left, and right, which is why four action concepts were introduced. Each action concept comprises an initial state concept and a final state concept. For instance, let's consider the c_moveRight concept that represents an individual moving towards the right of the board; the initial state denotes standing on a square immediately adjacent to another square on the right, while the final state represents standing on the right square. However, the correct conditions must exist before the action is carried out; in this case, the square to the right must be available and present on the board.

Occurrences differ from the Conceptual System as they comprise physical entities existing within the environment. The representation is based on concepts but now provides explicit values. Out of the ten examples, nine depict squares. An individual, identified as Me, is located on a square marked as s_0_0. As a result, the attainable attribute is deemed false because the individual cannot move to this square.

To complete the representation of Occurrences, it is crucial to create a comprehensive list of existing connections. This provides details about the board's layout, which helps the individual to understand the environment. It should be noted that the description of Occurrences represents an initial state in which relationships and attributes may change as actions unfold. For example, the concept c_adjoinRight establishes a relationship between squares s_0_0 and s_1_0, indicating that square s_0_0 is located to the right of square s_1_0. A fragment of Occurrences description, containing description of two squares is presented in Figure 2.

```
{
        "name": "s_0_0",
        "instanceConcept": "c_square",
        "features": [

            "name": "isSquare",
            "value": [true]
          },
          {
            "name": "available",
            "value": [false]
          }
        ]
      },
      {
        "name": "s_1_0",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "isSquare",
            "value": [true]
          },
          {
            "name": "available",
            "value": [true]
          }
        ]
      }
```

Figure 2 Square s_0_0 and s_1_0 description

Defining the problem is the final part of the complete description. Similar to the action concept, a problem represents a type of change that is expected to occur, hence the need for a starting and an ending state. Based on this foundation, finding a solution should be initiated by recognizing the difference between these two states. In the example provided, an individual is located on square s_0_0 when the environment is in the initial state and on square s_2_2 when it is in the final state. The individual's knowledge includes occurrences that exist in the environment and can be used to perceive a solution. Problem description is provided in Figure 3.

```
"experience": [
    {
      "problem": {
        "name": "Move to position",
        "individualKnowledgeFragment": {
          "instances": [
            "Me",
            "s_0_0", "s_1_0",
            "s_2_0", "s_0_1",
            "s_1_1", "s_2_1",
            "s_0_2", "s_1_2",
            "s_2_2"
          ]
        },
        "initialStateConcept": {
          "instanceConceptVariables": [
            {
              "name": "Me",
              "instanceConcept": "c_individual",
              "features": {}
            },
            {
              "name": "s_0_0",
              "instanceConcept": "c_square",
              "features": {}
            }
          ],
          "relationships": [
            {
              "relationshipConcept": "c_isPlacedOn",
              "role1": "Me",
              "role2": "s_0_0"
            }
          ]
        },
        "finalStateConcept": {
          "instanceConceptVariables": [
            {
              "name": "Me",
              "instanceConcept": "c_individual",
              "features": {}
            },
            {
              "name": "s_2_2",
              "instanceConcept": "c_square",
              "features": {}
            }
          ],
          "relationships": [
            {
              "relationshipConcept": "c_isPlacedOn",
              "role1": "Me",
              "role2": "s_2_2"
            }
```

Figure 3 Problem description

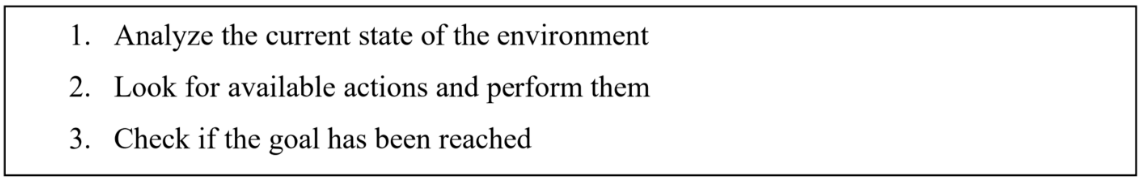
The example aims to illustrate the level of detail required to create an appropriate description of an environment. Despite the example's simplicity, data on adjacent squares, their availability, and their correlation to the individual is deemed crucial in achieving a thorough comprehension.

4. Algorithm description

As emphasized in the preceding section of the document, the algorithm responsible for finding a solution to any problem described in GEDL notation is essential to make general-purpose systems practical. A solution is a sequence of actions that ultimately leads to the desired result - the outcome that represents the problem's final state - when carried out in the prescribed order. The algorithm needs to recognize the existence of various possible outcomes; however, since the goal function is absent in the given example, and there is no way to select the ideal one, the algorithm chooses the first among them as the ultimate response.

As reaching the final state of a problem is necessary for finding a solution, the initial approach to dealing with this challenge was to begin the process from that point. The idea behind this concept was to examine the final state and evaluate the results of all possible actions. If a particular action was discovered to produce the desired result, the search would restart, this time looking for another action that had the potential to enable the previously identified action. However, this idea was abandoned as it became increasingly complex to deal with more complicated examples. It had the potential to overlook the instances involved in the problem-solving process since the final state of a problem only shows the outcome. In this situation, it was reasonable to select a defective device (if one was available) to execute an action as it was impossible to differentiate between working and non-working machines. Moreover, tracing from the final state to the initial state to solve problems created a variety of possibilities, making the decision-making process difficult. In specific situations, several actions may require to be executed before a new opportunity arises; reversing this process retrospectively made the task inefficient and vulnerable to errors.

Instead, a new approach was proposed. The effective solution follows the parallel of the problem-solving process in the human brain, as shown in Figure 4.



1. Analyze the current state of the environment
2. Look for available actions and perform them
3. Check if the goal has been reached

Figure 4: Steps of the proposed algorithm

In Phase 1, the algorithm examines the current state of the environment, including initial positions, attributes, conditions, and interrelationships among objects in the setting. Based on the gathered information, a simulated reality is created to carry out problem-solving.

In Phase 2, we pursue an actionable step that is feasible within the current state. It is worth noting that this methodology allows for identifying the instance that carries out an action at any given moment. After identifying the action, it is executed, bringing about changes in the environmental state. This could include the displacement of objects, changes in their properties, or the establishment of new relationships within the environment.

During Phase 3, the algorithm assesses whether the desired goal has been reached. If successful, the sequence of actions is considered a possible solution. If the goal has not been reached, the algorithm restarts the process, starting from the updated state of the environment. This involves finding the next available action and carrying it out, causing further modifications to the environment. This process continues in a loop until the final solution is achieved or there are no more available actions. Figure 5 provides a visual representation of the complete procedure.

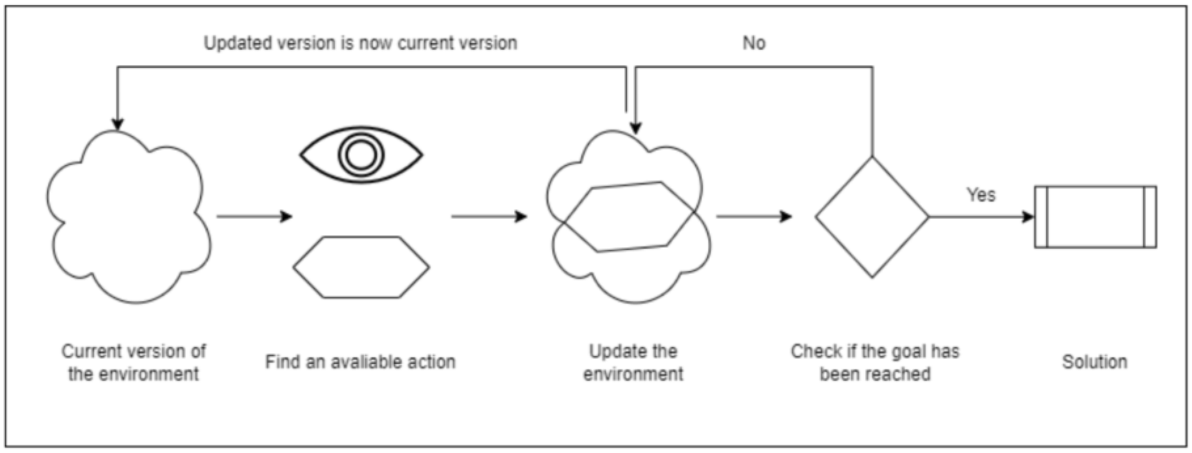

Updated version is now current version
No
Yes
Current version of the environment
Find an avaliable action
Update the environment
Check if the goal has been reached
Solution

Figure 5: Illustrated process of finding a solution

The proposed solution is based on the idea of creating different courses of action and identifying a specific approach for each. When multiple options are available, a copy of the current environmental state is created and then modified using the appropriate action. If the modification hinders progress while the objective is still unmet, the specific course of action is abandoned. It is plausible that several courses of action may result in the same exact environmental state. For example, adding detergent before putting clothes in the washing machine produces the same result as doing these tasks in the reverse order.

The algorithm comprises three crucial components. In the first phase, the current environmental state is examined to determine whether it aligns with the final state of the problem. If the answer is positive, it implies that either a solution has been discovered or the problem has already been resolved. To determine whether further action is required, the attribute values of the respective instances and existing connections are compared. Upon identifying discrepancies, an investigation is initiated to explore feasible modifications. To prevent insignificant comparisons when the available resources are insufficient to provide a solution, a procedure to determine the practicality of executing an action is introduced.

This aspect of the algorithm determines the actions that are practicable to execute in the current environmental condition. This evaluation is carried out by comparing the attributes and links of existing instances with the initial state of the proposed action. When a match is identified, a new alternative is recognized, leading to the creation of a new path. Lastly, the environment is adjusted to simulate the execution of the selected action, with values overwriting in agreement with the end-state specifications.

It is appropriate to presume that the process of determining the feasibility of an action prevents situations where a transition from the starting point to the end point is impractical. This process occurs across all paths at the same time. Each outcome is called a 'modified problem' because taking effective action creates a new problem that needs to be addressed. If the problem persists, and progress is not possible, the path is closed and can no longer be explored further.

4.1. Results and improvements

The solution was implemented using the Python programming language. The first step involves extracting environmental details from JSON files. In order to enable the algorithm to work with objects efficiently, the Conceptual System, Occurrences, and Experience descriptions are converted into classes, ensuring easy access to the key data.

Within the context of moving on a board, the initial accurate response was located after 10 iterations. The proposed solution asserted that an individual could move upwards twice and rightwards twice to arrive at the destination. Even though alternate solutions exist, like moving rightwards and then upwards twice, the algorithms terminated after detecting the initial precise response.

The solution was discovered reasonably promptly and without obstacles, although this may not always be the case. It transpired that the leading cause of slowing down the algorithm execution is its inclination to conduct repetitive iterations, even though it lacks sound reasoning based on human intelligence principles. For instance, it is quite apparent that opening a door allows the option to shut it immediately afterwards, and vice versa. The current iteration of the algorithm becomes trapped in a loop when it detects that two actions can be executed in immediate succession. Initially, the concept to resolve this matter was to disallow repeating a specific action. However, this method is not entirely correct, as many circumstances require the same action to be repeated. The algorithm was revised by including regulations that are specifically tailored to optimization. In the context of the given algorithm, optimization defines a factor that can eliminate specific opportunities that do not lead to the required solution.

Regarding the previously mentioned problem, a condition that must be fulfilled in order to avoid loops and to add a new possibility does not allow the complementary actions to be performed one after the other. As a result, there are fewer iterations and the solution is found more quickly.

Another difficulty present in the current version of the algorithm pertains to multi-criteria decision making. Assessment of the discovered solution typically entails selecting from various alternatives or options, thereby rendering the decision-making process quite intricate. Furthermore, every possibility examined is saved in a graph format. That means it necessitates the creation and storage of an environment every time a modification is made, resulting in storage difficulties due to the solution's rapidly increasing size. Additionally, examining a structure as intricate as one that can be generated whilst tackling more advanced problems demands a substantial amount of time.

Determining whether a potential solution constitutes a definitive one is a significant challenge. The algorithm must be run until either all options have been explored or there is confidence that any remaining possibilities would not surpass the current optimal solution. While examining each possibility may prove impractical due to the vastness of the environment, identifying whether the best outcome has been reached necessitates running a high level of analysis.

5. Summary

The history of the development of automated systems illustrates their impact on various industries. Advancements in computer technology and the emergence of control engineering have additionally contributed to the automation of processes. Furthermore, artificial intelligence has the potential to transform the accomplishment of different tasks without human supervision. Nevertheless, the current method is 'narrow AI' that concentrates on particular tasks instead of general intelligence. It is crucial to have versatile systems capable of learning autonomously and adapting to diverse environments to facilitate further progress. Developing general-purpose systems necessitates an algorithm that can find solutions to any problem, which is also the focal point of this study.

The General Environment Description Language offers a means of describing the different environments where robots or agents function. The inspiration for this language is drawn from how the human mind organizes knowledge, including the ability to learn from mistakes, transfer concepts to others, and save time and resources. With its ability to provide precise definitions of conceptual systems, occurrences, and experience, GEDL is capable of solving a wide range of problems. Successfully, a description of an environment that involves a washing machine, a manipulator, and a basket of clothes was created to solve the problem of doing laundry.

The algorithm is the critical component of the study, enabling the resolution of problems described in the GEDL notation environment. The solution was found in both evaluated tests, demonstrating that the algorithm is universally applicable and capable of functioning correctly under diverse conditions. The research entailed certain challenges that had to be addressed to ensure the outcome was general and unrestricted. Nevertheless, no constraints render it relatively easy for the program to enter loops, as each condition introduced may limit some possibilities. Accelerators were introduced in an attempt to speed up the process of solving the problem. They have been adapted to the examples mentioned in the paper, but it is possible to make them more universally applicable in further development.

6. References

[1] S. Bennett, A brief history of automatic control, in IEEE Control Systems Magazine, vol. 16, no. 3, pp. 17-25, June 1996, doi: 10.1109/37.506394.

[2] B. Goertzel, I. Matthew, J. Wigmore, The Architecture of Human-Like General Intelligence, In: Theoretical Foundations of Artificial General Intelligence, Atlantis Thinking Machines, vol 4. Atlantis Press, Paris, doi: 10.2991/978-94-91216-62-6_8.

[3] B. Goertzel, Artificial General Intelligence: Concept, State of the Art, and Future Prospects, Journal of Artificial General Intelligence, vol.5, no.1, 3914, pp.1-48. 2014, doi: 10.2478/jagi-2014-0001.

[4] I. Mordatch, P. Abbeel, Emergence of Grounded Compositional Language in Multi-Agent Populations, Proceedings of the AAAI Conference on Artificial Intelligence, 32, 1, April 2018, doi: 10.1609/aaai.v32i1.11492.

[5] M. Thielscher, A general game description language for incomplete information games. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10), AAAI Press, 994–999, July 2010

[6] M. Johansen, M. Pichlmair and S. Risi, Video Game Description Language Environment for Unity Machine Learning Agents, 2019 IEEE Conference on Games (CoG), London, UK, 2019, pp. 1-8, doi: 10.1109/CIG.2019.8848072.

[7] T. Konnerth, B. Hirsch, S. Albayrak, JADL – An Agent Description Language for Smart Agents. In: Baldoni, M., Endriss, U. (eds) Declarative Agent Languages and Technologies IV. DALT 2006. Lecture Notes in Computer Science, vol 4327. Springer, Berlin, Heidelberg. doi: 10.1007/11961536_10.

[8] F. Bergenti, E. Iotti, S. Monica A. Poggi, Agent-Oriented Model-Driven Development for JADE with the JADEL Programming Language. Computer Languages, Systems & Structures, June 2017, doi: 50. 10.1016/j.cl.2017.06.001.

[9] C. Browne and F. Maire, Evolutionary Game Design, in IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 1, pp. 1-16, March 2010, doi: 10.1109/TCIAIG.2010.2041928.

[10] M. Thielscher, GDL-III: A Description Language for Epistemic General Game Playing. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 1276-1282, August 2017, doi: 10.24963/ijcai.2017/177.

[11] K. Zatwarnicki, W. Pokuta, A. Bryniarska, A. Zatwarnicka, A. Metelski, E. Piotrowska, General Environment Description Language, Applied Sciences. January 2021; 11(2):740. Dou: 10.3390/app11020740.

[12] A. Roli, J. Jaeger, S. Kauffman, How Organisms Come to Know the World: Fundamental Limits on Artificial General Intelligence, Frontiers in Ecology and Evolution. January 2022, doi: 10.3389/fevo.2021.806283.

[13] H. de GARIS, S. Chen, B. Goertzel, L. Ruiting, A world survey of artificial brain projects, Part I Large-scale brain simulations. Neurocomputing. 74. 3-29, December 2010, doi: 10.1016/j.neucom.2010.08.004.

[14] J.-J. Zhu, J. Jiang, M. Yang, and Z. J. Ren, ChatGPT and Environmental Research Environmental Science & Technology Article ASAP, doi: 10.1021/acs.est.3c01818.

[15] J. Deng, Y. Lin, The Benefits and Challenges of ChatGPT: An Overview, Frontiers in Computing and Intelligent Systems. 2, 81-83, January 2023, doi: 10.54097/fcis.v2i2.4465.

[16] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, Y. Zhang, Sparks of Artificial General Intelligence: Early experiments with GPT-4, 2023, url: https://arxiv.org/abs/2303.12712, doi: 10.48550/arXiv.2303.12712.

7. Appendix 1

Appendix contains Conceptual System, Occurence and Experience description.

7.1. Conceptual System

```
{
  "ConceptualSystem": {
    "features": [
      {
        "name": "available",
        "set": [true, false]
      },
      {
        "name": "isSquare",
        "set": [true, false]
      },
      {
        "name": "isIndividual",
        "set": [true, false]
      }
    ]
  }
}
{
  "instanceConcepts": [
    {
      "name": "c_individual",
      "features": [
        {
          "name": "isIndividual",
          "set": [true]
        }
      ],
      "relationshipConcepts": [
        {
          "name": "c_isPlacedOn"
        }
      ],
      "actionConcepts": [
        {
          "name": "c_moveRight"
        },
        {
          "name": "c_moveLeft"
        },
        {
          "name": "c_moveTop"
        },
        {
          "name": "c_moveBottom"
        }
      ]
    },
    {
      "name": "c_square",
```

```
      "features": [
        {
          "name": "available",
          "set": [true, false]
        },
        {
          "name": "isSquare",
          "set": [true]
        }
      ],
      "relationshipConcepts": [],
      "actionConcepts": []
    }
  ]
}
],
"relationshipConcepts": [
      {
        "name": "c_isPlacedOn",
        "role1": "c_individual",
        "role2": "c_square"
      },
      {
        "name": "c_adjoinLeft",
        "role1": "c_square",
        "role2": "c_square"
      },
      {
        "name": "c_adjoinRight",
        "role1": "c_square",
        "role2": "c_square"
      },
      {
        "name": "c_adjoinTop",
        "role1": "c_square",
        "role2": "c_square"
      },
      {
        "name": "c_adjoinBottom",
        "role1": "c_square",
        "role2": "c_square"
      }
    ],

"actionConcepts":
[
  {
    "name": "c_moveRight",
    "initialStateConcept": {
      "instanceConceptVariables": [
            {
              "name": "Me",
              "instanceConcept":
"c_individual",
              "features": []
            },
            {
              "name": "squareOn",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [true]
                }
              ]
            },
            {
              "name": "squareAdjoinRight",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [true]
                }
              ]
            }
          ],
          "relationships": [
            {
              "concept": "c_adjoinRight",
              "role1": "squareOn",
              "role2": "squareAdjoinRight"
            },
            {
              "concept": "c_isPlacedOn",
              "role1": "Me",
              "role2": "squareOn"
            }
          ]
        },
        "finalStateConcept": {
          "instanceConceptVariables": [
            {
              "name": "Me",
              "instanceConcept":
"c_individual",
              "features": []
            },
            {
              "name": "squareOn",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [true]
                }
              ]
```

```
            },
            {
              "name": "squareAdjoinRight",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [false]
                }
              ]
            }
          ],
          "relationships": [
            {
              "concept": "c_adjoinRight",
              "role1": "squareOn",
              "role2": "squareAdjoinRight"
            },
            {
              "concept": "c_isPlacedOn",
              "role1": "Me",
              "role2": "squareAdjoinRight"
            }
          ]
        }
      },
      {
        "name": "c_moveLeft",
        "initialStateConcept": {
          "instanceConceptVariables": [
            {
              "name": "Me",
              "instanceConcept":
"c_individual",
              "features": []
            },
            {
              "name": "squareOn",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [false]
                }
              ]
            },
            {
              "name": "squareAdjoinLeft",
              "instanceConcept": "c_square",
              "features": [
                {
                  "name": "available",
                  "value": [true]
                }
              ]
            }
          ],
    "relationships": [
      {
        "concept": "c_adjoinLeft",
        "role1": "squareOn",
        "role2": "squareAdjoinLeft"
      },
      {
        "concept": "c_isPlacedOn",
        "role1": "Me",
        "role2": "squareOn"
      }
    ]
  },
  "finalStateConcept": {
    "instanceConceptVariables": [
      {
        "name": "Me",
        "instanceConcept":
"c_individual",
        "features": []
      },
      {
        "name": "squareOn",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "available",
            "value": [true]
          }
        ]
      },
      {
        "name": "squareAdjoinLeft",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "available",
            "value": [false]
          }
        ]
      }
    ],
    "relationships": [
      {
        "concept": "c_adjoinLeft",
        "role1": "squareOn",
        "role2": "squareAdjoinLeft"
      },{
        "concept": "c_isPlacedOn",
        "role1": "Me",
        "role2": "squareAdjoinLeft"
      }
{
  "name": "c_moveTop",
  "initialStateConcept": {
    "instanceConceptVariables": [
      {
        "name": "Me",
        "instanceConcept":
"c_individual",
        "features": []
      },
      {
        "name": "squareOn",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "available",
            "value": [false]
          }
        ]
      },
      {
        "name": "squareAdjoinTop",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "available",
            "value": [true]
          }
        ]
      }
    ],
    "relationships": [
      {
        "concept": "c_adjoinTop",
        "role1": "squareOn",
        "role2": "squareAdjoinTop"
      },
      {
        "concept": "c_isPlacedOn",
        "role1": "Me",
        "role2": "squareOn"
      }
    ]
  },
  "finalStateConcept": {
    "instanceConceptVariables": [
      {
        "name": "Me",
        "instanceConcept":
"c_individual",
        "features": []
      },
      {
        "name": "squareOn",
        "instanceConcept": "c_square",
        "features": [
          {
            "name": "available",
            "value": [true]
          }
        ]
      },
          {
            "name": "squareAdjoinTop",
            "instanceConcept": "c_square",
            "features": [
              {
                "name": "available",
                "value": [false]
              }
            ]
          }
        ],
        "relationships": [
          {
            "concept": "c_adjoinTop",
            "role1": "squareOn",
            "role2": "squareAdjoinTop"
          },
          {
            "concept": "c_isPlacedOn",
            "role1": "Me",
            "role2": "squareAdjoinTop"
          }
        ]
      }
    },
    {
      "name": "c_moveBottom",
      "initialStateConcept": {
        "instanceConceptVariables": [
          {
            "name": "Me",
            "instanceConcept":
"c_individual",
            "features": []
          },
          {
            "name": "squareOn",
            "instanceConcept": "c_square",
            "features": [
              {
                "name": "available",
                "value": [false]
              }
            ]
          },
          {
            "name": "squareAdjoinBottom",
            "instanceConcept": "c_square",
            "features": [
              {
                "name": "available",
                "value": [true]
              }
            ]
          }
        ],
        "relationships": [
          {
            "concept": "c_adjoinBottom",
            "role1": "squareOn",
            "role2": "squareAdjoinBottom"
          },
          {
            "concept": "c_isPlacedOn",
            "role1": "Me",
            "role2": "squareOn"
          }
        ]
      },
      "finalStateConcept": {
        "instanceConceptVariables": [
          {
            "name": "Me",
            "instanceConcept":
"c_individual",
            "features": []
          },
          {
            "name": "squareOn",
            "instanceConcept": "c_square",
            "features": [
              {
                "name": "available",
                "value": [true]
              }
            ]
          },
          {
            "name": "squareAdjoinBottom",
            "instanceConcept": "c_square",
            "features": [
              {
                "name": "available",
                "value": [false]
              }
            ]
          }
        ],
        "relationships": [
          {
            "concept": "c_adjoinBottom",
            "role1": "squareOn",
            "role2": "squareAdjoinBottom"
          },
          {
            "concept": "c_isPlacedOn",
            "role1": "Me",
            "role2": "squareAdjoinBottom"
          }
```

7.2. Occurrence

```
   {
     "occurrences": {
       "instances": [
         {
           "name": "Me",
           "instanceConcept":
"c_individual",
           "features": [
             {
               "name": "isIndividual",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },
         {
           "name": "s_0_0",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [false]
             }
           ]
         },
         {
           "name": "s_1_0",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },
         {
           "name": "s_2_0",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },
         {
           "name": "s_0_1",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },

         {
           "name": "s_1_1",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },
         {
           "name": "s_2_1",
           "instanceConcept": "c_square",
           "features": [
             {
               "name": "isSquare",
               "value": [true]
             },
             {
               "name": "available",
               "value": [true]
             }
           ]
         },
         {
          "name": "s_0_2",
          "instanceConcept": "c_square",
          "features": [
            {
              "name": "isSquare",
              "value": [true]
            },
            {
              "name": "available",
              "value": [true]
            }
          ]
        },
        {
          "name": "s_1_2",
          "instanceConcept": "c_square",
          "features": [
            {
              "name": "isSquare",
              "value": [true]
            },
            {
              "name": "available",
              "value": [true]
            }
          ]
        },
        {
          "name": "s_2_2",
          "instanceConcept": "c_square",
          "features": [
            {
              "name": "isSquare",
              "value": [true]
            },
            {
              "name": "available",
              "value": [true]
            }
          ]
        }
      ]
    }
  }
  {
  "relationships": [
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_0_0",
        "role2": "s_1_0"
      },
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_1_0",
        "role2": "s_2_0"
      },
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_0_1",
        "role2": "s_1_1"
      },
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_1_1",
        "role2": "s_2_1"
      },
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_0_2",
        "role2": "s_1_2"
      },
      {
  "relationshipConcept":"c_adjoinRight",
        "role1": "s_1_2",
        "role2": "s_2_2"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_2_0",
        "role2": "s_1_0"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_1_0",
        "role2": "s_0_0"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_2_1",
        "role2": "s_1_1"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_1_1",
        "role2": "s_0_1"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_2_2",
        "role2": "s_1_2"
      },
      {
  "relationshipConcept":"c_adjoinLeft",
        "role1": "s_1_2",
        "role2": "s_0_2"
      },


  {

"relationshipConcept":"c_adjoinTop",
        "role1": "s_0_0",
        "role2": "s_0_1"
      },
      {
```

```
"relationshipConcept":"c_adjoinTop",
         "role1": "s_0_1",
         "role2": "s_0_2"
       },
       {
"relationshipConcept":"c_adjoinTop",
         "role1": "s_1_0",
         "role2": "s_1_1"
       },
       {
"relationshipConcept":"c_adjoinTop",
         "role1": "s_1_1",
         "role2": "s_1_2"
       },
       {
"relationshipConcept":"c_adjoinTop",
         "role1": "s_2_0",
         "role2": "s_2_1"
       },
       {
    "relationshipConcept":"c_adjoinTop",
         "role1": "s_2_1",
         "role2": "s_2_2"
       },
       {
    "relationshipConcept":"c_adjoinBottom",
         "role1": "s_0_2",
         "role2": "s_0_1"
```

```
        },
        {
    "relationshipConcept":"c_adjoinBottom",
          "role1": "s_0_1",
          "role2": "s_0_0"
        },
        {
     "relationshipConcept":"c_adjoinBottom",
          "role1": "s_1_2",
          "role2": "s_1_1"
        },
        {
     "relationshipConcept":"c_adjoinBottom",
         "role1": "s_1_1",
          "role2": "s_1_0"
        },
        {
    "relationshipConcept":"c_adjoinBottom",
          "role1": "s_2_2",
          "role2": "s_2_1"
        },
        {
    "relationshipConcept":"c_adjoinBottom",
          "role1": "s_2_1",
          "role2": "s_2_0"
        },
        {
    "relationshipConcept":"c_isPlacedOn",
          "role1": "Me",
          "role2": "s_0_0"
        }
      ]
    }
```

7.3. Experience

```
{
  "experience": [
    {
      "problem": {
        "name": "Move to position",
        "individualKnowledgeFragment": {
          "instances": [
            "Me",
            "s_0_0",
            "s_1_0",
            "s_2_0",
            "s_0_1",
            "s_1_1",
            "s_2_1",
            "s_0_2",
            "s_1_2",
            "s_2_2"
          ]
```

```
          },
          "initialStateConcept": {
            "instanceConceptVariables": [
              {
                "name": "Me",
                "instanceConcept":
"c_individual",
                "features": {}
              },
              {
                "name": "s_0_0",
                "instanceConcept":
"c_square",
                "features": {}
              }
            ],
            "relationships": [
              {
                  "relationshipConcept":
"c_isPlacedOn",
                  "role1": "Me",
                  "role2": "s_0_0"
                }
              ]
            },
            "finalStateConcept": {
              "instanceConceptVariables": [
                {
                  "name": "Me",
                  "instanceConcept":
"c_individual",
                  "features": {}
                },
                {
                  "name": "s_2_2",
                  "instanceConcept":
"c_square",
                  "features": {}
                }
              ],
              "relationships": [
                {
                  "relationshipConcept":
"c_isPlacedOn",
                  "role1": "Me",
                  "role2": "s_2_2"
                }
              ]
            }
          }
        }
      ]
    }
```