EPiC Series in Computing

Volume 63, 2019, Pages 231–240

Proceedings of 32nd International Conference on Computer Applications in Industry and Engineering

A Generic Model of the World and Intelligence

Harris Wang
Athabasca University, Alberta, Canada
harrisw@athabascau.ca

Abstract

The resurgence of interest in Artificial Intelligence and advances in several fronts of AI, machine learning with neural network in particular, have made us think again about the nature of intelligence, and the existence of a generic model that may be able to capture what human beings have in their mind about the world to empower them to present all kinds of intelligent behaviors. In this paper, we present Constrained Object Hierarchies (COHs) as such a generic model of the world and intelligence. COHs extend the well-known object-oriented paradigm by adding identity constraints, trigger constraints, goal constraints, and some primary methods that can be used by capable beings to accomplish various intelligence, such as deduction, induction, analogy, recognition, construction, learning and many others.

In the paper we will first argue the need for such a generic model of the world and intelligence, and then present the generic model in detail, including its important constructs, the primary methods capable beings can use, as well as how different intelligent behaviors can be implemented and achieved with this generic model.

Keywords: agent-based systems, intelligent systems, knowledge management and ontologies, knowledge-based systems, machine learning, modelling of the world and intelligence, intelligent system development

1 Introduction

Since as early as the 50s of last century, the focus of research and development in Artificial Intelligence has been mostly on the discovery of theories and technologies enabling machines to do individual and specific intelligent tasks. These intelligent tasks that have been explored for computers to complete include search, automated reasoning, planning, perception, automated theorem proof and others [1][2][3][4]. In some successful applications such as robotics different intelligent tasks are involved and implemented but still in a confined environment dictated by a specific mission. Even the general problem solver (GPS) was only limited to solving certain problems [5][6].

So far, the most challenging undertaking for AI may be self-driving cars, or unmanned ground vehicles in the real traffic, landed in an unknown environment such as the Moon or the Mars. Even in

Q. Yuan, Y. Shi, L. Miller, G. Lee, G. Hu and T. Goto (eds.), CAINE 2019 (EPiC Series in Computing, vol. 63), pp. 231–240

those applications of AI, the intelligent tasks involved are still rather limited and clear to the designer and developer. These intelligent tasks in the case of unmanned ground vehicles in the real traffic would include

a. localization and mapping with Bayesian network-based algorithms such as SLAM [9],
b. perception or sensing of lane, traffic signals and signs, as well as moving and static obstacles heavily relying on neuro networks and deep learning technologies,
c. route planning, and
d. decision making on speed and direction.

If we assume, in the case of unmanned self-driving vehicles, there is a robot siting on the driver seat doing all the decision making and vehicle maneuvering, with the help of the sensors, how can we further empower the robot to be able to do something else, such as repair the build a vehicle, or just to deal with off-road situations? What knowledge and primary capability should the robot have in order to carry out these additional tasks? How can these new knowledge and skills be learned and easily integrated into its existing knowledge system and skillset of the driving robot?

We know a newborn baby can grow to fit into different professions requiring different knowledge and skills, with the same brain to learn in the same way even for many different challenging tasks in different domains. For robots or capable beings in general, is there a generic way to model the world for them act intelligently like our human beings?

To see the needs for a unified way to model the world for capable beings, let consider modelling a small world involving tables, chairs and other objects as needed, for a capable being so that the capable being will be able to

1. tell if an object is a chair or table
2. tell if a chair can be used as table, and vice versa
3. tell how many chairs should be made for a table already made
4. construct a table
5. construct a chair
6. decide, when construct a table, if some chairs can be used as legs of the table when it is necessary
7. use other objects as chairs in some circumstances
8. make plan for other capable beings to make tables and chairs
9. tell if a chair or table needs to be fixed, and
10. fix a chair or table when needed

As one can imagine, the completion of these tasks requires different intelligent capabilities including deduction, induction (analogy reasoning), planning, diagnose, and so on. What should a human being know and have in order to be able to complete these tasks? What is the best way for a capable being to acquire the necessary knowledge? Apparently, it is not the best way to write a specific program for each of the tasks. The goal of our research is to find a unified framework that can be used to model the world for all capable beings so that they can act intelligently and learn to become more capable.

In the remainder of this paper, we will present a unified approach to modelling the world for capable beings to complete various intelligent tasks. We will first introduce the static constructs of the model, and then discuss some primary methods that form the foundation of the intelligence. This will be followed by a presentation of how various intelligent tasks can be completed by capable beings equipped with a necessary model of the world.

2 The Constructs of the Generic Model

Our generic model of the world and intelligence is called Constrained Object Hierarchies or COHs. It is devised to capture the stereotype information of the world, any object within the world, and something the world is part of, such as the universe. It is built on an evolving consensus about the world as stated below:

1. The world is made of objects including people or capable beings in general, and
2. Anything can be seen and treated as an object or a smaller world
3. Any object is made of small ones, which are called parts
4. Any object has attributes
5. A capable being can do certain things with or within an object.
6. The things a capable being can do are called methods.
7. For an object, parts, attributes and methods do not exist in isolation.
8. Rather, the parts, the attributes and the methods are related or constrained somehow for the object to exist, function or live, or distinct itself from others.
9. An intelligent system consists of one or more capable beings, such as a human, a robot, a software agent equipped with a enough knowledge of the world they will be act in or upon, and the world or a simulated environment of the world
10. For an intelligent system to be intelligent, actions must be taken by a capable being autonomously, and these actions should be automatically triggered when certain conditions are met.
11. For an intelligent system to be useful and purposeful, goals should be set or developed for the world whenever possible, made aware to the capable beings

Compared to the object model of Object-Oriented Programming (OOP), our generic model of the world and intelligence has added three types of constraints, namely, identity constraints, trigger constraints and goal constraints. Together with other enhancements to the classical object model of OOP, these three new constructs have made COH capable of modelling the living world and the intelligence.

Formally, a constrained object hierarchy has the following constructs:

1. A name or ID used to identify the individual constrained object hierarchy
2. The class the individual constrained object hierarchy belongs to. For an object yet to be identified, the class is unknown. The first task that requires the intelligence of a capable being is to investigate the object to identify its parts and properties and their relationships, to identify the class it may belong to. If the object doesn't fit any existing class, then a new class may need to be created
3. Parts or objects this object is made of
4. Attributes the object has
5. Methods a capable being may use to act within or with the object
6. Identity constraints that define the object and differentiate the object from other objects, even those in the same class.
7. Trigger constraints that define the actions to be taken when certain conditions are met.
8. Goal constraints that define the desire state of the object

Note that OOP's abstraction principle still holds in our COHs, such that a class definition of COH should contain as many above-listed constructs as needed. When defining subclasses and making instances of the class, certain changes can be made to a class definition or its instances. For example, a table class may be modelled in COH as follows:

```
class Table of COH:
    ATT number_of_legs: Int  # number of legs – an attribute (ATT) of the table
```

```
        PRT table_top: (Tops, 10)          # table_top is an important (10) part (PRT) of a table
        PRT table_legs: [number_of_legs]Legs      # table_legs are part of a table
        ATT cost: LocalCurrency            # cost of the table
        ATT location_placement: [] Location       # where it will be placed
        ATT seating_number: [] Int         # how many seats can be fit around the table
        # defining the identity of tables
        Identities: Table_top is not NULL and number_of_legs>2 and
                for any j, k in range(number_of_legs): length(table_legs[j])==length(table_legs[k])
                number_of_legs == 1 => table_legs[0].diameter > 1/3*table_top.length
                number_of_legs==2 => for any j in [0, 1] table_legs[j].length > 1/3*table_top.width
                table_top.fit(location_placement)
        # defining triggers – when something specific can be done with a table object.
        Triggers:
                if table_top is NULL: table_top.toFind()
                for any j in range(number_of_legs): if table_legs[j] is NULL: table_legs[j].toFind()
                if table_top.isBroken(): table_top.toFix()
                for any j in range(number_of_legs): if table_legs[j].isBroken(): table_legs[j].toFix()
        Goals:
                _maximize_(seating_number)
                _minimize_(cost)
    End of Table
```

In the above class definition, we explicitly used all the constructs we mentioned before, except for methods, which will be inherited from super class COH, overridden later when needed. For the goal constraints, we set two goals, one if to minimize the cost whereas the other is maximize the seating. further added goal constraints of a class and objects of the class. An object can have more one goal. The following predicates can be used to prescribe the goals. There are other primary methods that can be used to define goals, which will be presented in the next section.

In our generic model, an intelligent system consists of one or some capable beings and a model of the world as a cluster of constrained object hierarchies. The trigger constraints will make the intelligent system autonomous, the identity constraints will ensure the identity of the world and the objects, whereas the goal constraints will lead the way for the system to evolve.

Theoretically and practically, we can model any object/world with this generic model and implement the intelligence with some primary methods a capable being can use. For example, COHs can be easily built for objects such as

1. Vehicle: car, SUV, truck, train, bus, and so on.
2. Building: residential, commercial, manufacturing, storage buildings, etc.
3. Animal: mammal, birds, fish, insects, etc.
4. People: students, professors, tutors
5. Courses
6. Classes
7. Programs
8. Languages
9. Universities, colleges, schools
10. Tables, desks, chairs
11. Lights
12. Machines

In the case of modelling languages, we can model Chinese characters use COH as follows, for the purpose of handwriting and speech recognition as well as generation:

COH Chinese_Character:

```
        UniCoding: UNICODE     # every Chinese character has a unique Unicode
        Spellings: []PinYin          # list of Pinyins because one character can be read differently
        StandardWritings: []ChineseCalligraphy       #one character can be written in different ways
        StandardPronounces: []ChinesePronounces # one character can be read differently
        SentinelPrefix: []UniCode # characters appear before this character
        SentinelAffix: []UniCode              # characters appear behind this character
        SampleWritings: []ChineseCalligraphy         # some positive samples of handwriting
    End Chinese_Character
```

Note that in the above COH definition we didn't define any constraint, but it can be added to an instance of Chinese_Character, when the character is known. Let's take 的 as an example to make an instance of Chinese_Character:

```
U7684x = Chinese_Character(UniCoding = '0X7684', Spellings = ['de', 'di4'],
                           SentinelPrefix = ['0x76ee', '0X6709', '0x4e00,0x8bed,0x4e2d'],
                           SentinelAffix =  ['0x653e,0x77e2'],
Triggers = ["If prefixed('0x4e00,0x8bed,0x4e2d') => spelled('di4')",
      "if affixed('0x653e,0x77e2') => prefixed('0X6709') and spelled('di4')])
```

In the above instance, we use the Unicode of the Chinese character as identifier to refer to the object, initialized with the Unicode, spellings in Mandarin Pin Yin, and characters that may appear before or after. More importantly, we added trigger constraints to the instance, which tells when certain characters prefixed or affixed, the character will be spelled as 'di4'. These constraints can be useful in speech generation.

In real applications, the instances of Chinese_Character can be quickly generated by scanning through existing documents to get the Unicode and context info in lieu of sentinel prefix and affix. The standard writings and pronunciations will need to be scanned or recorded from samples; the constraints can be learned or taught.

Because as we said that anything can be treated as object, so are methods, tasks, plans, schedules, and even abstract and virtual things such as scientific theories, technical or engineering protocols, the proposed framework can also be conveniently used in planning, scheduling, protocol analysis, and project management.

3 Primary methods of intelligence

Human and capable beings in general are born with some primary intelligence, and more advanced intelligence are developed from or implemented with the primary intelligence. This is reflected in our COH as primary methods that can be used, redefined, and evolved by capable beings. The following are some of these methods:

1. _distance_from_()

   The first primary intelligence of capable beings is the ability to judge and calculate the distance from one object to another. In our framework, this method is named _distance_from_(), and built-in with every object, though the real definition of the method can be different. So, ox._distance_from_(oy) will return the distance between ox and oy. Deep learning is a process of building a better function to calculate the distance between target objects and source objects using neural networks. the distance method can be implemented by calculating the distance in function, distance in structure, and distance in attribute.

2. _functional_distance_from_()
   this method is used to find out the distance between two objects in terms of their functionalities, which sometimes are concerned the most. For example, a newborn baby should be more to find an object that can feed her or him, rather than her or his biological mom. When we want to find something to sit on, we are more interested to find an object that can function as a chair for your purpose, rather than a super comfortable sofa, or fancy stool, as another example.
3. _attribute_distance_from_()
   this method is used to find out the distance between two objects in terms of their attributes, such as color, smell, taste, touch-feel. A newborn may accept any object that may provide the milk she or he wants, though he or she may still consider a human closer than a milk bottle, his biological mom than someone else. It will become even more so after some time when she or he already learned some of the attributes of the desired object, such as smell.
4. _structure_distance_from_()
   this method is used to find out the distance between two objects in terms of their structures. In the case of vehicle recognition, handwriting recognition, speech recognition or even face recognition, the structure of the objects matters the most.

   Note that when calculating the distances between two objects, it is not necessary to measure all the constructs of each objects. For example, when a human is looking for an ideal girlfriend or boyfriend, he or she may be only interested in some aspects of the objects.
5. _is_similar_()
   Being able to tell if two objects are similar is a primary intelligence of human and capable beings. This method will be using the distance methods discussed above. Because similar is very subjective, there will be a threshold on the distances between objects.
6. _is_the_same_()
   this method is also based on the methods on distance, when the distance is 0
7. _is_bigger_()
   this method is also based on the methods on distance, when the distance is a positive number
8. _is_smaller_()
   this method is also based on the methods on distance, when the distance is a negative number
9. _is_better_(), _is_worse_()
   these method is also based on the methods on distance, but that whether a positive distance or a negative distance is better is rather context-based and rather subjective
10. _is_part_of_()
    this method is to state that one object is part of another
11. _is_cause_of_()
    this method is to state that in this world or object, one thing is the cause of others
12. _is_result_of_()
    this method is to state that in this world or object, one thing is the result of others
13. _is_satisfied_()
    this method is to check if all identity constraints for the object are all satisfied
14. _maximize_()
    this method is to maximize the value of an attribute, and often used to define goal constraint
15. _minimize_()
    this method is to minimize the value of an attribute, and often used to define goal constraint

16. _stay_at_()
    this method is to maintain the value of an attribute, and often used to define goal constraint
17. _optimize_()
    this method is to find the best values for a list of attributes , and often used to define goal constraint

Note that the methods from item 5 to 13 have two versions: one is used in assertion, while the other is used in query.

The intelligence of capable beings with or within objects modelled as COHs can be demonstrated with the following primary methods:

1. _coh_()
   this is the first primary method a capable being can use to create a skeleton of an object observed, by decomposing it. For example, when someone is seeing an animal that he may not know what it is, he can still describe about its head, eyes, height, length, color of each body part, and so forth. All these parts and attributes would make up an object hierarchy. He may also be able to describe the relationships between the observed parts and attributes, which would make the object hierarchy constrained.
2. _add_part_()
   this method is used to add one part or a list of parts to an object. This and the following methods may be applied to both classes of objects, as well as their instances.
3. _delete_part_()
   this method is used to delete one part or a list of parts to an object
4. _add_attribute_()
   this method is used to add one attribute or a list of attributes to an object
5. _delete_attribute_()
   this method is used to delete one attribute or a list of attributes to an object
6. _add_method_()
   this method is used to add one method to a class or an object
7. _delete_method_()
   this method is used to delete one method from a class or an object
8. _add_identity_()
   this method is used to add one identity constraint to a class or an object
9. _delete_identity_()
   this method is used to delete one identity constraint from a class or an object
10. _add_trigger_()
    this method is used to add one trigger constraint to a class or an object
11. _delete_trigger_()
    this method is used to delete one trigger constraint from a class or an object
12. _add_goal_()
    this method is used to add one goal constraint to a class or an object
13. _delete_goal_()
    this method is used to delete one goal constraint from a class or an object
14. _satisfy_()
    this method will try to satisfy the identity constraints of the object by finding values for the parts and attributes involved. This process can be triggered through the trigger constraints or called by a capable being to start the process.

15. _assign_all_()
    this method will try to find proper values for all the parts and attributes while ensuring the identity constraints are satisfied. This process may also activate some triggers defined in the trigger constraints. For example, when building a home, the completion of framing will trigger a call for framing inspection.
16. _verify_all_()
    this method is used on an existing object. It will check the validity of the value every part or attribute member holds. If any value is invalid, a call to _validate_() will be made. In the case of self-driving vehicle, for example, if a tire is found flat, a call to _validate_ will be made to repair the tire.
17. _validate_()
    this method is used to make an invalid value of a part valid again, by either replacing the part entirely or part of it. This method can be called by _verify_all_ or triggered within the trigger constraints.
18. _enhance_()
    this method is used to find a better value for a part of the object. This method will call _distance_from_ method to look for better alternatives.

4 Implementation of high-level intelligence with COHs

In this section, we will explain how high-level intelligence can be further implemented and conducted by capable beings with or within COHs using the primary methods above.

Deductive reasoning

Deductive reasoning about the attributes of individual objects can be done by conducting queries on the object hierarchies directly or activated through triggers constraints. Firing of one trigger constraint may change the status of some objects, which may further fire off other trigger constraints. Hence, a chain of deductive reasoning is formed.

Inductive reasoning

When a collection of unknown or unclassified objects are observed, a capable being such a software agent will create a COH skeleton for each object, and then try to find a class it may fit by calculating the distances. If no such a class is found, it will try to find the similarities among the unknowns and derive a new class for them to best fit. Hence, inductive reasoning is achieved.

Analogical reasoning

Analogical reasoning occurs when the distance between two objects such as patients is close enough, such as all the observed symptoms have matched, then we could reasonably guess that some constructs of the two objects are most likely the same. In the case of patients, if all or most symptoms are the same, and a treatment has been proved effective on one patient, the treatment will most like be effective on the other patient. Hence, analogical reasoning is achieved.

Reasoning with uncertainties

In deductive reasoning, the condition of a trigger constraint can be stated with the _is_similar_(), _is_better_(), _is_worse_() or the like, from which the results would have uncertainties. More importantly, these uncertainties will then be propagated through the chain of reasoning according to some rules set by/for the capable being. Hence reasoning with uncertainties is achieved.

Planning, scheduling, and management

According to the generic model of the world and intelligence, we know that everything can be considered as objects, so are plans, schedules and projects. In plans, schedules and projects activities and tasks need to be sequenced for available and capable entities/beings to achieve set goals under certain conditions. All these can be naturally modelled as constrained object hierarchies. Once the modelling is done, a call to validate will start a planning or scheduling process. In these plans or schedules, each activity is bound with certain resources and capable entities. The sequencing and the relationships between activities, capable entities and resources can be prescribed in the identity constraints. A call to satisfy the constraints will start the execution of the plan, the schedule or the project. If a scheduled/planned capable being becomes unavailable during the execution, then a rescheduling will need to occur as an essential part of plan/schedule execution or project management. Hence, planning, scheduling and project execution and management are achieved.

Production: construction

Products and buildings can be easily modelled as COHs too. Once the modelling of desired product or building is done, the task of production or construction is to make an instance of the product or building, adding constraints/specs of customized product or building, and then call _assign_all_() to fill out all the parts and attributes while keep the constraints satisfied.

Diagnose and repair

Diagnose and repair are needed on an existing object as scheduled or activated by trigger constraints at higher level constrained object hierarchies. Diagnose is done by calling the _verify_all_() method to check the validity of all parts of the object. If a part is found invalid, a call _to_validate_() is made to make it part valid again. Hence, diagnose and repair are achieved.

Learning and discovery

Learning is the process of acquiring existing knowledge that is already known to others, whereas discovery is the process of creating and acquiring new knowledge that doesn't exist yet. However, the mechanisms of the two are sometimes the same or similar for some individual learners. In terms of our generic model of the world and intelligence, knowledge is the structure of constrained object hierarchies (COHs), the objects within the hierarchies, the attributes of all objects, the methods of actions capable beings can take within or upon the constrained object hierarchies, as well as the relationships among the objects, the attributes and methods of actions. In our generic model of the world and intelligence, the creation of COH skeletons for observed objects, and the addition of parts, attributes, methods of actions, the identity constraints, trigger constraints, and goal constraints are all the means of learning and discovery. Especially the process of establishing methods of distance between objects can be considered as the generalization of what neural networks and deep learning [10] are intended to do. Imagine a capable being repeatedly call distance methods that take an array of inputs representing the components and attributes of an observed object, and calculate the distances between the observed object and each of possible target objects such as human faces, fingerprints, characters in the Unicode table, and then choose the target object that has the shortest distance with the observed object. This is exactly what neural networks and deep learning do at testing stage.

Recognition and classification

As we have seen above, by using the distance methods best-fit objects can be found for unknown objects such as human faces, fingerprints, voices and handwritings. Hence recognition is achieved. When a best-fit class of objects (COH) is found or created for a collection of unknown or even known objects, classification is achieved. A new sub-class may be created for a collection of known objects in order to best describe the common features of the objects in the collection and differentiate them

from the other objects in the same parent class. For example, a new sub-class may be created for birds that don't fly.

5 Discussions

In the above sections, we have presented in detail a generic framework called constrained object hierarchies for the modelling of the world, and all kinds of worlds, for capable beings such as computers and robots, to act intelligently. We showed how the framework can be used, what primary methods capable beings need to have, how different intelligent capabilities can be achieved within a world by capable beings quipped with the model of the world.

Based on the generic framework, a programming language called GISMO (for Generic Intelligent System Modeler) is being developed and implemented and will be released soon.

This should be considered as a successful attempt towards a generic model of the world and intelligence that can effectively model the world for capable beings to act intelligently like our human beings. The model presented here may not be powerful enough now to capture every aspect of human intelligence, but we hope it will evolve and grow to be a true model of the world and intelligence.

References

[1] Pearl, J. Heuristics: Intelligent search strategies for computer problem solving. United States: N. p., 1984.

[2] B. Bonet and H. Geffner, Planning as Heuristic Search, Artificial Intelligence Journal, 129(1-2), pp. 5-33, 2001.

[3] Ghallab M., Nau D., and Traverso P., Automated Planning, theory and practice. Morgan Kaufmann, 2004.

[4] Christopher M. Bishop. Pattern Recognition and Machine Learning. Heidelberg, Germany: Springer 2006. i - xx, pp740

[5] M. Minsky, The Society of Mind, New York: Simon and Schuster, 1988

[6] Newell, A.; Shaw, J.C.; Simon, H.A. Report on a general problem-solving program. Proceedings of the International Conference on Information Processing. pp. 256–264. 1959.

[7] Schmidhuber, J. Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117, 2015

[8] Automated Reasoning, Stanford Encyclopedia. Last retrieved June 25, 2019, https://plato.stanford.edu/entries/reasoning-automated/

[9] Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping. IEEE Robotics & Automation Magazine. 13 (2): 99–110, June 2006.

[10] Huval, Brody; Wang, Tao; Tandon, Sameep; Kiske, Jeff; Song, Will; Pazhayampallil, Joel. An Empirical Evaluation of Deep Learning on Highway Driving. arXiv:1504.01716 [cs.RO]. 2015