A Domain Ontology for Task Instructions

Aaron Eberhart✉[1] [0000−0003−3007−5460], Cogan Shimizu[1] [0000−0003−4283−8701], Christopher Stevens[2], Pascal Hitzler[1] [0000−0001−6192−3472], Christopher W. Myers[2] [0000−0003−0556−5935], and Benji Maruyama[3]

[1] DaSe Lab, Kansas State University, Manhattan, KS, USA
[2] Air Force Research Laboratory, Wright-Patterson AFB, OH, USA
[3] Air Force Research Laboratory, Materials & Manufacturing Directorate, Wright-Patterson AFB, OH, USA
{aaroneberhart, coganmshimizu, hitzler}@ksu.edu, {christopher.myers.29, christopher.stevens.28, benji.maruyama}@us.af.mil

Abstract. Knowledge graphs and ontologies represent information in a variety of different applications. One use case, the Intelligence, Surveillance, & Reconnaissance: Mutli-Attribute Task Battery (ISR-MATB), comes from Cognitive Science, where researchers use interdisciplinary methods to understand the mind and cognition. The ISR-MATB is a set of tasks that a cognitive or human agent perform which test visual, auditory, and memory capabilities. An ontology can represent a cognitive agent's background knowledge of the task it was instructed to perform and act as an interchange format between different Cognitive Agent tasks similar to ISR-MATB. We present several modular patterns for representing ISR-MATB task instructions, as well as a unified diagram that links them together.

1 Introduction

Knowledge graphs facilitate data integration across highly heterogeneous sources in a semantically useful way. Knowledge graphs may be equipped with a schema, frequently an ontology, that combines the associative power of the knowledge graph with the semantics of the ontology. Due to this, they are uniquely suited to support research in cognitive science, where it is often necessary to incorporate information from fields like computer science, psychology, neuroscience, philosophy, and more.

Cognitive agents are a sub-field of cognitive science and an application of the more broad study of cognitive architectures. Cognitive architectures, like ACT-R [1] for example, are an approach to understanding intelligent behavior and cognition that grew out of the idea of Unified Theories of Cognition [8]. These systems have their roots in AI production systems and some types use rules-based cognition. Many in Computer Science are familiar with inductive themes from a different type, called Connectionism, due to its historic ties with artificial neural networks. Symbolic cognitive architectures, by contrast, are less widely known outside of cognitive science, and are abstracted and explicit like logic programming.

Both ontologies and cognitive architectures deal with symbolic knowledge. Symbolic cognitive architectures typically focus on the plausibility of knowledge and the way in which that knowledge is translated into human behavior within a specific task. Ontologies offer a set of robust mechanisms for reasoning over complex knowledge bases and could help cognitive architectures adapt to tasks in novel environments. One way the two may be integrated is by leveraging the ontology to reduce the specificity of a cognitive agent.

In general, cognitive agents are often specialized, or *differentiated*, to perform a specific task or set of tasks. An *undifferentiated* agent is one that has no specialization. The purpose of such an agent is to be adaptable to new tasks as needed. As part of initial work to develop such an undifferentiated cognitive agent, we have developed a modular ontology that captures instructions for a specific cognitive agent task called ISR-MATB. We discuss this platform in more detail in the next section.

Currently, the ontology supports the memory of a cognitive agent by adding structure to its knowledge and providing new varieties of query-like recall. And due to design methodology used during the modeling process, the ontology is general enough that it could model other cognitive agent experiments, which could then be evaluated against each other in a structured way. This allows the ontology to act as an invaluable interchange format between researchers developing cognitive agents.

The rest of this paper is organized as follows. Section 2 provides a brief overview of the use-case: ISR-MATB. Section 3 provides an in-depth examination of the ontology. Finally, in Section 4, we briefly conclude and discuss next steps.

2 ISR-MATB

ISR-MATB is a series of cognitive tasks that could be completed by a Cognitive Agent or a human [3]. A trial starts with one very simple task, the evaluation then branches into two sub-tasks that relate back to the first task. After the two sub-tasks are complete the agent completes one final task requiring integration of remembered information from all previous tasks. The final task is made more difficult by the possibility of incorrect feedback as the agent learns. ISR-MATB is intended to be repeated for a fixed time so that researchers can observe changes in the agent's response time and develop better computational cognitive agents.

2.1 Psychomotor Vigilance Test

The Psychomotor Vigilance Test is one of the more basic cognitive tasks [2]. In this task, there is an area of the screen where a letter could appear. The letter will be drawn with a specific color. When the letter does appear an agent must press a button that acknowledges they have seen it. If the agent pushes the button too soon a false start is recorded and the task continues normally. If too much time passes before the agent pushes the button then the task will continue

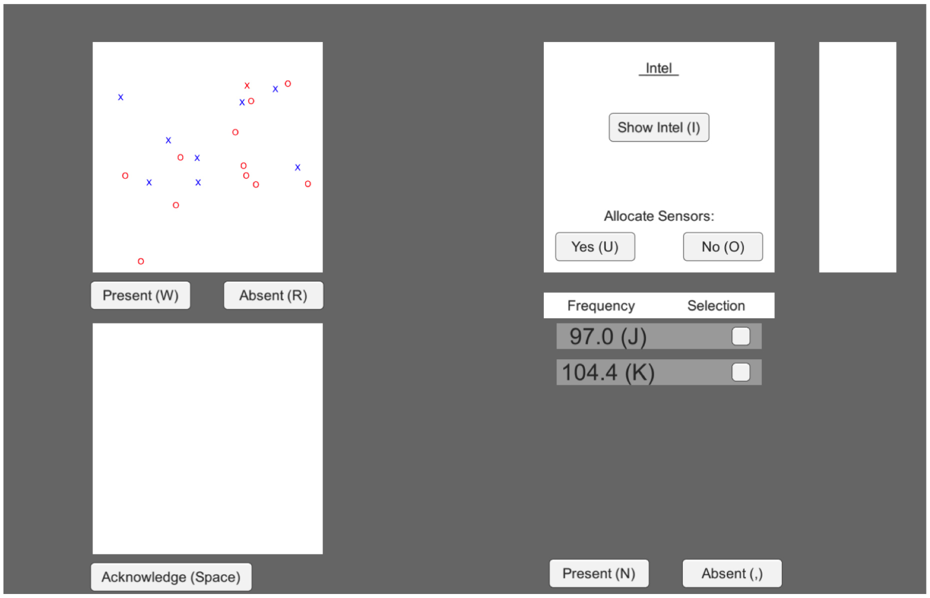


Fig. 1: An example depicting the four ISR-MATB tasks in a single interface.

with the letter unacknowledged. The next two tasks reference this letter and color, so the agent is instructed to remember them.

2.2 Visual Search

The Visual Search task requires that the agent determine if the letter they remember is among a group of many letters that appear on the screen [10]. The other letters are distractors, and may be the same letter as the target with a different color, or the same color as the target with a different letter, or both color and letter different. The target may or may not appear among the distractors, and never appears more than once. The agent pushes a button to indicate whether the the letter is present or absent.

2.3 Auditory Search

The Auditory Search task is very similar to the Visual Search, except of course that the agent must listen instead of look. In this task there are between one and four audio messages that each include a spoken color and letter. If one of the messages is the same as as the letter and color from the first task the agent pushes a button to indicate that it is present, otherwise they indicate that it is absent.

2.4 Decision Making

The final task, Decision Making, requires agents to infer a relationship between the outcomes of the Visual and Auditory Search tasks together with a new binary piece of information called “Intelligence” that appears after choosing whether to hypothetically allocate sensors or not. The rule the agent must guess is not too hard, but it is complex enough that it must be learned by trial-and-error over multiple attempts. Learning the rule is made more difficult by the unlikely but not impossible event that the program responds incorrectly even when a correct answer is given. Responding ‘yes’ or ‘no’ to this sub-task ends one ISR-MATB trial.

3 Ontology Description

In this section we present the Instruction Ontology, a domain ontology built for use with the ISR-MATB experiment platform. This ontology was produced by following the Modular Ontology Modeling (MOM) methodology, outlined in [7,6] MOM is designed to ensure the high quality and reusability of the resulting ontology, both in terms of scope and in terms of granularity, which is a desired outcome.

The ontology consists of six modules: ISR-MATB Experiment, Instruction, SituationDescription, ItemRole, Action, and Affordance. For each module, we describe its purpose, provide a schema diagram,[4] and state its axiomatization in both description logic syntax and natural language. The OWL file for this ontology can be found online[5] as well as the official documentation.[6] Figure 5 shows the schema diagram for the entire ontology.

3.1 ISR-MATB Experiment.

The ISR-MATB Experiment module is the core module for the ontology. The two main classes are ISR-MATB Experiment and ISR-MATB Task. As noted in Section 2, an experiment consists of up to four tasks that may require that information be carried between them, where each Task resides in a specific quadrant of the interface. Each Task provides roles to different Items, as well as a set of Instructions for the agent to carry out. We discuss these classes in more detail in their respective Module sections. The schema diagram for this module is shown in Figure 2c.

[4] A schema diagram is an informal, but intuitive way for conveying information about the structure and contents of an ontology. We use a consistent visual syntax for convenience, detailed in Figure 2.

[5] See `https://raw.githubusercontent.com/undiffagents/uagent/develop/ontology/uagent.owl`.

[6] See `https://daselab.cs.ksu.edu/content/domain-ontology-instruction`

Axiomatization:

$$\top \sqsubseteq \forall \mathsf{affords.Affordance} \quad (1)$$
$$\mathsf{ISR\text{-}MATBTask} \sqsubseteq \geq 1\ \mathsf{hasInstruction.Instruction} \quad (2)$$
$$\mathsf{ISR\text{-}MATBExperiment} \sqsubseteq \leq 4\ \mathsf{hasTask.ISR\text{-}MATBTask} \quad (3)$$
$$\top \sqsubseteq \forall \mathsf{hasLocation.Location} \quad (4)$$
$$\top \sqsubseteq \forall \mathsf{hasName.xsd{:}string} \quad (5)$$
$$\mathsf{ISR\text{-}MATBTask} \sqsubseteq =1\ \mathsf{hasName.xsd{:}string} \quad (6)$$
$$\mathsf{ISR\text{-}MATBTask} \sqsubseteq \forall \mathsf{providesRole.ItemRole} \quad (7)$$
$$\mathsf{ISR\text{-}MATBTask} \sqsubseteq \forall \mathsf{informs.ISR\text{-}MATBTask} \quad (8)$$

Explanation of axioms above:

1. Range. The range of affords is Affordance.
2. Minimum Cardinality. An ISR-MATBTask has at least one Instruction.
3. Maximum Cardinality. An ISR-MATBExperiment consists of at most four ISR-MATBTasks.
4. Range. The range of hasLocation is Location.
5. Range. The range of hasName is xsd:string.
6. Scoped Range. The range of providesRole is ItemRole when the domain is ISR-MATBTask.
7. Scoped Range. The range of informs is ISR-MATBTask when the domain is ISR-MATBTask.

3.2 Action

The Action module is an instantiation of the Explicit Typing meta-pattern described in [9].[7]

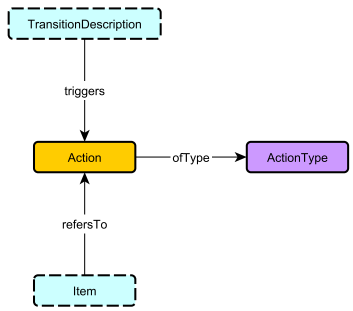
In this case, we use a class, ActionType, to represent a controlled vocabulary. We believe that using a controlled vocabulary to represent this type information is less invasive to the ontology. This way, adding or removing types of actions from the controlled vocabulary does not actually change the ontology. Some instances of the controlled vocabulary are listed in Figure 5.

An Action, in this context, is the physical, actual action that takes place to transition between different states of the experiment, e.g. 'the action of clicking a button.' The schema diagram for this module is shown in Figure 2a.

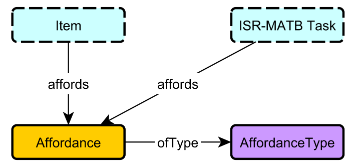
Axiomatization:

$$\mathsf{Action} \sqsubseteq =1 \mathsf{ofType.ActionType} \quad (1)$$

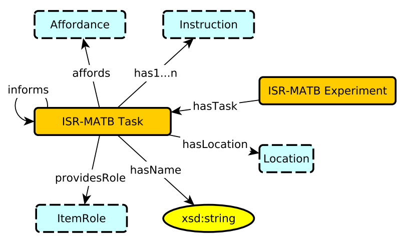
[7] [9] is a modular ontology design library; it contains a set of frequently used patterns and respective documentation.



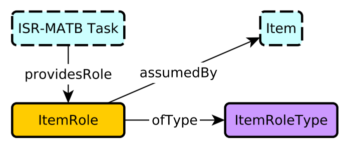
(a) The schema diagram for the Action module.



(b) The schema diagram for the Affordance module.



(c) The schema diagram for the ISR-MATB Experiment module.



(d) The schema diagram for the ItemRole module.

Fig. 2: Orange boxes are classes and indicate that they are central to the diagram. Blue dashed boxes indicate a reference to another diagram, pattern, or module. Gray frames with a dashed outline contain modules. Arrows depict relations and open arrows represent subclass relations. Yellow ovals indicate data types (and necessarily, arrows pointing to a datatype are data properties). Finally, purple boxes represent controlled vocabularies. That is, they represent a controlled set of IRIs that are of that type.

Explanation of axioms above:

1. Exact Cardinality. An Action has exactly one ActionType.

3.3 Affordance

The Affordance module is also instantiated from the Explicit Typing meta-pattern, explained in more detail in Section 3.2 and [9]. An Affordance is essentially some quality of an Item that indicates that "something" may be done with it. Familiar examples might include *clickable* buttons or text highlighted in blue (perhaps indicating that it's a hyperlink). Instances of the AffordanceType can be found in Figure 5. The schema diagram for this module is shown in Figure 2b.

Axiomatization:

$$\text{Affordance} \sqsubseteq {=}1\text{hasAffordanceType.AffordanceType} \quad (1)$$

Explanation of axioms above:

1. Exact cardinality. An Affordance has exactly one AffordanceType.

3.4 ItemRole

The ItemRole module is an instantiation of the AgentRole pattern, which may also be found in [9]. We also equip it with an explicit type, in the same manner as Action and Affordance.

Each ISR-MATB Task may provide roles to Items. That is, certain items may be a target or distractor, but not always. This allows us to assign certain roles to items that may, if they were qualities, be ontologically disjoint. The schema diagram for this module is shown in Figure 2d.

Axiomatization:

$$\text{ISR-MATBTask} \sqsubseteq \forall\text{providesRole.ItemRole} \quad (1)$$
$$\top \sqsubseteq \forall\text{hasItemRoleType.ItemRoleType} \quad (2)$$
$$\text{ItemRole} \sqsubseteq \forall\text{assumedBy.Item} \quad (3)$$
$$\text{ItemRole} \sqsubseteq \exists\text{assumedBy.Item} \quad (4)$$

Explanation of axioms above:

1. Scoped Range. The range of providesRole is ItemRole when the domain is ISR-MATBTask.
2. Range. The range of hasItemRoleType is ItemRoleType.
3. Scoped Range. ItemRoles are assumedBy Items.
4. Existential. Every ItemRole is assumedBy an Item.

3.5 SituationDescription

For this module, we opted to use the Situation and Description approach. We chose to use this conceptualization due to the non-linear nature of the instructions.[8] That is, an ISR-MATB Task is not a sequence of instructions, but a collection of directions or descriptions.

An Instruction, is a description of a way to transition between two states. In order to follow out an instruction the state described in the the pre-SituationDescription would need to be met. Following through would result in a new state, the Post-Situtation Description.

[8] For a deeper discussion on Descriptions, Situations, and Plans, see [4].

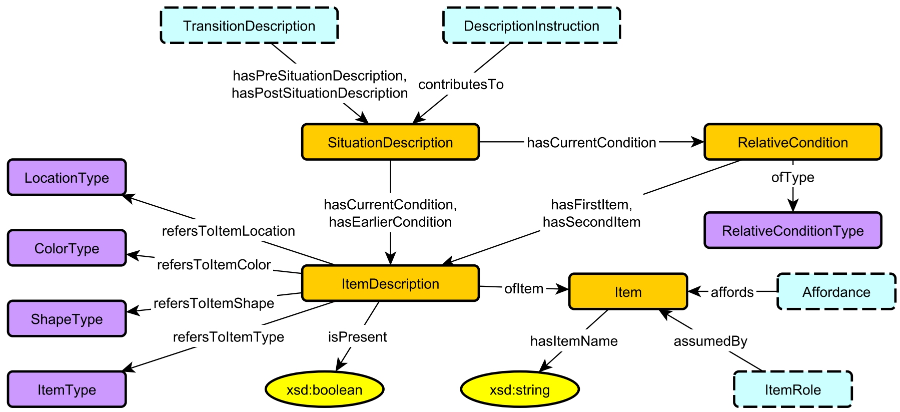


Fig. 3: The schema diagram for the SchemaDiagram module. Color and shape usage is the same as in previous diagrams.

Furthermore, the SituationDescription will indicate the presence, or absence, of an item, as well as its description. Descriptions, in this case, are relegated to controlled vocabularies in the same manner as Affordance or Action. We call this an ItemDescription because it is inherent to the Instruction and not the Item, itself.

The schema diagram for this module is shown in Figure 3.

Axiomatization:

$$\mathsf{SituationDescription} \sqsubseteq \forall \mathsf{hasCurrentCondition}.(\mathsf{RelativeCondition} \sqcup \mathsf{ItemDescription}) \tag{1}$$

$$\mathsf{SituationDescription} \sqsubseteq \forall \mathsf{hasEarlierCondition}.\mathsf{ItemDescription} \tag{2}$$

$$\top \sqsubseteq \forall \mathsf{hasRelativeConditionType}.\mathsf{RelativeConditionType} \tag{3}$$

$$\mathsf{RelativeCondition} \sqsubseteq \forall \mathsf{hasFirstItem}.\mathsf{ItemDescription} \tag{4}$$

$$\mathsf{RelativeCondition} \sqsubseteq \forall \mathsf{hasSecondItem}.\mathsf{ItemDescription} \tag{5}$$

$$\mathsf{ItemDescription} \sqsubseteq \forall \mathsf{ofItem}.\mathsf{Item} \tag{6}$$

$$\mathsf{ItemDescription} \sqsubseteq {=}1\ \mathsf{isPresent}.\mathsf{xsd{:}boolean} \tag{7}$$

$$\top \sqsubseteq \forall \mathsf{refersToItemLocation}.\mathsf{LocationType} \tag{8}$$

$$\top \sqsubseteq \forall \mathsf{refersToItemColor}.\mathsf{ColorType} \tag{9}$$

$$\top \sqsubseteq \forall \mathsf{refersToShapeType}.\mathsf{ShapeType} \tag{10}$$

$$\top \sqsubseteq \forall \mathsf{refersToItemType}.\mathsf{ItemType} \tag{11}$$

$$\mathsf{ItemDescription} \sqsubseteq {\geq}0\ \mathsf{refersToItemLocation}.\mathsf{LocationType} \tag{12}$$

$$\mathsf{ItemDescription} \sqsubseteq {\geq}0\ \mathsf{refersToItemColor}.\mathsf{ColorType} \tag{13}$$

$$\text{ItemDescription} \sqsubseteq \geq 0 \ \text{refersToItemShape.ShapeType} \tag{14}$$

$$\text{ItemDescription} \sqsubseteq \geq 0 \ \text{refersToItemType.ItemType} \tag{15}$$

$$\top \sqsubseteq \forall \text{hasItemName.xsd:string} \tag{16}$$

$$\exists \text{hasItemName}.\top \sqsubseteq \text{Item} \tag{17}$$

Explanation of axioms above:

1. Scoped Range. The range of hasCurrentCondition is a RelativeCondition or ItemDescription when the domain is SituationDescription.
2. Scoped Range. The range of hasEarlierCondition is ItemDescription when the domain is SituationDescription.
3. Range. The range of hasRelativeConditionType is RelativeConditionType.
4. Scoped Range. The range of hasFirstItem is ItemDescription when the domain is RelativeCondition.
5. Scoped Range. The range of hasSecondItem is ItemDescription when the domain is RelativeCondition.
6. Scoped Range. The range of ofItem is Item when the domain is ItemDescription.
7. Scoped Range. An ItemDescription has exactly one Boolean flag indicating whether or not it is present.
8. Range. The range of refersToItemLocation is LocationType.
9. Range. The range of refersToItemColor is ColorType.
10. Range. The range of refersToItemShape is ShapeType.
11. Range. The range of refersToItemType is ItemType.
12. Structural Tautology. An ItemDescription may refer to a LocationType.
13. Structural Tautology. An ItemDescription may refer to a ColorType.
14. Structural Tautology. An ItemDescription may refer to a ShapeType.
15. Structural Tautology. An ItemDescription may refer to an ItemType.
16. Range. The range of hasItemName is xsd:string.
17. Domain Restriction. The domain of hasItemName is restricted to Items.

3.6 Instruction

Instructions are the atomic units of a task. They come in two varieties: descriptions and actions. The former are instructions that are prescriptive or descriptive. They are statements that indicate information about the environment or the task. They may, in natural language, take such form as "There is a button named 'Present'." The latter type of instruction instructs when or where to do something. For example, "Press the button if a high-pitched tone is heard." An Action-Instruction prescribes some transition between descriptions of situations, whereas Description-Instructions directly contribute to said SituationDescription. The module also uses a data property to capture the natural language formulation of the Instruction. The schema diagram for this module is shown in Figure 4.

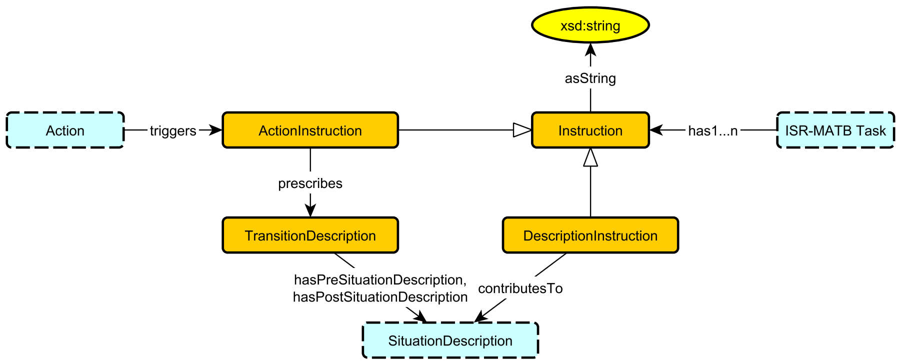


Fig. 4: The schema diagram for the Instruction module. Color and shape usage is the same as in previous diagrams.

Axiomatization:

$$\mathsf{ActionInstruction} \sqsubseteq \mathsf{Instruction} \quad (1)$$

$$\mathsf{ActionInstruction} \sqsubseteq \forall \mathsf{prescribes.TransitionDescription} \quad (2)$$

$$\top \sqsubseteq \forall \mathsf{asString.xsd{:}string} \quad (3)$$

$$\mathsf{Instruction} \sqsubseteq \geq 0\ \mathsf{asString.xsd{:}string} \quad (4)$$

$$\mathsf{DescriptionInstruction} \sqsubseteq \mathsf{Instruction} \quad (5)$$

$$\mathsf{DescriptionInstruction} \sqsubseteq \forall \mathsf{contributesTo.SituationDescription} \quad (6)$$

$$\top \sqsubseteq \forall \mathsf{hasPreSituationDescription.SituationDescription} \quad (7)$$

$$\top \sqsubseteq \forall \mathsf{hasPostSituationDescription.SituationDescription} \quad (8)$$

Explanation of axioms above:

1. Subclass. Every ActionInstruction is an Instruction.
2. Scoped Range. The range of prescribes is TransitionDescription when the domain is ActionInstruction.
3. Range. The range of asString is xsd:string.
4. Structural Tautology. An Instruction may have a string representation.
5. Subclass. Every DescriptionInstruction is an Instruction.
6. Scoped Range. The range of contributesTo is SituationDescription when the domain is DescriptionInstruction.
7. Range. The range of hasPreSituationDescription is SituationDescription.
8. Range. The range of hasPostSituationDescription is SituationDescription.

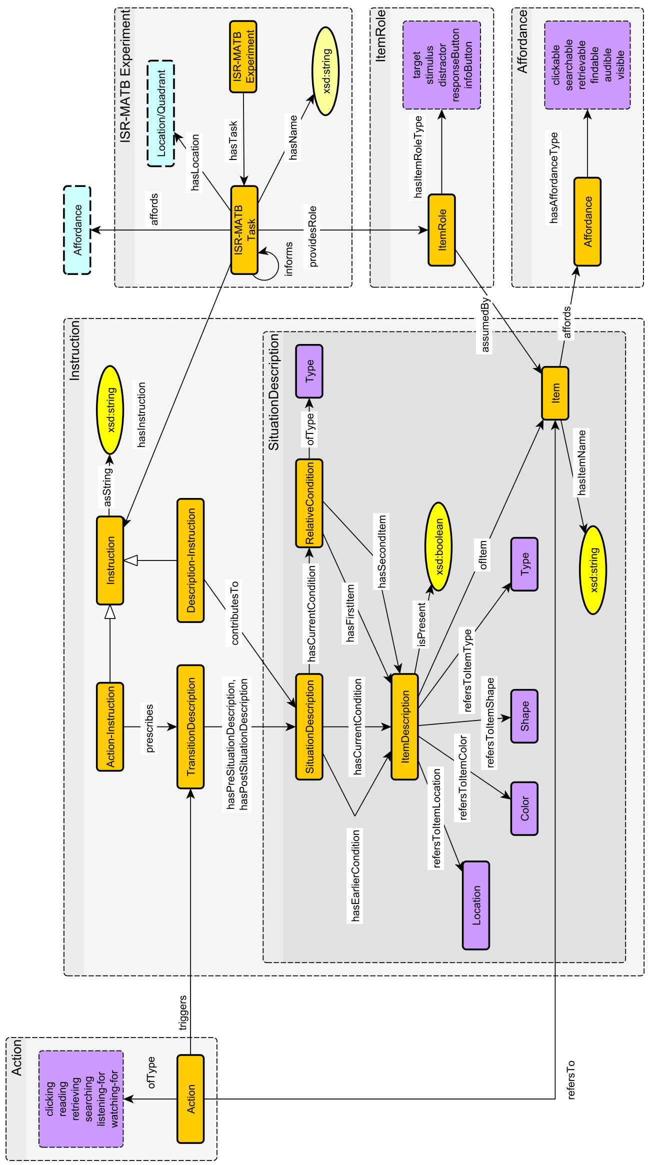


Fig. 5: The schema diagram for the entire ontology. Note that the SituationDescription module is nested in the Instruction Module. Color and shape usage is the same as in previous diagrams.

4 Conclusion

In this paper we have presented an ontology for modeling the ISR-MATB cognitive agent task instructions. This ontology can be used, as we have, to directly support the memory of a cognitive agent performing tasks. It also could support experiment design, irrespective of any agent, by providing a structured basis for evaluating similar tasks. The modular structure facilitates adapting the ontology to other use cases and scenarios by replacing or adapting the existing modules. It is also possible to create new modules from the referenced patterns via template-based instantiation [5].

4.1 Future Work

In the future we plan to extend this ontology so that it can support a fully undifferentiated agent. This will include tasks like ISR-MATB, but also many others that could be very different. One such task is supporting materials science research that uses the Autonomous Research System (ARES) framework. An undifferentiated cognitive agent could operate a robotic system that performs research, using software like ARES, saving materials researchers hours of potentially hazardous lab work.

Acknowledgement This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0386.

References

1. Anderson, J.R.: How can the human mind occur in the physical universe? Oxford University Press (2007)
2. Dinges, D.F., Powell, J.W.: Microcomputer analyses of performance on a portable, simple visual rt task during sustained operations. Behavior research methods, instruments, & computers 17(6), 652–655 (1985)
3. Frame, M., Lopez, J., Myers, C., Stevens, C., Estepp, J., Boydstun, A.: Development of an autonomous management system for human-machine teaming with multiple interdependent tasks. In: Presented to the Annual Meeting of the Psychonomic Society Conference, Montreal, QC, Canada, November 2019 (2019)
4. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE – OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Lecture Notes in Computer Science, vol. 2888, pp. 689–706. Springer (2003)
5. Hammar, K., Presutti, V.: Template-based content ODP instantiation. In: Hammar, K., Hitzler, P., Krisnadhi, A., Lawrynowicz, A., Nuzzolese, A.G., Solanki, M. (eds.) Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the 7th edition of the Workshop on Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, 18th October 2016]. Studies on the Semantic Web, vol. 32, pp. 1–13. IOS Press (2016), https://doi.org/10.3233/978-1-61499-826-6-1

6. Hitzler, P., Krisnadhi, A.: A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. CoRR abs/1808.08433 (2018), http://arxiv.org/abs/1808.08433
7. Krisnadhi, A., Hitzler, P.: Modeling with ontology design patterns: Chess games as a worked example. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns – Foundations and Applications, Studies on the Semantic Web, vol. 25, pp. 3–21. IOS Press (2016)
8. Newell, A.: Unified theories of cognition. Harvard University Press (1994)
9. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: A modular ontology design library. In: WOP@ISWC. CEUR Workshop Proceedings, vol. 2459, pp. 47–58. CEUR-WS.org (2019)
10. Treisman, A.M., Gelade, G.: A feature-integration theory of attention. Cognitive psychology 12(1), 97–136 (1980)