We, Vertiport 6, Are Temporarily Closed: Interactional Ontological Methods for Changing the Destination

Seungwan Woo[1], Jeongseok Kim[2] and Kangjin Kim[1,*]

[1]Department of Drone Systems, Chodang University, Jeollanam-do, Korea
[2]AIX, SK Telecom, Seoul, Korea

Abstract

This paper presents a continuation of the previous research on the interaction between a human traffic manager and the UATMS. In particular, we focus on the automation of the process of handling a vertiport outage, which was partially covered in the previous work. Once the manager reports that a vertiport is out of service, which means landings for all corresponding agents are prohibited, the air traffic system automates what it has to handle for this event. The entire process is simulated using Knowledge Representation and Reasoning to describe the detailed process of reasoning about UAM operations. Moreover, two distinct perspectives are respected for the human supervisor and the management system, and related ontologies and rules are introduced. We believe that applying non-monotonic reasoning can verify each step of the process and explain how the system works. After a short introduction with related works, this paper continues with problem formulation, primary solution, discussion, and conclusions.

Keywords

UAM, UATM, KRR, Answer Set Programming, Articulating Agent

1. Introduction

Urban Air Mobility (UAM) is a novel and rapidly expanding mode of transportation. Utilizing urban airspace is viewed as a crucial solution to traffic problems in densely populated areas. The concept of a vertical airport, or vertiport for short, allows small aircraft to take off and land without runways around tiny structures. There is increased pressure to standardize the UAM air traffic management system, or UATM for short, despite its increasing popularity as a new mode of transportation. At the same time, this becomes problematic because it is difficult to accommodate everyone's desires. Introducing a new system into an established environment sounds like a challenge. Specifically, the airspace should be shared with the existing aircraft system. By dividing the required altitude, potential conflicts between the two methods are avoided. We aim to have this new transportation system operational by 2035, despite the difficulties associated with low-altitude travel and heavy air traffic. This study investigates how each vertiport and the UATM communicate with one another. We are primarily interested in discussing what vertiport administrators and the UATM do and how they respond to incidents. By constructing a graph-based model of the traffic system, we simply characterize the system and provide an illustration

RobOntics 2023: Workshop on Ontologies in Autonomous Robotics, August 28, 2023, Seoul, South Korea
*Corresponding author.
wo1998@naver.com (S. Woo); jeongseok.kim@sk.com (J. Kim); kangjinkim@cdu.ac.kr (K. Kim)

of a common occurrence at a vertiport. In the previous paper, we took the initial step toward our objective. The primary difference between the previous one and this one is that the previous one believed certain corridors should be avoided when determining where the plane should go, whereas this one believes the current target should be avoided. This implies that the route must be altered.

1.1. Related Works

The papers [1, 2, 3, 4] give an in-depth examination of UAM technology, regulatory context, possible advantages, and obstacles. Recently the article in [5] suggests a layered system for managing airspace for UAM vehicles, with different layers defined based on the type of vehicle, level of automation, and altitude. The paper [6] discusses a variety of ATM projects in German Aerospace Center (DLR) initiatives. The paper [7] suggests a collision risk assessment model between a vehicle and obstacles. The paper [8] discusses state-of-the-art Deep Learning (DL) solutions for ATM in depth.

However, none of previous attempts to avoid crashes based on ATM systems are ready to be widely adopted because of complex UATM systems, various stakeholders, mass agent activities, and possibly catastrophic events. Considering multiple stakeholders, our initial work [9] was to detour agents' route, avoiding particular corridors. In that paper, nonmonotonic reasoning was used in the operation of the complex UATM system to describe how to control unexpected cases or easily changeable information.

In this regard, the paper [10] serves as a valuable guide for the explanation of complex systems—explainable AI (or XAI for short). To validate the complex system, their proposed epistemological model is based on knowledge representation and reasoning in particular. However, their approach is philosophical. Hence, as a concrete, motivating use case that can be easily adapted to our case, it was not sufficient.

The paper [11] examines a method for developing automatic classifiers capable of offering explanations based on an ontology. The paper [12] briefly reviews five approaches based on association rule mining, formal concept analysis, inductive logic programming, computational learning theory, and neural networks, and provides how to build descriptive logic (DL) ontologies. Even though these papers provide a basis on which reasoning can be linked to the ontology, the use of their approaches in our system has not been a practical solution.

For the practical point of view, the paper [13] provides a framework for intra-logistics problems based on ASP. The papers [14, 15] propose to perform task assignment and vehicle routing by means of ASP for automated guided vehicles (AGVs). While they concentrate on showing how to calculate their path and give a series of subtasks, we concentrate on the logic utilized in the process of interaction between a human manager and the system, as well as across systems.

The paper [16] provides a systematic survey for projects at the intersection of ontologies and autonomous robotics and compares them according to what terms are defined in their ontologies, how these terms are used to support different cognitive capabilities of a robot, and in which application domain they are used. However, the authors indicate that ontologies for behaviors and interaction have been relatively less focused than other terms in several scopes.[1] In order

[1] For more details, see Table 1 in [16].

to illustrate the interaction, rather than solely conceptualizing in ontologies, we chose a hybrid approach (as the approach introduced in [17]) combining the ontology with the reasoning process by ASP. While the paper [17] focuses on the user-centered explanation and the generalization itself, we focus on the trace of reasoning to provide facts collected by each agent's information in real time, which is dynamically changeable.[2]

1.2. Contributions

As an illustration, we present a destination-changing scenario that may occur frequently in UAM environments. Then, we elaborate on our demonstration of nonmonotonic reasoning and explainability.

1.3. Outlines

In the following sections, we describe the problem formulation, provide the solution, continue discussions, and conclude this paper.

2. Problem Formulation

Examine the graph depicted in Fig. 1. Each vertex denotes a vertiport, while the edges denote corridors. Large circles represent UAM Air Traffic Management (UATM) systems covering these vertiports. These corridors transport agents between this network's vertiports. UATMs can communicate directly with agents within their coverage area. This study presumes that the 'UATM Network' possesses a communication relaying protocol that enables UATMs to relay messages to one another. Upon the agent's departure from this UATM Network, direct communication becomes impossible.

Each agent typically commences a voyage as follows: It begins in a vertiport, where preparations for flight are made. This agent requests departure from the vertiport traffic manager. After receiving permission, the agent enters the airspace.

The destination and route are configured prior to flight. The agent uses UATMs to update its velocity, GPS coordinates, and other information as it travels through corridors to the destination vertiport. UATMs can then supervise the agent while considering traffic.

The agent requests landing from the vertiport traffic manager when close.

Typically, the traffic manager directs it to the vertipad after confirmation. This traffic system operates autonomously. Managers of human traffic are in control of each vertical airport. Monitoring traffic, vertiports, and legacy traffic systems is their responsibility. They improve the system's adaptability.

2.1. A Vertiport Closure Scenario

The human manager's perspective: While a human manager in vertiport 6 (or vp_6 for short) monitored landing some agents, he or she found that there was a safety issue. It appears that

[2]We note that specifying the concept of 'real time' in this paper is out of scope.

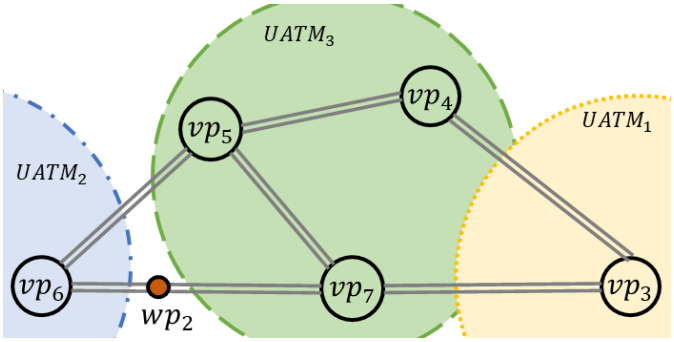

Figure 1: Partly sampled UATM Network composed of five vertiports ranging from vp_3 to vp_7, bidirectionally connected corridors between adjacent vertiports, three UATMs ranging from $UATM_1$ to $UATM_3$, and their coverage indicated by outside circles.

approving landing requests from the following agents would be risky. Hence, he or she reported this status to the UATM (here, $UATM_2$). In particular, the vp_6 is temporarily closed, so it is required for all vp_6 heading agents to change their destination to vp_5, which is the candidate vertiport of the vp_6.

The $UATM_2$'s perspective: The $UATM_2$ received a report that the vp_6 is closed. Since all vp_6 heading agents should detour their route to the new destination, it should find all the related agents. While finding them, it also looks up other UATMs. Once it finds all, it checks the vp_6's candidate vertiport. We assume that the candidate vertiports are carefully chosen, depending on previous records, including congestion, landing turnover ratio, and capacity. After getting the candidate vertiport and computing the route to the new destination, it sends to all these agents the replaced path, which has the newly appended route at the end. Once all these agents' paths are changed, the $UATM_2$ replies back to the vp_6 human manager.

3. Solution

This section presents our hybrid approach depicted in Fig. 2. This approach consists of the following two steps:

1. The problem is declared in Answer Set Programming. Goals and rules are defined, and each interaction is addressed.
2. Composed ontology is used in this problem. As a semantic layer, we develop the ontology model against a unified data source. Class and property information for essential classes such as Human Traffic Managers, Agents, and UATMs provide additional information for reasoning in this layer. Then, relations are constructed for these classes based on the predicates used in the ASP area.

The flow we covered in the scenario is as follows: Once the human manager reports the vertiport closure, the reasoning component computes the necessary process and returns the detailed result. The result is then updated in the ontology component. Then a human manager can trace the result through the ontology.

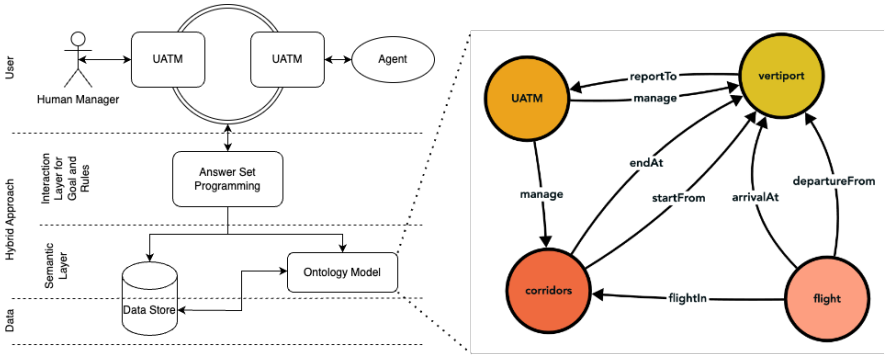

Figure 2: Conceptual diagram of hybrid approach using ontology model and ASP. Through the relationship between each entity on semantic layer, it is possible to infer the UATM associated with the vertiport and the flight information managed by the UATM.

Table 1: Each Predicate and Description

Predicate	Description
`uatm/1`	a set of UATMs
`agent/1`	a set of agents
`vp/1`	a set of vertiports
`edge/2`	a set of corridors
`cover/2`	the relation in which the UATM covers which vertiport
`edge_range/3`	each corridor's distance bound
`covered_wp/4`	the coverage provided by each UATM for each corridor
`candidate_vp/2`	a pair of vertiports: a vertiport and its candidate vertiport for backup
`step/1`	a range of step
`loc/4`	an agent's location at a step
`plan/4`	an agent's plan to move one step at a time
`source/3`	the initial vertiport of an agent
`target/3`	the final vertiport of an agent

3.1. Common Setting

The whole approach begins with two common settings for the world and the agents. Due to the page limitation, we here only introduce the used predicates in Table 1.[3]

3.2. Queries

In order to make the contents more structured, we will enumerate the queries with the codes first and then provide the results. We note that each code is generalized. Hence, when executing the

[3]The complete code listings are available as a technical report: https://arxiv.org/abs/2307.03558. We note that, for simplicity, we now intentionally omit the arity part for each predicate. For example, we will write `edge_range/3` to `edge_range`.

code, instead of vp_6, other vertiports can be specified.[4]

Find all vp_6 heading agents within the UATM network: In order to answer this query, there is an initial predicate `covered_agent` in Code 1. This rule is essential for determining whether each agent is covered by the UATM network. Additionally, it searches only agents whose target is vp_6. Then, as shown in Table 2, `covered_by_uatm2` and `covered_by_other` present five agents: 3, 5, 6, 1, and 2.

Code 1: Find all vp_6 heading agents within the UATM network

```
covered_uatm(TM) :- cover(TM, V1), V1 == vp.
covered_agent(A, TM) :- loc(A, T, U, V, WP), covered_wp(U, V, TM, WP), target(A, T, V1), V1 == vp.

covered_by_target_uatm(A) :- covered_agent(A, TM), covered_uatm(TM).
covered_by_other(A) :- loc(A, T, U, V, _), covered_agent(A, TM), covered_uatm(TM1), TM != TM1.

trigger_query :- covered_agent(A, TM).
covered :- 1 <= #count{A: covered_by_target_uatm(A); A:covered_by_other(A)}.
:- trigger_query, not covered.

#show loc/5. #show covered_uatm/1. #show covered_by_target_uatm/1.
#show covered_by_other/1.
```

Change the destination of the agents we found: Since we already found these agents from Code 1, the next procedure for the $UATM_2$ has the followings:

- it finds the candidate vertiport for the vp_6,
- it creates the new plans for the relayed agents: Here, relayed agents include agents covered by other UATMs as well as agents covered by $UATM_2$,
- it appends the new plan to the end of the existing plan for these agents,
- it sends target change requests to other UATMs, and
- once the plan is ready and target change requests are received, UATMs do the target change for these agents.

The vp_6 human manager, then, is known by the `target_change` as shown in the Table 2.

Handle a landing request from agent 4: Consider the vp_6 human manager received a landing request from agent 4 shortly after he or she informed the `target_change`. The agent 4 was in waypoint 2 (or wp_2 for short) in Fig. 1 when $UATM_2$ searched all the vp_6 heading agents. We remark that even though every inch in each corridor is guaranteed to be within the coverage of the UATM network, relaying the message for each agent may take some time, and some of the agents may send the landing request before their targets are changed. Shortly after the search, let us assume that agent 4 moves closer to vp_6 and into coverage. In particular, suppose it is located at `loc(4, 2, 7, 6, 17)`.[5]

[4]With the '-c' command option in clingo, it is possible to insert a specific value into the code. For example, to designate vp_6 as the vertical airport of interest, Code 1 can be executed as follows: $ clingo world.lp agent.lp code1.lp -c vp=6. This option replaces the vp in the first two lines of Code 1 with the value 6.

[5]This indicates that in step 2, agent 4 located itself at a quantified position, say 17, along the corridor between vp_7 and vp_6. Note that the quantified position for the corridor has been bounded at `edge_range` (see Table 1), and this additional agent location is provided as a separate input file to keep the program general.

Code 2: Change the destination of the found agents

```
relayed(A) :- covered_by_target_uatm(A).
related(A) :- covered_by_other(A).

new_plan(A, T+1, V, V1) :- plan(A, T, U, V), target(A, T, V), V == vp, relayed(A),
↪  candidate_vp(V, V1), step(T+1), not new_plan(A, T, V, V1).
target_change_request(A, T) :- relayed(A), new_plan(A, T, V, V1).

plan(A, T+1, V, V1) :- plan(A, T, U, V), target(A, T, V), new_plan(A, T+1, V, V1), step(T+1).
plan(A, T+1, U, V) :- plan(A, T, U, V), step(T+1).

target_change(A, T) :- plan(A, T, U, V), new_plan(A, T, U, V), target_change_request(A, T).
:- not target_change(A, T), new_plan(A, T, U, V), target_change_request(A, T).

#show relayed/1. #show new_plan/4. #show target_change_request/2.
#show target_change/2.
```

Code 3: Handle the landing request from agent 4

```
vp_heading_agent_number(N) :- N = #count{A:target(A, T, V), V==vp}.

covered_wp(U, V, TM, WP).
landing_request(A, T+1, V) :- not target(A, T+1, _), target(A, T, V), loc(A, T+1, U, V, WP), V ==
↪  vp, covered_wp(U, V, TM, WP).

new_plan(A, T+1, V, V1) :- plan(A, T, U, V), landing_request(A, T, V), candidate_vp(V, V1),
↪  step(T+1), not new_plan(A, T, V, V1).
plan(A, T+1, V, V1) :- plan(A, T, U, V), landing_request(A, T, V), new_plan(A, T+1, V, V1),
↪  step(T+1).

target_change_request(A, T+1) :- landing_request(A, T, V), new_plan(A, T+1, V, V1).
plan(A, T+1, V, V1) :- plan(A, T, U, V), landing_request(A, T, V), new_plan(A, T+1, V, V1),
↪  target_change_request(A, T+1), step(T+1).
- not target_change(A, T+1), landing_request(A, T, V), step(T+1).

#show vp_heading_agent_number/1. #show target_change_request/2.
#show target_change/2. #show landing_request/3.
```

In the vp_6 manager's perspective, the landing request would not be welcome. However, after checking that its target hasn't been updated, he or she can just report to the $UATM_2$ that the vp_6 is closed again.

In the $UATM_2$'s perspective, since agent 4 is in the coverage of $UATM_2$ now, handling the `landing_request` in Code 3 is not much different from what we have seen in Code 1 and Code 2. The `target_change(4, 3)` in Table 2 explains that agent 4 has a new target at step 3.

4. Discussion

We here discuss current progress, limitations, and future directions.

Nonmonotonicity: Through the entire program, the initial background knowledge given is that vertiport 6 is temporarily closed. Hence, this is a given fact. In order to illustrate the nature of nonmonotony, agents' locations can vary. However, due to the unified demonstration of the codes and the result, the locations of each agent are not selectable while running the system model.

Table 2: Each Query and Result: Each query is declared in the code. Its running result is shown in the next column of the query. The results of the corresponding queries also keep the results of their predecessors without changing or removing them.

Query	Result
Find all vp_6 heading agents within the UATM network:	`covered_by_other(3) covered_by_other(5) covered_by_other(6)` ↪ `covered_by_target_uatm(1) covered_by_target_uatm(2)` ↪ `covered_uatm(2) loc(1,1,7,6,20) loc(2,1,7,6,18)` ↪ `loc(3,1,7,6,8) loc(4,1,7,6,14) loc(5,1,3,7,17)` ↪ `loc(6,1,7,6,3)`
Change the destination of the agents we found:	`relayed(2) relayed(1) new_plan(1,2,6,5) new_plan(2,2,6,5)` ↪ `target_change_request(2,2) target_change_request(1,2)` ↪ `target_change(2,2) target_change(1,2)`
Handle a landing request from agent 4:	`loc(4,2,7,6,17) landing_request(4,2,6) new_plan(4,3,6,5)` ↪ `target_change_request(4,3) target_change(4,3)` ↪ `vp_heading_agent_number(6)`

Since all the corresponding rules in the queries we covered have a general declaration, we believe that this can be easily generalized by changing their locations through choice rules.

Explainability: The query asking whether '**Changing the destination of the agents we found**' is successful is represented as the predicate `target_change` and its supporting rule just followed by the predicate. The answer shows that `target_change(A, T)` is true when it is satisfiable. In this query, the validation of the explanation is checked by the combination of the predicate, `target_change`, which is regarded as a fact when the body of the rule is true, and the safe rule, which ensures the fact's consistency. Assuming that all the derived rules and relationships lead to the body of the rule as true, the logical consequence makes the predicate `target_change` by also being connected, and this justifies the answer to the query. We note that the explanation is somewhat abstract depending on the ontological features, and the deeper explanation is desirable.

Future Directions: The research related in the explanation of the complex system is in early stage yet. More scenarios are necessary. Agent movement should be considered more carefully, and the scenario can be reactively simulated with proper foundations. We hope to deliver a comprehensive air traffic management system in the near future.

5. Conclusions

We have described a scenario involving UATMs. Through knowledge representation and reasoning, the scenario was simulated by expressing the predicates in ontology. Applying the Answer Set Programming framework, queries and their responses were examined from two distinct perspectives: those of a human manager and the UATM system. The discussion then turned to nonmonotonicity and explainability.

Acknowledgments

This work is supported by the Korea Agency for Infrastructure Technology Advancement(KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2022-00143965).

References

[1] C. Reiche, R. Goyal, A. Cohen, J. Serrao, S. Kimmel, C. Fernando, S. Shaheen, Urban Air Mobility Market Study, National Aeronautics and Space Administration (NASA) (2018). URL: https://escholarship.org/uc/item/0fz0x1s2. doi:http://dx.doi.org/10.7922/G2ZS2TRG.

[2] L. A. Garrow, B. J. German, C. E. Leonard, Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research, Transportation Research Part C: Emerging Technologies 132 (2021) 103377. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21003788. doi:https://doi.org/10.1016/j.trc.2021.103377.

[3] U. T. Korea, K-uam concept of operations, v1.0 (2021).

[4] O. A. Marzouk, Urban air mobility and flying cars: Overview, examples, prospects, drawbacks, and solutions, Open Engineering 12 (2022) 662–679. URL: https://doi.org/10.1515/eng-2022-0379. doi:doi:10.1515/eng-2022-0379.

[5] F. A. Administration, Faa's urban air mobility (uam) concept of operations version 2.0, 2023. URL: https://www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20%28UAM%29%20Concept%20of%20Operations%202.0_0.pdf.

[6] B. I. Schuchardt, D. Geister, T. Lüken, F. Knabe, I. C. Metz, N. Peinecke, K. Schweiger, Air traffic management as a vital part of urban air mobility – a review of dlr's research work from 1995 to 2022, Aerospace 10 (2023). URL: https://www.mdpi.com/2226-4310/10/1/81. doi:10.3390/aerospace10010081.

[7] D. Kim, K. Lee, Surveillance-based risk assessment model between urban air mobility and obstacles, Journal of the Korean Society for Aviation and Aeronautics 30 (2022) 19–27. URL: https://doi.org/10.12985/ksaa.2022.30.3.019. doi:10.12985/ksaa.2022.30.3.019.

[8] E. C. Pinto Neto, D. M. Baum, J. R. d. Almeida, J. B. Camargo, P. S. Cugnasca, Deep learning in air traffic management (atm): A survey on applications, opportunities, and open challenges, Aerospace 10 (2023). URL: https://www.mdpi.com/2226-4310/10/4/358. doi:10.3390/aerospace10040358.

[9] J. Kim, K. Kim, Agent 3, change your route: possible conversation between a human manager and UAM Air Traffic Management (UATM), in: Robotics: Science and Systems (RSS) workshop on Articulate Robots: Utilizing Language for Robot Learning, Daegu, Korea, 2023. URL: https://arxiv.org/abs/2306.14216. doi:https://doi.org/10.48550/arXiv.2306.14216.

[10] J. Borrego-Díaz, J. Galán-Páez, Knowledge representation for explainable artificial intelligence: Modeling foundations from complex systems, Complex & Intelligent Systems 8 (2022). doi:10.1007/s40747-021-00613-5.

[11] G. Bourguin, A. Lewandowski, M. Bouneffa, A. Ahmad, Towards ontologically explainable classifiers, in: I. Farkaš, P. Masulli, S. Otte, S. Wermter (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2021, Springer International Publishing, Cham, 2021, pp. 472–484.

[12] A. Ozaki, Learning description logic ontologies: Five approaches. where do they stand?, KI - Künstliche Intelligenz (2020) 1–11.

[13] M. Gebser, P. Obermeier, T. Otto, T. Schaub, O. Sabuncu, V. Nguyen, T. C. Son, Experimenting with robotic intra-logistics domains, TPLP 18 (2018) 502–519.

[14] M. Gebser, P. Obermeier, T. Schaub, M. Ratsch-Heitmann, M. Runge, Routing driverless transport vehicles in car assembly with answer set programming, TPLP 18 (2018) 520–534.

[15] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, W. Yeoh, Generalized target assignment and path finding using answer set programming, in: IJCAI, ijcai.org, 2017, pp. 1216–1223.

[16] A. Olivares-Alarcos, D. Beßler, A. Khamis, P. Goncalves, M. K. Habib, J. Bermejo-Alonso, M. Barreto, M. Diab, J. Rosell, J. Quintas, J. Olszewska, H. Nakawala, E. Pignaton, A. Gyrard, S. Borgo, G. Alenyà Ribas, M. Beetz, H. Li, A review and comparison of ontology-based approaches to robot autonomy, The Knowledge Engineering Review 34 (2019) e29. doi:10.1017/S0269888919000237.

[17] S. Chari, O. Seneviratne, D. M. Gruen, M. A. Foreman, A. K. Das, D. L. McGuinness, Explanation ontology: A model of explanations for user-centered ai, in: J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web – ISWC 2020, Springer International Publishing, Cham, 2020, pp. 228–243.