Explaining ASP-based Operating Room Schedules

Riccardo Bertolucci[1][0000−0001−7356−1579], Carmine Dodaro[2][0000−0002−5617−5286], Francesco Ricca[2][0000−0001−8218−3178], Giuseppe Galata[3][0000−0002−1948−4469], Marco Maratea[1][0000−0002−9034−2527], and Ivan Porro[3][0000−0002−0601−8071]

[1] University of Genoa, Genova Italy
[2] University of Calabria, Arcavacata, Rende CS, Italy
[3] SurgiQ srl, Genova, Italy

Abstract. The Operating Room Scheduling (ORS) problem refers to the task of assigning patients to operating rooms. An automated solution able to solve the ORS problem in real world scenarios should also be "explainable" to be fully acceptable. Answer Set Programming (ASP) has been applied successifully to solve the ORS problem. However, when the available resources are not enough to satisfy user's requirements (e.g., insufficient number of free beds) the system cannot compute a schedule, and also cannot provide an explanation for that "negative" result.
In this work, we present an extension of the aforementioned ASP-based approach to ORS problem that is also able to provide explanations (in terms of input facts) that caused the absence of solutions. The explanation computation technique builds on the ideas employed by the ASPIDE debugger for ASP programs. Preliminary experimental results show the viability of the approach.

Keywords: Answer set programming · Explainability · Operating Room Scheduling.

1 Introduction

The increasing use of Artificial Intelligence (AI) methods in applications is affecting all parts of our lives and the need for explainable methods is becoming even more important. Explainability is one of the most heavily debated topics when it comes to the application of AI in healthcare. Even though AI-driven systems have been shown to outperform humans in certain tasks, the lack of explainability features continues to spark criticisms (Miller 2019, Das and Rad 2020). For these reason, we focus on improving explainability techniques for transparent models. Due to its rich set of high level language constructs Answer Set Programming (ASP) is a well suited AI language to develop explainable models (Teppan and Zanker 2020, Patil and Främling 2021, Alviano et al. 2020). During the validation of a tool recently-devised to solve the Operating Room Scheduling (ORS) problem (Dodaro, Galatà, et al. 2019), we recognized the need to unfold the decision-making process to an user not specialized in AI. The ORS problem amounts to computing an assignment of patients to beds on operating room, a task that become even more relevant task during the COVID19 pandemic.

The above-mentioned tool for ORS is based on Answer Set Programming (ASP). ASP (Brewka, Eiter, and Truszczyński 2011; Van Harmelen, Lifschitz, and Porter 2008) is a popular AI paradigm for decision-making and problem-solving that has proven useful in a variety of application areas, such as biology (Gebser et al. 2011), psychology (Chaudhri and Inclezan 2015; Balduccini and Girotto 2010), and many others (Erdem, Gelfond, and Leone 2016). ASP is a declarative programming language used to specify a problem in terms of general inference rules and constraints, along with concrete information about the application scenario. Despite the declarative nature of ASP, the development of an explainability layer on top of ASP systems is still subject of research and debate (Fandinno and Schulz 2019). The ORS tool presented in (Dodaro, Galatà, et al. 2019) resulted to be effective in practice in computing schedules, especially when there are enough resources. However, when the available resources are not enough to satisfy user's requirements (e.g., insufficient number of free beds) the system cannot compute a schedule, and also cannot provide an explanation for that "negative" result. This makes the tool not fully acceptable in practice, since neither explanation nor a hint on how to solve the problem is provided.

In this work we present a framework for the generation of the needed explanations. In particular, our approach aims at generating explanations in case the instance modeling the ORS problem is incoherent, hence the system is not able to compute a solution. More in detail, our aim is to isolate the facts in the input that led to the incoherence: this procedure it is called *facts checking*. Furthermore, we make this knowledge available in a readable fashion for inexperienced users such as medical staff. The explanation computation technique builds on the ideas employed by the ASPIDE debugger for ASP programs (Dodaro, Gasteiger, et al. 2019). Preliminary experimental results show the viability of the approach. In the following, we assume the reader is familiar with ASP.

2 Explainability layer

The aim of this work is to integrate the work done and a debugging tool able to perform fact checking. Indeed, our aim is to identify which atoms modeling the input led to the unsatisfiability. For this reason, we extended the capabilities of an already existing tool (Dodaro, Gasteiger, et al. 2019) in order to identify the single or the set of facts atoms that led to the inconsistency of the encoding. Roughly, the tool described in (Dodaro, Gasteiger, et al. 2019) works by adding adornments atoms to the rules of the program, and then detecting a minimal set of adorned atoms that cause the inconsistency (reason for the inconsistency). We build on this idea and, instead of adorning the rules of the program (which is provably correct), we apply the adornment to input facts. Moreover, in order to improve the performances of the system, we perform some preprocessing on the atoms we are testing which consist on the selection of the atoms that are most likely to led to inconsistency. This selection was carried out under some assumptions: (i) The order in which the atoms are checked affect the performances. (ii) The faulty atoms typology can be: *beds(SP,AV,D)*, which defines the beds availability, or *duration(N,O,S)*, which defines duration of operations. Once the atoms are identified, the system will extract the information from the atoms identified and use this information to give to the user the most complete answer on the cause of the impossibility to find a proper schedule.

3 Preliminary test

We tested our framework under the following assumptions: (i) the test were carried out with a single encoding with 5 different configuration of the data: 2 representing the data necessary for the schedule of 5 days of operations and 3 representing the data necessary for the schedule of 10 days of operations, (ii) for each configuration we are measuring the computation time from the beginning to the moment in which the error is found, (iii) the set of facts included in the search are subject to a *shuffle*, in which the atoms are shuffled each time a new search begin, or to a *reverse*, in which the atoms are reversed once instead. Under these assumptions we were able to identify two facts typologies that could lead to the inconsistency of the problem: (a) the lack of available beds in a certain speciality, given by facts of the type *beds(SP,AV,D)*, and (b) the duration of a certain operation is higher than the available time of each operating room of each day in a certain specialty, given by facts of the type *duration(N,O,S)*.

As a sum up of our results, the reasons that can make a fact cause of the incoherence are multiple. In fact, analysing the knowledge represented by each fact, it is possible, for a domain expert, to categorize the source of the incoherence: If the error is related to the number of beds available in a certain speciality SP in the day D the causes of the inconsistency could be: (i) the lack of beds in the speciality in that specific day, (ii) the number of patients to be scheduled in the speciality are too many, (iii) the maximum time for the schedule is too short, or rather it is impossible to schedule all the patients in the amount of given days. If the error is related to the fact that duration N of a certain session S is higher than the available time of each operating room of each day in a certain specialty, the causes of the inconsistency could be: (i) The time that a patient must spend in an operating room is too high, or (ii) the sum of the times that all patients, of a certain specialty (SP), must spend in an operating room is too high.
When all the faulty atoms are isolated, we are able to extract many useful information to retrieve to user in order to let him/her decide which solution fits best such as: the specialty that is causing the fault, the list of patients with high priority in those specialities and all their data (surgery duration, length of the stay, etc), the day in which the fault occurs, and the number of beds available in that day in the given specialty.
With all these information the user can choose the best way to solve the problem, e.g.: increasing the number of beds, increasing the number of maximum days to schedule, decreasing the number of patients. The initial results gathered

after the preliminary test shows that this approach is able to extract information from real-world scenarios data and generate explanations understandable by an inexperienced user.

4 Conclusion

In this paper, we present an approach to explaining the outcome of an ASP-based approach to the problem of operating room scheduling.The objective is to explain why, in a certain situation, no appropriate schedule could be found, in other words, why no answer set could be computed. The preliminary tests show that the presented approach is able to identify the facts which led to the inconsistency of the solution. However, the generation of the explanations is highly dependent on the treated domain: in order to be able to present an explanation comprehensible to inexperienced users, the knowledge represented by the faulty facts must be analysed by a domain expert with a deep understanding of the encoding. As we stated in Section 2, the presented approach is able to find the minimal set of adorned atoms. Since a single faulty fact is enough to lead to the inconsistency, our strategy will be able to find one faulty fact at the time. To overcome this problem and be able to find the complete set of faulty facts, we included in our architecture a module that handles the inconsistency: the process used in this model is highly dependent by the domain. However, the process of finding a single faulty fact is applicable to any domain. However, the explanation generated by our approach take under consideration only the input facts: to generate exhaustive explications it is necessary to include this approach in a more complex system which compute and join explanations from both the set of input facts and the set of rules.

References

[Alv+20] Mario Alviano et al. "Answer Set Programming in Healthcare: Extended Overview." In: *IPS-RCRA@ AI* IA*. 2020.

[BET11] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. "Answer set programming at a glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103.

[BG10] Marcello Balduccini and Sara Girotto. "Formalization of psychological knowledge in answer set programming and its application". In: *Theory and Practice of Logic Programming* 10.4-6 (2010), pp. 725–740.

[CI15] Vinay K Chaudhri and Daniela Inclezan. "Representing states in a biology textbook". In: *2015 AAAI Spring Symposium Series*. 2015.

[Dod+19a] Carmine Dodaro, Giuseppe Galatà, et al. "An ASP-based solution for operating room scheduling with beds management". In: (2019), pp. 67–81.

[Dod+19b] Carmine Dodaro, Philip Gasteiger, et al. “Debugging non-ground ASP programs: Technique and graphical tools”. In: *Theory and Practice of Logic Programming* 19.2 (2019), pp. 290–316.

[DR20] Arun Das and Paul Rad. “Opportunities and challenges in explainable artificial intelligence (xai): A survey”. In: *arXiv preprint arXiv:2006.11371* (2020).

[EGL16] Esra Erdem, Michael Gelfond, and Nicola Leone. “Applications of answer set programming”. In: *AI Magazine* 37.3 (2016), pp. 53–68.

[FS19] Jorge Fandinno and Claudia Schulz. “Answering the “why” in answer set programming–a survey of explanation approaches”. In: *Theory and Practice of Logic Programming* 19.2 (2019), pp. 114–203.

[Geb+11] Martin Gebser et al. “Potassco: The Potsdam answer set solving collection”. In: *Ai Communications* 24.2 (2011), pp. 107–124.

[Mil19] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial intelligence* 267 (2019), pp. 1–38.

[PF21] Minal Suresh Patil and Kary Främling. *Towards Explainable Agency in Multi-Agents Systems Using Inductive Logic Programming and Answer Set Programming.* Tech. rep. EasyChair, 2021.

[TZ20] Erich Teppan and Markus Zanker. “Exploiting Answer Set Programming for Building explainable Recommendations”. In: *International Symposium on Methodologies for Intelligent Systems.* Springer. 2020, pp. 395–404.

[VLP08] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation.* Elsevier, 2008.