Answer Set Planning with Multiple Agents

Marvin Beese, mabeese@uni-potsdam.de, 786300

University of Potsdam

Abstract. Answer Set Programming (ASP) can be successfully used for planning problems in single- and multi-agent environments. This summary examines the planning problem with the action-language $\mathcal{B}$ and gives an overview of the main mechanisms for planning problems in multi-agent environments.

1 Introduction

This summary is based on the article "Answer Set Planning in Single- and Multi-agent Environments" by Tran Cao Son and Marcello Balduccini [1]. In Answer Set Programming (ASP), planning with agents is one of the most discussed applications and therefore there are various concepts for agents interacting alone or with other agents in a planning-environment. In the following I am going to describe the main ideas of answer set planning with the action language $\mathcal{B}$, where definition of the problem domain has priority. Further I am going to set the focus on planning in multi-agent environments. There, centralized and distributed planning are of importance, especially planning with fully collaborative and non- or partially-collaborative agents.

2 Answer Set Planning with Action Languages

In answer set planning, the planning problem is solved with ASP by translating the planning problem P into a logic program $\pi(P, n)$. The integer value n defines the maximum length of solutions to be calculated by this program. There are two parts for the encoding of the logic program $\pi(P, n)$, where the first one is the transitional system of the problem domain and the second one the initial state and the goal specification. The encoding of the goal specification is significant for the generation and verification of solution candidates.
For the definition of the planning problem domain, one can use the planning domain description language or the closely ASP-related action language $\mathcal{B}$. In $\mathcal{B}$, we consider a discrete set of fluents F and a discrete set of actions A. Here, the domain P over (F, A) is defined by a collection of action statements $impossible\_if(a, \psi)$, $causes(a, l, \psi)$ and $if(l, \psi)$, where $a \in A$ are actions, ψ is a set of literals and l is a fluent. The statement $impossible\_if(a, \psi)$ represents that an action a cannot be executed when the set of literals ψ is true, whereas $if(l, \psi)$ means that when ψ is true the fluent l must be true *(state constraint)*. A statement of the form $causes(a, l, \psi)$ states that a can be executed when ψ is

true, which causes l to become true.
A domain that changes over time due to occurrences of actions is called a dynamic domain, which can be represented as a directed graph, with states as nodes and actions as edges. Action languages, which can be used to define those graphs, can be translated easily into ASP syntax, based on their similarities. This is not discussed in this summary, because the focus is set on the basic mechanisms of multi-agent planning. Further information can be found in [1].

3 Multi Agent Planning

In multi-agent planning agents don't act alone in an environment, rather they have to interact with each other. The planning and reasoning activity can be treated here in two possible ways, either using only one main system for a centralized approach or using multiple systems for a distributed approach. In centralized planning, the encoding of a multi-agent domain D_m consists of a union of multiple set of rules which result in the encoding π_{D_m}, whose answer sets represent a solution for P_m. The set of rules contain the ground rules for variable instantiations, the rules of the planning module, the initial state of the world and the rules for the verification of the goal satisfaction. With these encodings, a plan can be made for multiple agents, whereby the hierarchical organization of the agents can not be exploited in the planning process.
Distributed planning facilitates the exploitation of the independence between agents or groups of agents. With fully collaborative agents, each group of agents is allowed to plan for itself. A scheduler then generates a centralized joint plan, where the output of all individual plans get merged. If there are conflicts between individual plans, because the individual groups plan without any knowledge of the other groups, the scheduler identifies the conflicting plans and requests different plans from the affected groups. The communication between the scheduler and the individual planning systems happens through client-server communication. Resolving the conflicts can take several iterations, which is therefore not a straight forward process. Nonetheless, with this process, the structural relationships between the agents can be exploited.
Non- or partially-collaborative agents are used for applications with less collaboration or rather more competition, which happens when agents withhold private information or when agents are competitive and have conflicting goals with other agents. The planning process takes place through peer-to-peer communications, where agents have to negotiate with each other. In the negotiation process, the agents have to formulate their own goals and need to negotiate the next steps with other agents, which get logged in a protocol. They continue their communication until an agent accepts or refuses the latest proposal. It is important that during the negotiation none of the participating agents change their location or execute any actions. In the event of a successful negotiation, the agents have extended information for further planning.

4 Conclusion

There exist valid main ideas for answer set planning through a definition of planning domains with action languages like $\mathcal{B}$. This is even possible for multi-agent environments, where the planning process can be centralized or distributed. As there does not exist any real-world implementations for multi-agent planning yet, according to the article [1], the non-collaborative planning shows the complex mechanism and the similarities to human planning and negotiation, like in competitive environments or in war. There might be time-complexity problems to be discussed for such complex processes like negations, as delays or non-halting during negations happens in complex real-world problems as well.

References

1. Tran Cao Son, Marcello Balduccini: Answer Set Planning in Single- and Multi-agent Environments. KI 32(2-3): 133-141 (2018)