24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Abstraction for ASP Planning

Zeynep G. Saribatur [1]

Abstract. Humans are capable of abstracting away irrelevant details or distinguishing the common properties when studying problems. This is especially noticeable for planning problems, as humans are able to disregard certain parts of the domain and focus on the key elements important for finding a plan. Recently, the notion of abstraction has been introduced for Answer Set Programming (ASP), a knowledge representation and reasoning paradigm widely used in problem solving, with the potential to understand the key elements of a program that play a role in finding a solution. This paper highlights the potential use of such an abstraction in getting to the essence of planning problems, expressed in ASP, by disregarding irrelevant details and computing abstract plans.

1 INTRODUCTION

Human reasoning and constructing explanations involve the use of abstraction, by reasoning over the models of the world that are built mentally [7]. Among the several interpretations on the meaning of abstraction, one that comes up is the capability of *abstract thinking*. This is achieved by removing irrelevant details and identifying the "essence" of the problem [8]. The notion of *relevance* is especially important in problem solving, since the problem at hand may become too complex to solve if every detail is taken into account. Another view on abstraction is the *generalization* aspect, which is the process of distinguishing the common properties among the objects.

Inspired from this human ability, recently, we introduced the notion of abstraction for ASP,[2] by means of clustering the elements of the domain [11] or omitting certain atoms in the program [10], and automatically constructing an over-approximation. The introduced CEGAR-style [2] abstraction-&-refinement methodology starts with an initial abstraction and refines it repeatedly using hints that are obtained from checking the abstract answer sets, until a concrete solution (or unsatisfiability) is encountered. The methodology is implemented in prototypical tools[3] and evaluated on different benchmarks. Employing such an abstraction showed potential for aiding program analysis as it allows for problem solving over abstract notions, by achieving abstract answer sets that reflect relevant details only or by detecting unsolvability at the abstract level. We recently showed the similarity of the obtained abstractions in unsatisfiable grid-cell problems with the zooming-in ability of humans' explanations [4].

Here, we highlight the potential use of domain abstraction for getting to the essence of the planning problems expressed in ASP, by abstracting over the unnecessary details. This ongoing research opens a wide-range of applications where an understanding is needed, and suggests human-inspired abstractions to tackle the challenges.

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria, email: zeynep@kr.tuwien.ac.at

[2] In collaboration with Thomas Eiter and Peter Schüller.

[3] http://www.kr.tuwien.ac.at/research/systems/abstraction/

2 BACKGROUND

We refer the readers to [1] for details on ASP.

ASP Planning. Planning is represented by adding a *time* variable to the atoms, and introducing action atoms that cause changes over time [9], in different ways. For example, the effects of moving a block on top of another can be expressed with the rules

$$onB(B, B_1, T+1) \leftarrow moveToBlock(B, B_1, T). \quad (1)$$

$$\neg onB(B, B_2, T) \leftarrow onB(B, B_1, T), B_1 \neq B_2. \quad (2)$$

where (1) and (2) show the direct and indirect effects, respectively. Alternatively, all effects can be expressed as direct effects by altering (2) to $\neg onB(B, B_2, T) \leftarrow moveToBlock(B, B_1, T), B_1 \neq B_2$.

Action preconditions can be defined in different forms, e.g., the condition that a block cannot sit on a smaller block can be expressed as a constraint $\bot \leftarrow onB(B, B_1, T), B_1 \leq B.$ or alternatively, the respective action can be forbidden if the precondition is not satisfied:

$$\bot \leftarrow moveToBlock(B, B_1, T), not\ precondmtb(B, B_1, T).$$

$$precondmtb(B, B_1, T) \leftarrow B < B_1, block(B), block(B_1).$$

The law of inertia can also be elegantly described by the rule $onB(B, B_1, T{+}1) \leftarrow onB(B, B_1, T), not\ \neg onB(B, B_1, T{+}1)$.

Domain Abstraction. Abstraction is on *over-approximating* a given program Π, with vocabulary $\mathcal{A}$, by constructing a simpler program Π' with a vocabulary reduced to $\mathcal{A}'$, and ensuring that the results of reasoning on the original program are not lost.

Definition 1 *Π' is an* abstraction *of Π, if there exists a mapping $m : \mathcal{A} \rightarrow \mathcal{A}'$, where $|\mathcal{A}| \geq |\mathcal{A}'|$, such that for any answer set I of Π, $I' = \{m(\alpha) \mid \alpha \in I\}$ is an answer set of Π'.*

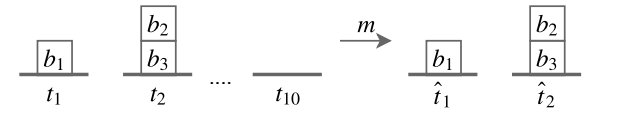
In [11], we introduced *(domain) abstraction mappings* $m : D \rightarrow \widehat{D}$ for non-ground programs Π with domain (Herbrand universe) D and a set $\widehat{D}$ ($|\widehat{D}| \leq |D|$) that divides D into *clusters* $\{d \in D \mid m(d) = \hat{d}\}$ of elements seen as equal. Any such mapping m naturally extends to the Herbrand base $\mathcal{A} = HB_\Pi$ of Π by $m(p(c_1, \ldots, c_n)) = p(m(c_1), \ldots, m(c_n))$. We showed how to construct an abstract program that achieves an abstraction over the abstract atoms.

Planning problems usually contain objects that actions have direct effect on, e.g., the blocks that can be moved, and objects only involved in the decision making, e.g., the table on which a block can be moved. Domain abstraction can be used to abstract over them.

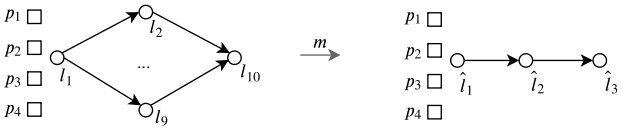
3 ABSTRACTING OVER DETAILS

We first show how an abstraction can be achieved over the details of objects that are only involved in the decision making. For demonstration, we consider two extended well-known planning domains.[4]

[4] Encodings can be found in http://www.kr.tuwien.ac.at/staff/zeynep/pub/ecai20_supp.pdf



(a) Blocksworld with multiple tables



(b) Package delivery with checkpoints

Figure 1. Initial states (concrete $\xrightarrow{m}$ abstract)

Multi-table blocksworld [11]: Figure 1(a) illustrates an example instance, where the blocks need to be piled up on table t_1 as a stack where b_1 is above b_2 and b_2 is above b_3. Although reaching the goal state does not depend on which tables the blocks are moved to before moving to t_1, the planner has to consider all such possible movements. An abstraction over the tables (Figure 1(a)-right) which distinguishes the chosen table t_1 and clusters the remaining tables into $\hat{t}_2$ (in short $\{\{t_1\}/\hat{t}_1, \{t_2, \ldots, t_n\}/\hat{t}_2\}$) makes it possible to compute a plan where the blocks are moved to the abstract table cluster $\hat{t}_2$ before reaching the goal state. This shows the human-interpreted essence on the irrelevancy of the tables other than the goal table.

Package delivery with checkpoints: Figure 1(b) illustrates an example instance, where the packages in location l_1 need to be carried to location l_{10}, by passing through a middle point; through which point the truck passes does not make a difference in reaching the goal state. For the abstract initial state with the abstraction mapping $\{\{l_1\}/\hat{l}_1, \{l_2, \ldots, l_9\}/\hat{l}_2, l_{10}/\hat{l}_3\}$ (Figure 1(b)-right) we get

$$\{load(p_4, \hat{l}_1, 1), load(p_3, \hat{l}_1, 2), load(p_1, \hat{l}_1, 3), load(p_2, \hat{l}_1, 4),$$
$$drive(\hat{l}_2, \hat{l}_1, 5), drive(\hat{l}_1, \hat{l}_3, 6), unload(p_3, \hat{l}_3, 7),$$
$$unload(p_1, \hat{l}_3, 8), unload(p_4, \hat{l}_3, 9), unload(p_2, \hat{l}_3, 10)\}$$

which describes a plan that loads all the packages,moves to the middle cluster location $\hat{l}_1$, moves to the goal location $\hat{l}_3$, and unloads the packages. Furthermore, this is a *faithful abstraction* (without any spurious abstract solutions) when the abstract answer sets are projected to the actions $load, unload, drive$.

The above-mentioned abstractions can be automatically computed by our methodology [11], given an initial abstraction that clusters the domain elements into one. The obtained abstractions are able to disregard details that are not of importance for the essence of the plan feasibility. The faithful abstractions give an understanding of the problem by realizing its focus points.

4 COMPUTING ABSTRACT PLANS

In ASP encodings, abstracting only over the objects causes to compute plans with the original *time* domain and thus have abstract actions such as $load(\hat{p}, l_1, 1)$ which can not be mapped back to original actions that can load all packages in $\hat{p}$ in one step.

Abstracting also over the *time* makes it possible to talk about abstract instances of actions that abstract from the concrete order of action execution. For the Package Delivery (with two locations l_1, l_2 and no checkpoints), consider two abstraction mappings $m_{package} = \{\{p_1, \ldots, p_4\}/\hat{p}\}$ and $m_{time} = \{\{1, \ldots, 4\}/\hat{t}_1, \{5\}/\hat{t}_2, \{6, \ldots, 9\}/\hat{t}_3\}$. The constructed abstract program computes the abstract plan $\hat{\sigma}$: $load(\hat{p}, l_0, \hat{t}_1), drive(l_1, l_2, \hat{t}_2), unload(\hat{p}, l_2, \hat{t}_3)$ which abstracts over the order of package loading/unloading by having abstract actions over time clusters.

The abstraction over time steers the plan computation through the time clusters. Note that even if further packages are added, the plan $\hat{\sigma}$ will still be feasible, given that m_{time} is adjusted to the extended time domain. Automatically finding a suitable abstraction over the objects and the time remains a challenge.

5 DISCUSSION

The potential of the newly introduced notion of ASP abstraction in getting to the essence of planning problems positions our work also within interest to the Planning community, who has been studying over decades ways of getting rid of symmetry and computing generalized plans, such as [13] and, recently, [6, 12], with a focus on PDDL-based representations and classical planning.

By approaching from the ASP perspective, we allow for further expressivity that can represent complex planning problems. Investigating how to incorporate such abstractions in action languages [5], that represent dynamic domains via transition systems, will give us an understanding of these abstractions in the state space, and make it possible to connect our work also to the long-standing abstraction methods in planning, e.g., abstraction by omission [10] and pattern databases [3] are both on projections.

The plans obtained by the abstraction can be used to reach the goal state from any original instance that can be mapped to the abstract initial state. How these plans fit to the area of generalized planning from the ASP perspective remains an interesting open question.

ACKNOWLEDGEMENTS

This work was supported by the FWF project W1255-N23.

REFERENCES

[1] Gerhard Brewka, Thomas Eiter, and Mirosaw Truszczyski, 'Answer set programming at a glance', *Comm. of the ACM*, **54**(12), 92–103, (2011).
[2] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith, 'Counterexample-guided abstraction refinement for symbolic model checking', *JACM*, **50**(5), 752–794, (2003).
[3] Stefan Edelkamp, 'Planning with pattern databases', in *ECP*, (2001).
[4] Thomas Eiter, Zeynep G. Saribatur, and Peter Schüller, 'Abstraction for zooming-in to unsolvability reasons of grid-cell problems', in *XAI@IJCAI*, (2019).
[5] Michael Gelfond and Vladimir Lifschitz, 'Action languages', *Elect. Trans. on AI*, **3**(16), (1998).
[6] Len Illanes and Sheila A. McIlraith, 'Generalized planning via abstraction: Arbitrary numbers of objects.', in *AAAI*, (2019).
[7] Philip N. Johnson-Laird, *Mental models: Towards a cognitive science of language, inference, and consciousness*, number 6, Harvard University Press, 1983.
[8] Jeff Kramer, 'Is abstraction the key to computing?', *Comm. of the ACM*, **50**(4), 36–42, (2007).
[9] Vladimir Lifschitz, 'Answer set programming and plan generation', *AIJ*, **138**(1-2), 39–54, (2002).
[10] Zeynep G. Saribatur and Thomas Eiter, 'Omission-based abstraction for answer set programs', in *KR*, pp. 42–51, (2018).
[11] Zeynep G. Saribatur, Peter Schüller, and Thomas Eiter, 'Abstraction for non-ground answer set programs', in *JELIA*, LNCS, 576–592, (2019).
[12] Silvan Sievers, Gabriele Röger, Martin Wehrle, and Michael Katz, 'Theoretical foundations for structural symmetries of lifted PDDL tasks', in *ICAPS*, volume 29, pp. 446–454, (2019).
[13] Siddharth Srivastava, Neil Immerman, and Shlomo Zilberstein, 'A new representation and associated algorithms for generalized planning', *AIJ*, **175**(2), 615–647, (2011).