Impact of Modeling Languages on the Theory and Practice in Planning Research

Jussi Rintanen*
Aalto University, Department of Information and Computer Science
Helsinki, Finland

Abstract

We propose revisions to the research agenda in Automated Planning. The proposal is based on a review of the role of the Planning Domain Definition Language (PDDL) in the activities of the AI planning community and the impact of PDDL on parts of its research agenda. We specifically show how specific properties of PDDL have impacted research on planning, by putting emphasis on certain research topics and complicating others. We argue that the development of more advanced modeling languages would be – analogously to the impact PDDL has had – a low overhead and smooth route for the ICAPS community shift its research focus to increasingly promising and relevant research topics.

Introduction

Planning in Artificial Intelligence covers a wide range of sequential decision-making problems of intelligent agents, including single-agent, multi-agent, deterministic, nondeterministic, perfect information, imperfect information, and other problem categories. Many of these problems are shared between multiple research communities. For example, papers on planning with uncertainty and incomplete information are regularly published in the conference on Automated Planning and Scheduling (ICAPS), the conference on Uncertainty in Artificial Intelligence (UAI), the general AI conferences, all the leading conferences on Machine Learning such as NIPS, ICML, and ECML, and they are equally investigated in other areas, for example in Intelligent Control in Control Theory.

The research community that is mostly associated with the so-called classical planning problem (perfect information, no uncertainty) and its direct extensions such as temporal planning, is the one that has formed around the ICAPS conference and especially its planning competitions (IPC). This community has adopted the Planning Domain Description Language (PDDL) (Ghallab et al. 1998) as the modeling language used in the competitions and but also as the main framework for developing, expressing and sharing planning problems for evaluating and comparing planning algorithms.

The initial impact of PDDL after its introduction in 1998 was positive, as it allowed direct comparisons between algorithm implementations for classical planning, and led to the development of standard benchmark problem sets. The ICAPS community felt early on the need to extend PDDL to handle more general classes of planning problems, including temporal planning and probabilistic planning. Of these extensions, the most lasting impact has been by the timed/temporal extension, known as PDDL 2.1 (Fox and Long 2003). Both the original PDDL definition and PDDL 2.1 have been used without major changes for over ten years.

In this paper, we evaluate the impact of PDDL on the research practice in the ICAPS community. We will be arguing that there is a mismatch between the concepts being represented in PDDL and the modeling capabilities of PDDL, and that this mismatch is having a negative impact on the whole research area. The main cause of the mismatch is the low abstraction level of PDDL with respect to important core concepts. In the case of the classical (1998) PDDL, the low abstraction level is most clearly visible in the restriction to Boolean (two-valued) state variables which has made it necessary to encode ubiquitous n-valued state variables as sets of Boolean variables. In temporal versions of PDDL the low abstraction level is visible in the lack of an explicit notion of *resources*, which has forced representing (simple forms of) resources as manipulation of state variables.

These issues have had consequences on the type of research done in the ICAPS/IPC community. In some cases where the higher level concepts are preferable, the ICAPS/IPC community has been developing *automated means* to recover the higher level concepts from the low-level PDDL representations. A prime example of this is the research topic of *invariants* or *mutexes*, which is primarily motivated by implicit n-valued state variables represented as Boolean variables. In a number of works this is the sole motivation. In other cases, the lack of important high level concepts has meant that only simple problems have ended up being modeled: the lack of resources as an explicit modeling concept has eliminated the consideration of temporal planning problems with anything but the simplest resource constraints. We will be further arguing that this has led to an unnecessary, unproductive and harmful disconnect from the

*Also affiliated with Griffith University, Brisbane, Australia, and the Helsinki Institute of Information Technology, Finland. This work was funded by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

by far most prominent types of planning&scheduling problems, namely standard pure scheduling problems with no planning component, and that this connection could be one of the most promising routes for planning research to have more substantial real-world relevance.

The properties of the modeling language might seem like a minor concern. However, a vast majority of all papers on classical and temporal planning have their experimental part exclusively test their ideas with the IPC benchmark problems and nothing else. For this reason the properties of the benchmark problems, and the precise way the benchmark problems have been modeled, have had a profound impact on how these (quite substantial) parts of the ICAPS/IPC community are conducting their research, and what they are researching.

As an action for rectifying these problems, we propose the adoption of more powerful modeling languages with high level concepts that better match the actual modeling requirements the community is already facing. Adoption of more advanced languages may seem like a minor trajectory change, but we believe that this could fundamentally change the type of problems being addressed by the ICAPS/IPC community, and would spur new type of research that would have more relevance from the applications perspective. The need for the ICAPS/IPC community to fundamentally change is widely recognized, but concrete proposals about how this could happen have not emerged. Our paper is one attempt to impact one central part of the ICAPS/IPC community's activities, the benchmarking of planning algorithms in connection with the competitions and most of the research on classical and temporal planning.

The Original PDDL Definition from 1998

PDDL (Ghallab et al. 1998) was the first modeling language widely used in the planning community for a wide range of different types of planners. Its conception coincides with the first planning competition in 1998. We will discuss two topics in connection with the classical (non-temporal) part of PDDL, the datatypes, and the measurement of quality of a plan.

Boolean State Variables

Historically, early works on planning focused on Boolean 2-valued state variables, presumably because they are theoretically sufficient for expressing any finite state transition system, and they are also in practice sufficient for representing quite complex problems.

The original PDDL definition includes state variables with Boolean, integer and real values, but a very large proportion of all planners only implement Boolean variables, and numeric variables do not appear in most of the commonly used benchmark problems.

However, many-valued state variables represented as sets of Boolean variables are present in almost all IPC benchmark problems, and lots of planning techniques are based on many-valued representations of these planning problems. Recovering the many-valued state variables from the Boolean representation is the main – yet often unstated – motivation behind invariant algorithms (Rintanen 1998; Edelkamp and Helmert 2000; Gerevini and Schubert 2000; Rintanen 2000; 2008; Helmert 2009; Bernardini and Smith 2011), and was similarly a main motivation for the planning graph construction of Blum and Furst (1995). A vast majority of the examples used in these works are for recovering many-valued variables representing e.g. locations (although some of the algorithms – but by no means all – can also find other types of invariants.)

Many-valued state variables are needed for constructing the so-called *domain transition graphs* (Jonsson and Bäckström 1998) that are behind many methods for computing domain-independent heuristics for explicit state-space search (Helmert 2004; Richter and Westphal 2010; Katz and Domshlak 2010), and, as in the original papers, have often been used also in more theoretical works. Many-valued state variables are already *de facto* being used by most modern planners: a standard preprocessing step is the automated extraction of a many-valued representation from Boolean state variables. This step can be quite expensive[1], and – in the big picture – it is completely unnecessary as the same information would at no cost be available in higher-level models.

The original PDDL definition included unbounded integers, but many of the standard benchmark problems still represent finite-range integer variables as sets of Boolean variables. For many search methods (basically, any other than forward search / explicit state-space search) this poses an unreasonable and completely unnecessary complication: actions with e.g. increments of integer variables with range $[0, n]$ are represented as n actions, each with precondition corresponding to some $i \in \{0, \ldots, n-1\}$ and an effect corresponding to $i + 1$. This split works perfectly fine with forward search, but with e.g. backward search it forces the planner to arbitrarily commit to one particular value of i (when, as is often the case, the current set of subgoals does not determine the value $i + 1$), rather than let the value be determined at later stages of the search. This problem has a dramatic performance penalty.

Similarly to invariants/mutexes for the automated detection of many-valued state variables, integer variables too would seemingly inspire an artificial research problem of automatically recognizing integer-valued state variables from Boolean representations to allow representing them more effectively.

The only reasonable way of resolving this unsatisfactory situation, of course, is to move to modeling languages and models where many-valued and bounded integer variables are represented explicitly. Such a move would be technically minor: both many-valued (enumerated) and bounded integer-valued variables can be directly and easily accommodated in all leading search methods used in planning and state-space search in general. The consequences of such a move would be, firstly, allowing far easier modeling of planning problems where reasoning about numeric values is crit-

[1] The FD framework can spend several minutes extracting (or attempting to extract) a many-valued representation from a Boolean PDDL representation, even with small problem instances with only a couple of hundred state variables.

ical, and, secondly, motivate the development of new search technologies that take advantage of the richer, more structured problem representations.

Quality Measures for Plans

Actions in PDDL can be associated with a cost, and the cost of a plan can be defined as the sum of the costs of the actions in the plan. This is a perfectly acceptable quality measure for problems that are sequential, especially single-agent planning problems in which the agent is able to take one action at a time. The per-action cost could be viewed as a monetary cost or a cost in terms of time.

However, real-world planning problems with this cost criterion are rare. An inspection of the IPC benchmark problems also shows that almost all are actually inherently parallel: multiple agents/actors/vehicles/machines can take multiple actions in parallel and independently of each other. In most of the real-world planning&scheduling problems the time span in which a plan can be executed (*makespan* in scheduling research) is an equally and often a more important factor in assessing the value of a plan.

So even though classical planning in its essence is about asynchronous/untimed *sequences* of actions, even in the standard benchmark sets almost all of the problems are actually instances of temporal planning with the time information stripped off. This raises the question, what is it that the planning community wants to achieve with classical planning? Would it be more productive to openly recognize the fact that almost all of the benchmark problems really are about temporal planning, and draw the obvious conclusions?

PDDL 2.1 for Planning with Time

Fox and Long's (2003) PDDL 2.1 extends the original definition with timed actions, continuous change, and a number of other features. This definition provides fixes to many of the questions that arose with the classical PDDL definition regarding concurrency and quality measures that refer to makespan: actions are associated with a duration and plans are associated with schedules which express at which time points actions are taken, allowing to determine makespans.

Although the language definition immediately faced criticism for most of its design decisions (Bacchus 2003; Boddy 2003; Geffner 2003; McDermott 2003; Smith 2003), the language still has been the de facto temporal planning language of the ICAPS/IPC community because of its use in the planning competitions.[2] We will be discussing some of its main properties in regard to planning research.

Resources

One of the main omissions in PDDL 2.1 – as pointed out by e.g. Geffner (2003) – is the lack of direct support of *resources*. Resources are one of the most central concepts in scheduling. The class of resources that can be most naturally represented in PDDL 2.1 is *unary resources*, with which pairwise overlapping of two actions can be prevented. More general forms of resources include general n*-ary resources*, which allow 1 or more actions to simultaneously use at most n units of the resource (possibly also including non-integer amounts of the resource), and *state resources*, which allow multiple actions to use the same resource if they require the resource to be in the same *state*. PDDL 2.1 has no direct mechanisms for representing these types of resources, although their representation is possible.

What the PDDL 2.1 semantics easily allows, and for which it seems to have been designed, is reduction of unary resource allocation to state-variable manipulation: an action allocates a resource implicitly related to Boolean state variable x by having x in its precondition and immediately falsifying x as its *at start* effect. Now two actions both depending on x cannot both be taken simultaneously, because PDDL 2.1 disallows an action with precondition x if it is made false by *another* action simultaneously. The implicit resource is freed as an *at end* effect of an action. In most of the IPC benchmark problems a resource associated with some object is implicit in the set of state variables representing the state of the object, for example the location. An action moving an object requires that the object is in a certain location, immediately falsifies the corresponding state variable, and makes another state variable, indicating a different location, *true* as its *at end* effect.

Resources are, in fact, a central concept in many types of (real world) planning and scheduling problems. Most forms of scheduling problems arise *only* because of the resource conflicts between their tasks or actions. (Temporal) planning has been traditionally viewed as an extension of scheduling where not only a schedule has to be found but also the tasks – which are fixed in scheduling problems – have to be selected and sequenced. In this sense PDDL 2.1 represents a radical departure from established ideas in the planning and scheduling field, by not directly supporting the explicit expression of the scheduling aspect of planning.

Planning competitions have mostly focused on modifications of the classical benchmark problems, essentially with only action durations added. Other benchmark problems focus on the concept of *required concurrency* (Cushing et al. 2007), which *forces* some actions to be concurrent. This is a property that is almost completely absent from most types of real-world scheduling problems, and hence it is not clear how generally important required concurrency is, even for planning. Focus on required concurrency was originally motivated by the observation that most early temporal planning benchmark problems could be easily solved by classical planners. Of course, the main issue in much of real-world planning&scheduling is that the *makespan* (total use of time) and resource use gets minimized, so the solution to overly simplistic planners that solve temporal problems by ignoring time is to take the minimization of makespan seriously as a quality measure for plans, not developing problems that require concurrency explicitly.

Scalability Implications of Resources in PDDL 2.1

A main issue with implicit low-level representations of resources is the difficulty of reasoning about them. Infer-

[2] However, few application-oriented works have adopted PDDL 2.1 as their modeling language. Research organizations such as NASA have developed their own temporal planning languages, such as ANML (Smith, Frank, and Cushing 2008).

ring that two actions cannot overlap cannot be checked by looking at the two actions' preconditions and effects only. One needs to check whether the precondition of the later action can become true after having been falsified by the first action, which is non-trivial inference involving the entire action set (Bernardini and Smith 2011; Rintanen 2014).

Additionally, the low abstraction level also in this case rules out efficient and the most natural implementations for example with constraint-based search methods. Shin and Davis (2005) presented a reduction from PDDL 2.1 to the SAT modulo Theories (Audemard et al. 2002) framework. A curious property of this reduction for almost all of the IPC benchmark problems is that – due to PDDL 2.1 representation of implicit unary resources – two consecutive actions that need the same (related) resource – for example two actions moving the same vehicle – necessarily have a non-zero gap between them. This is caused by the way the allocation of implicit unary resources is encoded as *at start* effects that falsify the action's own preconditions. For SMT-based representations of temporal planning this gap between actions typically means a factor of two increase in the number of steps required in a plan, in comparison to what is possible with an explicit notion of resources, with which the gap is avoided. This increase in the number of steps means – similarly to Planning as SAT (Kautz and Selman 1996; Rintanen, Heljanko, and Niemelä 2004) – an exponential increase in runtimes, and therefore poor scalability of many natural constraint-based representations.

Modeling for Planning: Requirements

We discuss requirements that advanced modeling languages for planning should fulfill.

Asynchronous Models

For classical (asynchronous) planning problems, a main requirement is the support of a wider range of data types. Bounded integers and enumerated types are standard in modeling languages in related research communities, for example in computer-aided verification. Much more realistic problems could be promoted in the IPC if richer datatypes were available and widely supported. Much of research on classical planning has limited not only to Boolean state variables, but also to the simplest possible action model, with (unconditional) assignments as effects, and conjunctions of positive literals as preconditions.

A more advanced modeling language would support the expression of problems that are structurally very different from the current benchmark problems. This would spur new research in all aspects of classical planning, and would help the ICAPS/IPC community acquire and use system models also from other research communities, and thereby demonstrate relevance also outside the IPC problems sets.

Timed Models

The main issue in PDDL 2.1 we have addressed is resources. A minimum requirement for an advanced language for temporal planning is a full support of the most basic resource types found in scheduling problems, including n-ary resources and state resources. Implications of explicit notions of resources are similar to the ones of advanced datatypes. Additionally, addressing the tight connection between temporal planning and scheduling would immediately give the planning community access to a rich collection of realistic large-scale models, with the possibility to enhance them with aspects of planning.

Actions

The obvious way forward is to adopt a new modeling language, covering both classical and temporal planning, without many of the unused features of current PDDL dialects, and with stronger modeling support for important high-level concepts such as resources and a richer set of datatypes. In the 2014 planning competition close to half of the planners were built on the Fast Downward framework, which is already using many-valued state variables internally. Dozens of planners could be transitioned to a richer modeling language simply by re-implementing and extending parts of FD. Accommodating the kind of extensions discussed in this paper do not appear to involve interesting technical challenges at the implementation level. For example, implementations would still be free to reduce many-valued variables to Booleans if it provides benefits at the implementation level.

Conclusion

We have argued that design decisions of existing modeling languages used by the ICAPS/IPC community have affected its research and other activities

Low abstraction level of a modeling language means that low-level decisions – sometimes with a strong computational impact – are unnecessarily made at the time when the model is designed. We have given examples of this for PDDL: many-valued (enumerated) state variables and finite-range integer state variables modeled as sets of Boolean state variables, as well as representation of resources as low-level manipulation of state variables.

The low abstraction level in PDDL has inspired or necessitated the development of abstraction methods for recovering the relevant higher level concepts from PDDL models. This has been most visible with many-valued state variables which have been recognized with algorithms for finding invariants.

PDDL in its different incarnations is showing its age (over 16 years), it has no formal semantics, and has many limitations in terms of what kind of problems are expressible in it. Replacing it with a modern modeling language with clean semantics and syntax would quickly take the planning community several steps toward more important applications in AI and elsewhere.

References

Audemard, G.; Bertoli, P.; Cimatti, A.; Korniłowicz, A.; and Sebastiani, R. 2002. A SAT based approach for solving formulas over Boolean and linear mathematical propositions. In Voronkov, A., ed., *Automated Deduction - CADE-18, 18th International Conference on Automated Deduc-*

tion, Copenhagen, Denmark, July 27-30, 2002, Proceedings, number 2392 in Lecture Notes in Computer Science, 195–210. Springer-Verlag.

Bacchus, F. 2003. The power of modeling - a response to PDDL2.1. *Journal of Artificial Intelligence Research* 20:125–132.

Bernardini, S., and Smith, D. E. 2011. Automatic synthesis of temporal invariants. In *Proceedings of the Ninth Symposium on Abstraction, Reformulation, and Approximation, SARA 2011, Parador de Cardona, Cardona, Catalonia, Spain, July 17-18, 2011*. AAAI Press.

Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. In Mellish, C. S., ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1636–1642. Morgan Kaufmann Publishers.

Boddy, M. S. 2003. Imperfect match: PDDL 2.1 and real applications. *Journal of Artificial Intelligence Research* 20:133–137.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is temporal planning really temporal. In Veloso, M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1852–1859. AAAI Press / International Joint Conference on Artificial Intelligence.

Edelkamp, S., and Helmert, M. 2000. Exhibiting knowledge in planning problems to minimize state encoding length. In *Recent Advances in AI Planning. 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999. Proceedings*, number 1809 in Lecture Notes in Artificial Intelligence, 135–147. Springer-Verlag.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Geffner, H. 2003. PDDL 2.1: Representation vs. computation. *Journal of Artificial Intelligence Research* 20:139–144.

Gerevini, A., and Schubert, L. K. 2000. Discovering state constraints in DISCOPLAN: some new results. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000) and the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, 761–767. AAAI Press.

Ghallab, M.; Howe, A.; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 161–170. AAAI Press.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173(5):503–535.

Jonsson, P., and Bäckström, C. 1998. State-variable planning under structural restrictions: algorithms and complexity. *Artificial Intelligence* 100(1-2):125–176.

Katz, M., and Domshlak, C. 2010. Implicit abstraction heuristics. *Journal of Artificial Intelligence Research* 39(1):51–126.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, 1194–1201. AAAI Press.

McDermott, D. V. 2003. PDDL2.1 - the art of the possible? commentary on Fox and Long. *Journal of Artificial Intelligence Research* 20:145–148.

Richter, S., and Westphal, M. 2010. The LAMA planner: guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2004. Parallel encodings of classical planning as satisfiability. In *Logics in Artificial Intelligence: 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004. Proceedings*, number 3229 in Lecture Notes in Computer Science, 307–319. Springer-Verlag.

Rintanen, J. 1998. A planning algorithm not based on directional search. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, 617–624. Morgan Kaufmann Publishers.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000) and the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, 806–811. AAAI Press.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *ECAI 2008. Proceedings of the 18th European Conference on Artificial Intelligence*, 568–571. IOS Press.

Rintanen, J. 2014. Constraint-based algorithm for computing temporal invariants. In *Logics in Artificial Intelligence, 14th European Conference, JELIA 2014, September 2014, Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, 665–673. Springer-Verlag.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence* 166(1):194–253.

Smith, D. E.; Frank, J.; and Cushing, W. 2008. The ANML language. ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS).

Smith, D. E. 2003. The case for durative actions: A commentary on PDDL2.1. *Journal of Artificial Intelligence Research* 20:149–154.