Project 3. Research review

Representation: domain representation language

Searching for solutions to a planning problem requires that the problem be represented in a way that enables efficient manipulation and reasoning. **Problem and plan representation** has therefore been an essential component of research in AI planning since its very beginnings.

Being able to encode planning domains and problems using a common formalism has been the focus of an early line of research in AI planning. In 1998, Ghallab et al. introduced the **Problem Domain Description Language (PDDL)**, which followed a number of other efforts (STRIPS, ADL, SIPE-2, Prodigy-4, UMCP, UNPOP, and UCPOP). In their paper *PDDL – The planning domain definition language* Ghallab et al. describe the syntax and the semantics of the proposed language, which was 'intended to **express the "physics" of a domain**, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are', leaving for further extension 'the search-control advice that most planners need' [1]. Concretely, the authors propose as the syntactic notation a form of Extended BNF, which can describe: the structure of a domain, actions, goals, effects, action expansions (an alternative to goal definitions, used in some classical hierarchical planners), axioms ('logical formulas that assert relationships among propositions that hold within a situation'), and safety constraints ('background goals that must be satisfied throughout the planning process') [1]. These elements allow the specification of problems, which are the object of planning systems.

PDDL has been adopted ever since its creation as the standard for the International Planning Competition. PDDL is still in use today, and is now at version 3.1. A number of variants and extensions to PDDL have been defined over time, each enhancing or modifying specific aspects; for example, in 2004 PPDDL (Probabilistic PDDL) introduced, among other things, probabilistic effects ('discrete, general probability distributions over possible effects of an action') [2]. It is thus fair to say that this influential work has succeeded in its stated aim of encouraging problem and algorithm sharing and comparison, thus making an important contribution to the advancement of the field of AI planning.

Sources (to access the source please click on the link):

1. Ghallab et al. 1998 paper: *PDDL – The Planning Domain Definition Language: Version 1.2*
2. Wikipedia article on Planning Domain Definition Language
3. Russel and Norvig's *AIMA* book, 3rd edition 2010

Representation: binary-decision diagrams

Alternative representations for plans have also been experimented with. For example, in the late 1990s and early 2000s some works borrowed **binary-decision diagrams (BDDs)** from logic synthesis and hardware verification, and applied to plan representation these data structures also known as branching programs and originally designed to represent Boolean functions [2, 3]. The applicability of BDDs to plan representation stems from the provability of their property of being a solution to a planning problem.

In a 1998 paper titled *Automatic OBDD-based generation of universal plans in non-deterministic domains*, Cimatti et al. tackle the automatic generation of solutions for **non-deterministic domains**, which are closer to reality than deterministic ones. They describe a planner based on **ordered binary-decision diagrams (OBDDs)** – 'representations of the assignment satisfying and falsifying a given propositional formula', which are used in symbolic model checking and are capable of expressing compactly very large plans, thus enabling the efficient exploration of very large state spaces. To account for non-determinism, the authors make use of **'universal plans'**, which 'extend the (classical) notion of plan (i.e. sequence of actions) to include non-deterministic choice, conditional branching, and iterative plans'. Cimatti et al. find OBDDs to be very well suited to represent symbolically universal plans, because OBDDs enable the compact representation of very large data structures, and thus their efficient manipulation for reasoning about plans and for executing them. The algorithm is **guaranteed to find 'strong solutions'**, i.e. to achieve the goal despite the nondeterministic nature of the environment, if a solution exists; **or, failing that, 'strong cycling solutions'**, i.e. solutions found through iterative trial-and-error strategies 'which are guaranteed to achieve the goal under the assumption that, if there is a possibility for the iteration to terminate, this will not be ignored forever' [1].

The successful application of techniques designed for one domain (formal verification) to a different one (AI planning) is an instance of a more general pattern found in the advancement of scientific research. It goes to show the potential universality across domains of some problem representations, manipulation techniques, and ultimately underlying problems proper. Solving problems is one of the fundamental aims of scientific research, and in that sense any technique that works in one domain is all the more interesting if it may potentially be transferable to a different domain.

Sources (to access the source please click on the link):

1. Cimatti et al. 1998 paper: *Automatic OBDD-based generation of universal plans in non-deterministic domains*
2. Wikipedia article on binary-decision diagrams
3. Russel and Norvig's *AIMA* book, 3rd edition 2010

Partial-order planning

The late 1970s saw the emergence of a planning paradigm that does not impose a specific sequence on the actions to be taken, if order does not matter. This is in contrast to total-order planning – the common approach to planning from the early 1970's –, which sees planning as the search for totally ordered sequences of actions [2, 3].

It was not until the late 1980s that a first clear formal exposition of partial-order planning was made [3]. This was Chapman's 1987 report *Planning for conjunctive goals*, which describes and proves correct TWEAK, a domain-independent planner intended to achieve several goals simultaneously. The algorithm consists of three layers: a plan representation (which is based on 'constraint posting', i.e. the incremental specification of partial descriptions, which means 'properties of the object being searched for do not have to be chosen until a reasoned decision can be made'; planning becomes the process of gradually completing an incomplete plan); a goal-achieving procedure (which is based on the nondeterministic interpretation of a 'modal truth criterion' defining the conditions for the necessary truth of a proposition in a situation); and a top-level control structure (which is 'a search through the space of alternate paths through the goal achievement procedure' using dependency-directed breadth-first search) [1]. In his report Chapman also provides rigorous proofs for the mathematical constructs he uses to prove the correctness of his algorithm.

Other partial-order systems and their descriptions followed in the ensuing years (SNLP, UCPOP, etc.). Partial-order planning became the mainstream paradigm in AI planning till the late 1990s, when it was displaced by methods considered to be faster, although in reality some partial-order planners still are extremely competitive [3].

Sources (to access the source please click on the link):

1. Chapman 1987 report: *Planning for conjunctive goals*
2. Wikipedia article on partial-order planning
3. Russel and Norvig's *AIMA* book, 3rd edition 2010