Research Review - Search and Planning

General Problem Solver

General Problem Solver (GPS) (Newell, Shaw, and Simon (1959)) has been one of the most influential search program in the field of AI. From a set of clauses, the program can model a search space and apply actions based on their preconditions. Most of these concepts are similar to the methods that we developed in our planning project.

At its inception, the program was supposed to open an new era of intelligent machines. Because the approach decouples problem descriptions from solving executions, any problem could potentially be formulated and solved through a common mechanism. However, the program suffered from several limitations (Norvig (1991)). For example, the program was lacking descriptive power and recognized as a NP-Hard problem, limiting its general usage and scalability. Through the history of AI, these issues have become a common denominator of many ideas.

Historically, we can highlight the initial ambition of the project and the inspiration it had on its descendants. The first major planning system, STRIPS (Fikes and Nilsson (1971)), has been modeled based on this paradigm. Moreover, recent cognitive architectures, such as SOAR (Laird (2012)), were also inspired by this approach.

Backtracking and Interleaving

Backtracking and interleaving are essential techniques for programmers and computer scientists (Knuth (1968), Abelson, Sussman, and Sussman (1996)). Coupled with search algorithms, backtracking can keep track of explored states and resume the search at a valid checkpoint. On the other hand, interleaving allows the exploration of different plans, alternating their order to avoid infinite branches. Some concepts behind these techniques, such as continuations and streams (Abelson, Sussman, and Sussman (1996)), have also been influential on their own.

One of the most well known implementation of these concepts is the Prolog language (Roussel (1975)). Compared to imperative languages, Prolog is a logic language that describes problems in term of declarative statements. Instead of a sequence of explicit steps that the program must follow, declarative programs require logical clauses that can be processed by a solving engine. Compared to GPS, first-order logic provides more descriptive power.

We can note that Prolog was inspired by an important family of languages for AI called Planner (Hewitt (1970)). The language has supported the creation of other major planning systems such as WARPLAN (Warren (1974)). Today, the language is still used in state-of-the-art AI systems like IBM Watson (Lally and Fodor (2011)).

Beyond Logicial Programming: Relationnal Programming

The notion of algorithm can be captured by the following slogan: Algorithm = Logic + Control (Kowalski (1979)). Thus, a logicial program has to avoid direct control over its flow to express a problem using only logical statements. This limitation can be a great constraint for programmers accustomed to control the flow of their program. Nethertheless, it opens the way to new types of interesting programming paradigm that can exploit this capacity.

Recently, relationnal languages such as miniKaren in Scheme (W. E. Byrd (2009)) or core.logic in Clojure (("Clojure - Core.logic, A Logic Programming Library for Clojure & ClojureScript," n.d.)), have attracted the curiosity of the developer community. With this languages, plannings and searches can be formulated in a bidirectional manner (Friedman, Byrd, and Kiselyov (2005)). For instance, a relational program could either solve a Sudoku problem or generate all the possible grid. I think this paradigm could lead the community to new research opportunities.

References

Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. 1996. *Structure and Interpretation of Computer Programs*. Justin Kelly.

Byrd, William E. 2009. "Relational Programming in Minikanren: Techniques, Applications, and Implementations." Thèse de doctorat, Citeseer.

"Clojure - Core.logic, A Logic Programming Library for Clojure & ClojureScript." n.d. https://github.com/clojure/core.logic.

Fikes, Richard E, and Nils J Nilsson. 1971. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial Intelligence* 2 (3-4). Elsevier: 189–208.

Friedman, Daniel P.., William E.. Byrd, and Oleg Kiselyov. 2005. *The Reasoned Schemer*. MIT Press.

Hewitt, Carl. 1970. "PLANNER: A Language for Manipulating Models and Proving Theorems in a Robot."

Knuth, D. 1968. "The Art of Computer Programming 1: Fundamental Algorithms 2: Seminumerical Algorithms 3: Sorting and Searching." *MA: Addison-Wesley*, 30.

Kowalski, Robert. 1979. "Algorithm= Logic+ Control." *Communications of the ACM* 22 (7). ACM: 424–36.

Laird, John E. 2012. *The Soar Cognitive Architecture*. MIT Press.

Lally, Adam, and Paul Fodor. 2011. "Natural Language Processing with Prolog in the IBM Watson System." *The Association for Logic Programming (ALP) Newsletter*. Citeseer.

Newell, Allen, John C Shaw, and Herbert A Simon. 1959. "Report on a General Problem Solving Program." In *IFIP Congress*, 256:64. Pittsburgh, PA.

Norvig, Peter. 1991. *Paradigms of Artificial Intelligence Programming: Case Studies in Common LISP*. pub-MORGAN-KAUFMANN:adr: pub-MORGAN-KAUFMANN.

Roussel, Ph. 1975. *PROLOG: Manuel de Reference et d'Utilisation*. Université d'Aix-Marseille II.

Warren, David HD. 1974. *WARPLAN: A System for Generating Plans*. University of Edinburgh. School of Artificial Intelligence. Department of Computational Logic.