# Automatic Analysis of Planning Domains

Derek Long
d.p.long@dur.ac.uk
Department of Computer Science, University of Durham, UK

## 1 Introduction

Efficient domain-independent planning has been a long-standing research goal [17, 4]. The objective is to create a system — a *planner* — that can receive as input a description of a collection of legal actions, a description of an initial state of the world and a characterisation of a set of goal states and for the system to produce a programme of activity made of instances of the actions whose execution would lead from the initial state of a state satisfying the goals. Many variations on this theme have been explored, including the construction of purely sequential plan structures through to more recent successes in building plans with complex temporal structure, exploiting concurrency and managing metric quantities.

Although this programme of research has made remarkable progress in the past decade, many other related areas of automated problem solving, such as constraint satisfaction, have tended to move from exploration of general, all-encompassing problem-solvers to focus on more specialised systems and on the development of tools that support the harnessing of general techniques in specific problem-solving tasks. One important reason for this is that systems that are intended to handle arbitrary problems cannot be easily tailored to exploit deep knowledge of the structures of the problems they are applied to, but must instead rely on general heuristics such as syntactic relaxations of the problems as a guide to search. Although relaxed solutions have proved very powerful in helping domain-independent planners become more successful [11, 1, 3], they do not supply an effective means for tackling many hard problems that arise within planning problems. Many of these problems are variants of problems that have been explored in their own right over many years — routing problems, scheduling, management of resources, workload balancing and so on. One way to address this problem is to combine general planning technology with specialised problem solvers for those parts of a planning problem that are both well understood and unsuited to the generic solving technology of a domain-independent planner. This observation has inspired our research into a variety of automatic domain analysis techniques and exploitation of the use of the analysis in decomposing planning problems for solution on hybrid architectures [12, 8, 5].

## 2 Identifying Problem Structure

Our early work in the area of automatic planning domain analysis [5] focussed on finding domain type structures and invariants. These properties can help a planner to prune useless branches from its search space, both ill-typed action instances and attempts to apply actions that violate basic invariant conditions [7]. This work developed into a more sophisticated analysis capable of identifying structures we call *generic types* [10, 14]. These are not types *within* a planning domain, but meta-types, bringing together families of types *across* domains that share certain behavioural commonality. The common behaviours form the foundation for aplying specialised problem-solvers to find solutions to parts of a planning problem that involve specific groups of generic types (configurations we call *generic clusters*). Examples include transportation clusters, resource allocation clusters, binary switches, construction clusters and processor clusters. Associated with these clusters are various problem-solvers that can be deployed to handle the part of the problem exhibiting this structure, abstracting it from the original planning problem.

As a byproduct of the current analysis we also identify objects that are indistinguishable with respect to their particular potential roles in a planning problem. These objects are symmetric: they can be used

interchangeably in any valid plan. We have exploited this in both static pruning [6] and in dynamic pruning [9]. Dynamic pruning arises when objects lose symmetry as roles are assigned to them in a developing plan, but gain new symmetries as other objects are assigned symmetric roles.

Other strands in our work on domain analysis include exploitation of generic types in instantiation of generic control rules [16], the investigation of the use of generic types as *design patterns* during the process of domain construction [18] and the extension of generic types to include temporal and metric structures.

# 3 Exploitation of Problem Structure

A variety of techniques are available to exploit problem structure once it has been found. The exploitation of invariants and symmetry structure is reasonably well understood, as noted above, but the exploitation of generic type structure is more complex. One approach that we have explored is to use a heuristic search planner, using a state evaluation function to guide its search, and to supplement the cost estimate for achieving the outstanding goals made by the planner with a further estimate of the cost of solving the associated sub-problem (route planning, resource management or whatever). This has proved successful in the cases we have applied it to [13, 2, 8, 15], but it is also clear that it is difficult to extend, with ad hoc arrangements of the integration in each case. We are now investigating a more open integration, in which communication between sub-solvers is made more uniform, allowing a smoother integration of a variety of solvers in flexible deployments. Part of this work involves an analysis of the relationships between sub-problems [14], in order to establish a hierarchy of responsibilities and a chain of communications between the sub-solvers that might be used in solving a single problem.

Although there remain many unresolved challenges in this architecture, we firmly believe that domain-independent planning is not a suitable technology for tackling the entirety of planning problems and that the use of specialised technologies for managing efficient solution of elements of problems is crucial to the scalability of planning technology.

# References


[1] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proc. of 14th National Conference on AI*, pages 714–719. AAAI/MIT Press, 1997.

[2] M. Clark. Construction domains: a generic type solved. In *Proceedings of 20th Workshop of UK Planning and Scheduling Special Interest Group*, 2001.

[3] S. Edelkamp and M. Helmert. On the implementation of mips. In *Proceedings of Workshop on Decision-Theoretic Planning, Artificial Intelligence Planning and Scheduling (AIPS)*, pages 18–25. AAAI-Press, 2000.

[4] R.E. Fikes and N.J. Nilsson. STRIPS: A New Approach to the Application of Theorem-Proving to Problem-Solving. *Artificial Intelligence*, 2(3), 1971.

[5] M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9, 1998.

[6] M. Fox and D. Long. The detection and exploitation of symmetry in planning problems. In *Proceedings of 16th IJCAI*, 1999.

[7] M. Fox and D. Long. Utilizing automatically inferred invariants in graph construction and search. In *Proceedings of the International AI Planning and Scheduling conference, AIPS 2000, Breckenridge, Colorado*, 2000.

[8] M. Fox and D. Long. Hybrid STAN: Identifying and Managing Combinatorial Sub-problems in Planning. In *Proc. IJCAI'01*, 2001.

[9] M. Fox and D. Long. Extending the exploitation of symmetry analysis in planning. In *Proc. of 6th International Conference on AI Planning Systems (AIPS'02)*. AAAI Press, 2002.

[10] Maria Fox and Derek Long. Planning with generic types. In G. Lakemayer and B. Nebel, editors, *Exploring AI in the New Millenium*, pages chapter 4:103–138. Morgan Kaufmann, 2003.

[11] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research*, 14:253–302, 2000.

[12] D. Long and M. Fox. Automatic synthesis and use of generic types in planning. In *International Conference on AI Planning and Scheduling*, 2000.

[13] D. Long and M. Fox. Multi-processor scheduling problems in planning. In *Proc. of International Conference on AI (IC-AI), Las Vegas*, 2001.

[14] D. Long, M. Fox, and M. Hamdi. Reformulation in planning. In S. Koenig and R. Holte, editors, *Proceedings of 5th International Symposium on Abstraction, Reformulation and Approximation (SARA)*, volume 2371 of *Lecture Notes in AI*. Springer-Verlag, 2002.

[15] D. Long, M. Fox, L. Sebastia, and A. Coddington. An examination of resources in planning. In *Proc. of 19th UK Planning and Scheduling Workshop, Milton Keynes*, 2000.

[16] L. Murray. Reuse of control knowledge in planning domains. In L. McCluskey, editor, *Knowledge Engineering Tools and Techniques for AI Planning: AIPS'02 Workshop*, 2002.

[17] A. Newell and H. Simon. GPS, a program that simulates human thought. In E.A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw Hill, NY, 1963.

[18] R. Simpson, L. McCluskey, D. Long, and M. Fox. Generic types as design patterns for planning domain specification. In L. McCluskey, editor, *Knowledge Engineering Tools and Techniques for AI Planning: AIPS'02 Workshop*, 2002.