

# GIPO: An Integrated Graphical Tool to support Knowledge Engineering in AI Planning

R. M. Simpson, T. L. McCluskey, W. Zhao,<sup>†</sup>  
R. S. Aylett, C. Doniat <sup>‡</sup>

Department of Computing Science, University of Huddersfield, Queens Gate, Huddersfield,  
HD1 3DH, UK <sup>†</sup>

Centre for Virtual Environments, University of Salford, Salford, M5 4WT, UK <sup>‡</sup>

**email:** r.m.simpson, t.l.mccluskey, w.zhao @hud.ac.uk

**email:** R.S.Aylett, c.doniat @salford.ac.uk

**Abstract** We describe a Graphical Interface for Planning with Objects called GIPO that has been built to investigate and support the knowledge engineering process in the building of applied AI planning systems. GIPO embodies an object centred approach to planning domain modelling. There are two reasons for providing knowledge engineering support for AI planning: (i) to apply a planning system to a new domain to test the planning system itself (ii) to tackle the end-user problem for the engineer who might be a domain expert but need not necessarily have a specialist knowledge of AI planning. Our research is primarily aimed at developing a method and tools to meet the requirements of the latter case (ii), although the benefits can also be enjoyed by planning experts.

## 1. Introduction

Planform [1] is a UK EPSRC grant funded research project in which we are developing an open platform for the systematic acquisition of planning domain models, and tools to combine these models with planners to create efficient planning applications. Part of the work involves the development of a knowledge acquisition method where knowledge is captured by describing changes that the objects in the domain undergo as the result of the application of operators. The method requires that we structure the domain definition around types of objects, the states that these objects may inhabit, and the possible transitions from state to state that the objects may undergo as a result of the application of planning operators. The content of this definition provides the basis for much of the validation and cross-checking that the tool is capable of performing and allows the domain developer to approach the task of defining operators in a structured and well-supported manner. The additional support provides the possibility of opening up domain definition to modellers who do not need to be as skilled in AI planning technology as has traditionally been the case.

In brief, *GIPO* provides (a) a graphical means of defining a planning domain model (b) a range of validation tools to perform syntactic and semantic checks of emerging domain models (c) dynamic tools to allow the modeller to verify that the domain specification can support known plans within the domain (d) tools to import and export domain definitions to the literal-based PDDL format for typed strips domains, with or without conditional effects (e) an interface that allows for the integration of third party planning algorithms to be run and animated from within the tools environment.

*GIPO* is designed on the assumption that the knowledge engineer will be trying to build descriptions of new domains using a method which imposes a loose sequence on the sub-tasks to be undertaken to develop an initial model. Once an initial rough model has been constructed, development may proceed in a more iterative and experimental manner. A key design goal in building the supporting GUI tool has been to allow the creation of a specification with the tool taking care of the detail of the syntax of the underlying specification. Use of the tool will never result in a syntactically ill-formed specification.

## 2. Domain Acquisition

The process of domain model development and the model's ontology on which this is based is detailed in the literature (see *GIPO* home page [2], which also contains a more detailed version of this paper). Here we sketch the main steps of the knowledge acquisition process, describing how the tool supports this process. We outline two important steps of the knowledge acquisition process - acquiring domain structure and acquiring domain actions.

The process starts with the identification of the kinds of objects that characterise the domain. The method requires that distinct collections of objects, which we call *sorts*, can be organised into a hierarchy. A visual tree editor is used to construct a sort tree and the relations and attributes that characterise the objects of each sort. The key step in the object centred modelling process is to characterise each valid state of objects of the sorts that are subject to change during the planning process by defining their relations and attributes. We refer to sorts subject to such change as *dynamic* whereas the sorts where the objects remain unchanged are *static*. A description of a state of an object we call a *substate definition*. Under classical assumptions each member object of a sort may be in only one such substate at any time, and that during plan execution the object goes through *transitions* which change its state from one such substate to another.

In parallel with the specification of substates the modeller can now assemble planning operators. The operator editor forms the heart of *GIPO*. This editor relies on the notion that operators and methods generally cause objects to change from one substate to another (called *object transitions*). Whereas substate definition captures domain structure, operator definition captures domain behaviour.

For each object changed by the application of an operator there will be a transition defining the set of substates the object may be in prior to the application of the operator and the definition of the precise substate the object will be in as a result of applying the operator. We enable the composition of operators by the domain modeller building a simple graph of the operator by selecting the elements of the transitions from an available list of the predefined substates the object/sort is capable of being in. Consider the remove wheel operator taken from a *tyre change* domain illustrated in figure 1. The rectangles describe states or generalisations on states of objects of identified sorts where the pair of rectangles in the same row represent the *transition* that the referenced object will make as a result of applying the operator, named in the *oval box*. Here there are two objects changed by the operation, the wheel itself and the hub that the wheel was attached to. The hub in the example moves from the state where it is jacked up and free to being jacked up and bare as a result of removing the wheel.

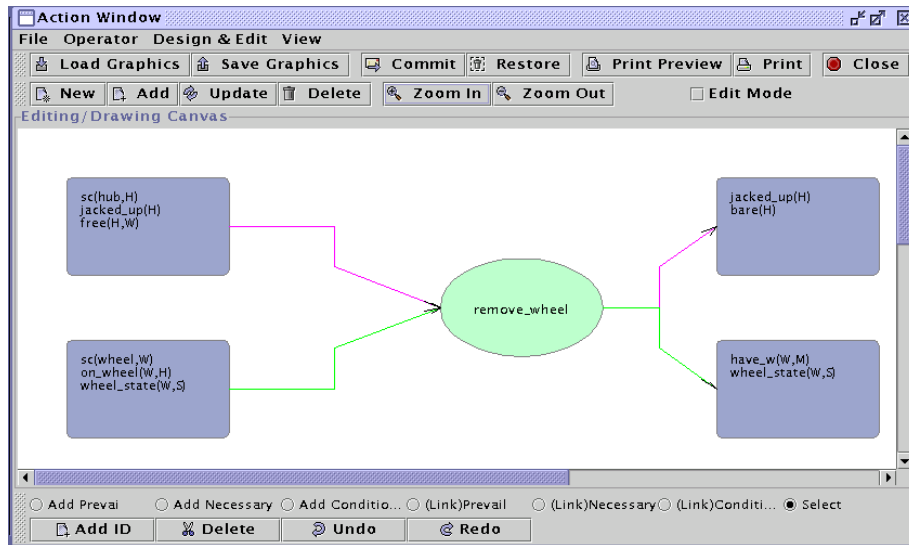


Fig. 1. The Operator Definition Editor

### 3. Domain Analysis

During domain model acquisition numerous *local verification checks* are applied to ensure consistency. Once an initial domain model has been defined as described above, the domain modeller can run *global verification tools* to further check the validity of the specification. Tools which we have developed include goal ordering generators, a random tasks generator, and a “reachability” analysis tool. The latter tool examines substates that are defined for a sort and indexes them against the operators that use them either as consumers or producers. This may reveal to the domain modeller that contrary to expectation some substates cannot be produced and hence could only ever be used in the initial state of an object, or that some cannot be consumed and hence either are only useful in the development of objects of other sorts, or are of the kind of specified only in a goal condition.

In addition to static analysis of the specification the domain modeller can dynamically check a domain against a set of problems either by using the manual *stepper* (shown in figure 2) or by running a selected planning algorithm against defined test problem cases. With the stepper, the engineer chooses actions to apply in the current state to generate the consequent state and proceeds in this manner to verify that the domain and operator definitions do support known plans for given problems within the domain. In the example shown in figure 2, again drawn from the *tyre change* domain, each column of circles represents the state of objects at one time instance. The linked oval is the operator applied to the states with the links tracing the objects changing and participating in the application of the operator. The inset dialog box shows an operator *fetch\_tool* in the process of the user choosing instantiations for the parameters. The panes at the sides show the task being attempted and a list of available operators in the domain.

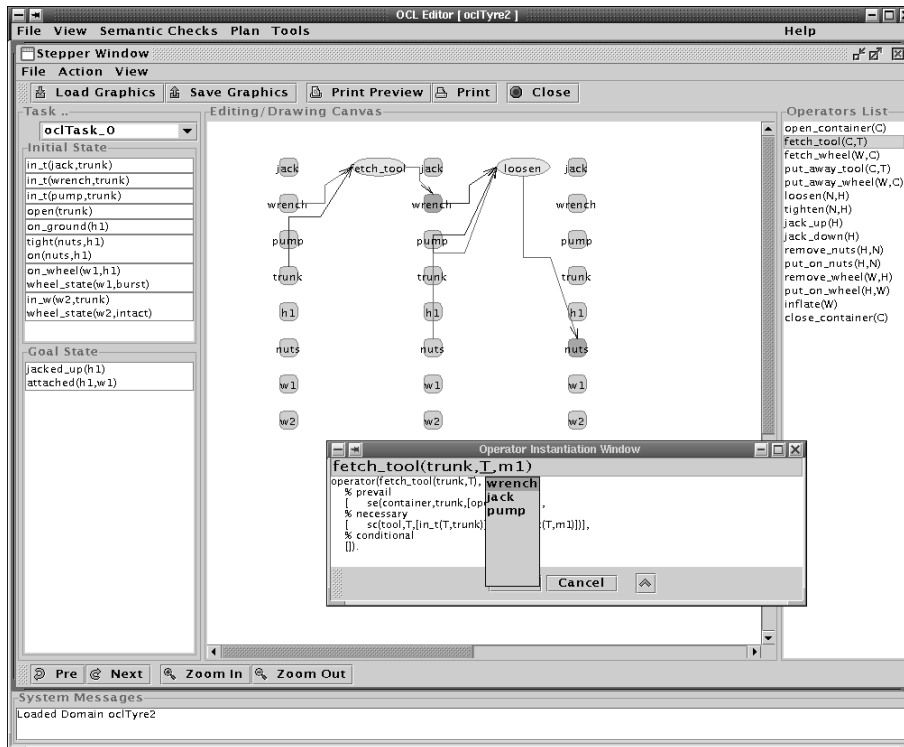


Fig. 2. The Plan Stepper

The *animator* allows integrated planners to run against defined problems and graphically displays the transitions made to objects as the plan unfolds. The animator is structurally very similar to the stepper except that the operator choice is determined by the results output by the planner.

#### 4. Future Work

Although the object centred method lifts domain acquisition to a conceptual level, the details of specifying substate definitions and transitions are still too theoretical for an unskilled user. We aim in the future to incorporate more inferencing mechanisms to aid the unskilled user in this task. We are also developing methods to assist the domain modeller to extract structured knowledge from informal textual descriptions of the domain and to assist the modeller to create new models by providing a library of previous domain models.

#### References

- [1] <http://scom.hud.ac.uk/planform>
- [2] <http://scom.hud.ac.uk/planform/gipo>