GIPO Planning with Objects

GIPO

Graphical Interface for Planning with Objects

GIPO is an experimental research environment for planning domain knowledge engineering. The purpose of the GIPO project is to explore and demonstrate the range and scope of tools required to support the knowledge engineering aspects of planning systems creation and validation. GIPO provides support for both classical pre-condition planning and hierarchical planning. GIPO is object oriented and gives the range of benefits typically associated with OO methods in other fields of software engineering such as highly visual development methods, code re-use and fast reliable development.

Language Levels Supported

- Classical Planning
- Hierarchical [Classical] Planning
- Planning with Durative Actions

Tool Elements

- Domain Definition Editors
- Task Definition Editors
- Static Analysis
- Dynamic Testing
- Manual Task Stepping
- Task Solution Generation
- Integrated Planners
- Third Party Planners
- Export to PDDL

Domain Definition Methods

Multiple methods

- **Follow Object Centric Method.**
  Define - Object Types - Predicates - Object States - Operators
- **Operator Induction**
  Define - Object Types - Predicates - Object States
  Deduce Operators from known plan traces.
  Restricted to Classical Mode and base operators within Hierarchical Mode.
- **Draw "Object Life Histories"**
  Draw stylised state transition diagrams defining the possible state transitions for each type of object and define the connections between the transition diagrams.
  Use re-usable fragments of domains stored in library
  Restricted to Classical and Durative action modes. May be used for base operators within Hierarchical mode.

Static Analysis

- Syntax Checked
  Only legal domain specification will be produced when checks passed.
- Semantics
  State usage analysis reveals states that form dead-ends and states that cannot be generated.
  Transparency analysis in hierarchical models guarantees that methods do achieve their post-conditions when their pre-conditions are met.

Dynamic Testing

- Animators to help inspect plans produced by integrated planning engines.
- Manual steppers to allow developer to check that domain definition does support known plans.

Libraries of Generic Types

- Object Life Histories share common structure that are re-usable.
- Complex structures can be hidden in packages.
- Users can save fragments of their own domains to the library

Durative Actions

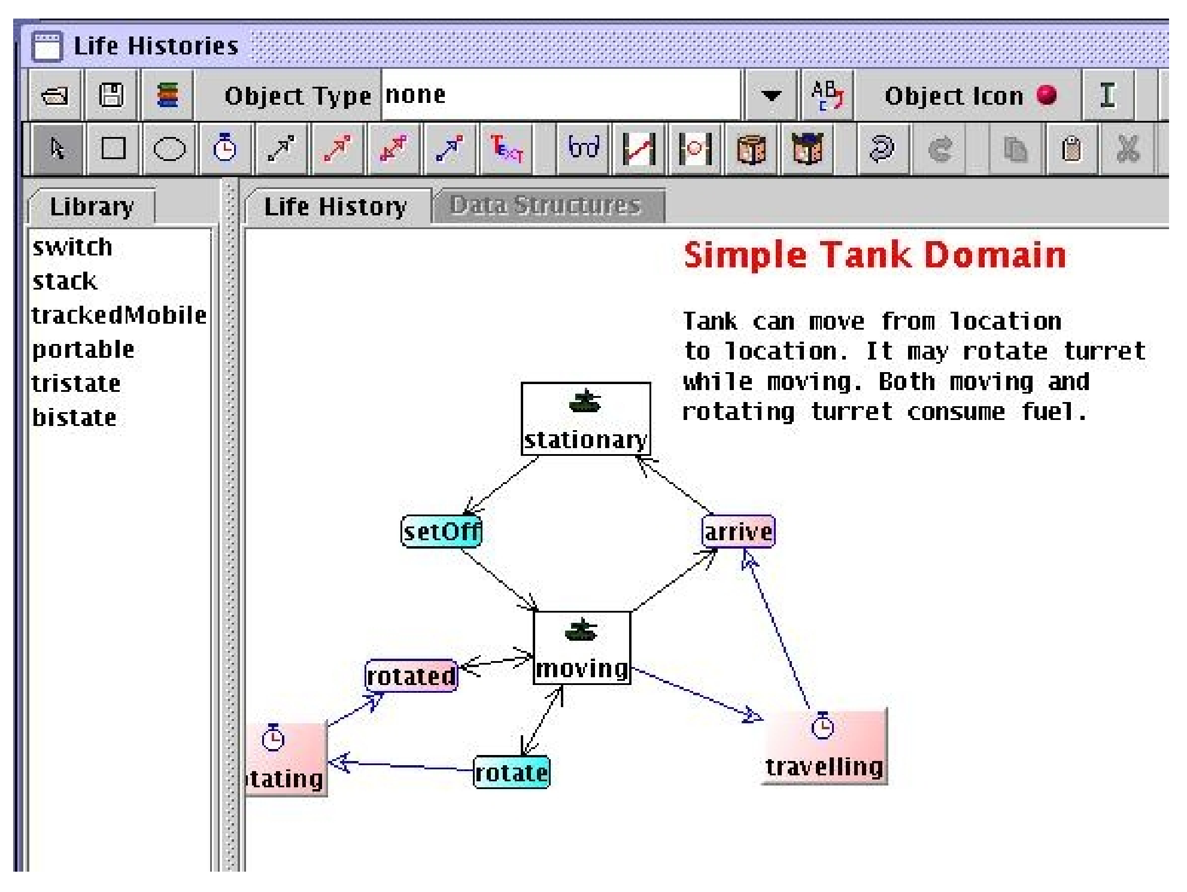
PDDL level 5 equivalent

- Processes.
- Events.
- Numeric Properties.

Supports

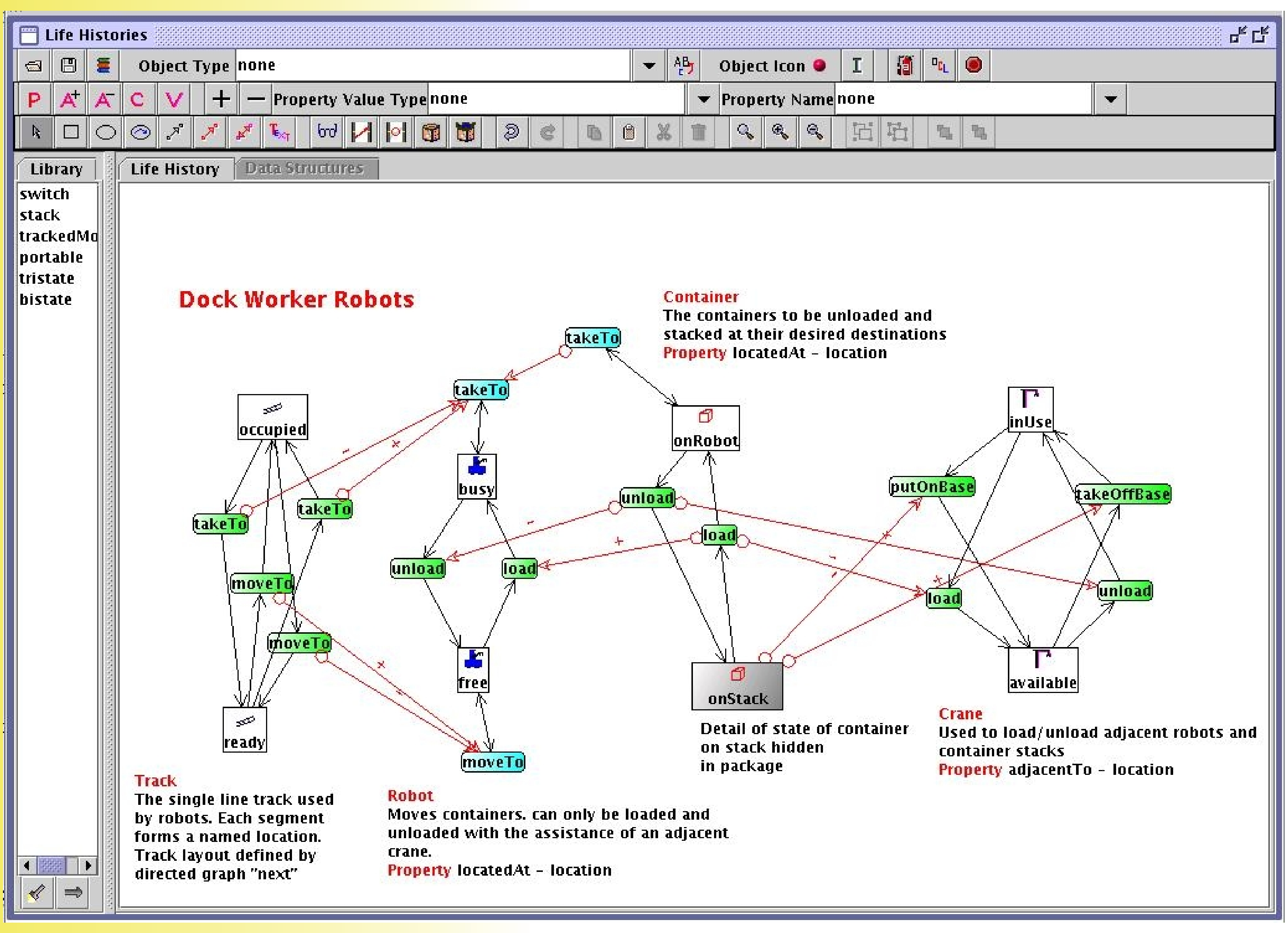
- Domain Design using Life Histories.
- Manual Plan stepping

Plan execution involves changing the state of a subset of objects within the sphere of interest from an initial state to a desired goal state. Domain Definition can be done by describing the possible changes in state that can be brought about to the different types of objects within the sphere of interest. The object metaphor guides the domain designer in structuring the domain definition.



Simple Tank Domain

Tank can move from location to location. It may rotate turret while moving. Both moving and rotating turret consume fuel.



Dock Worker Robots

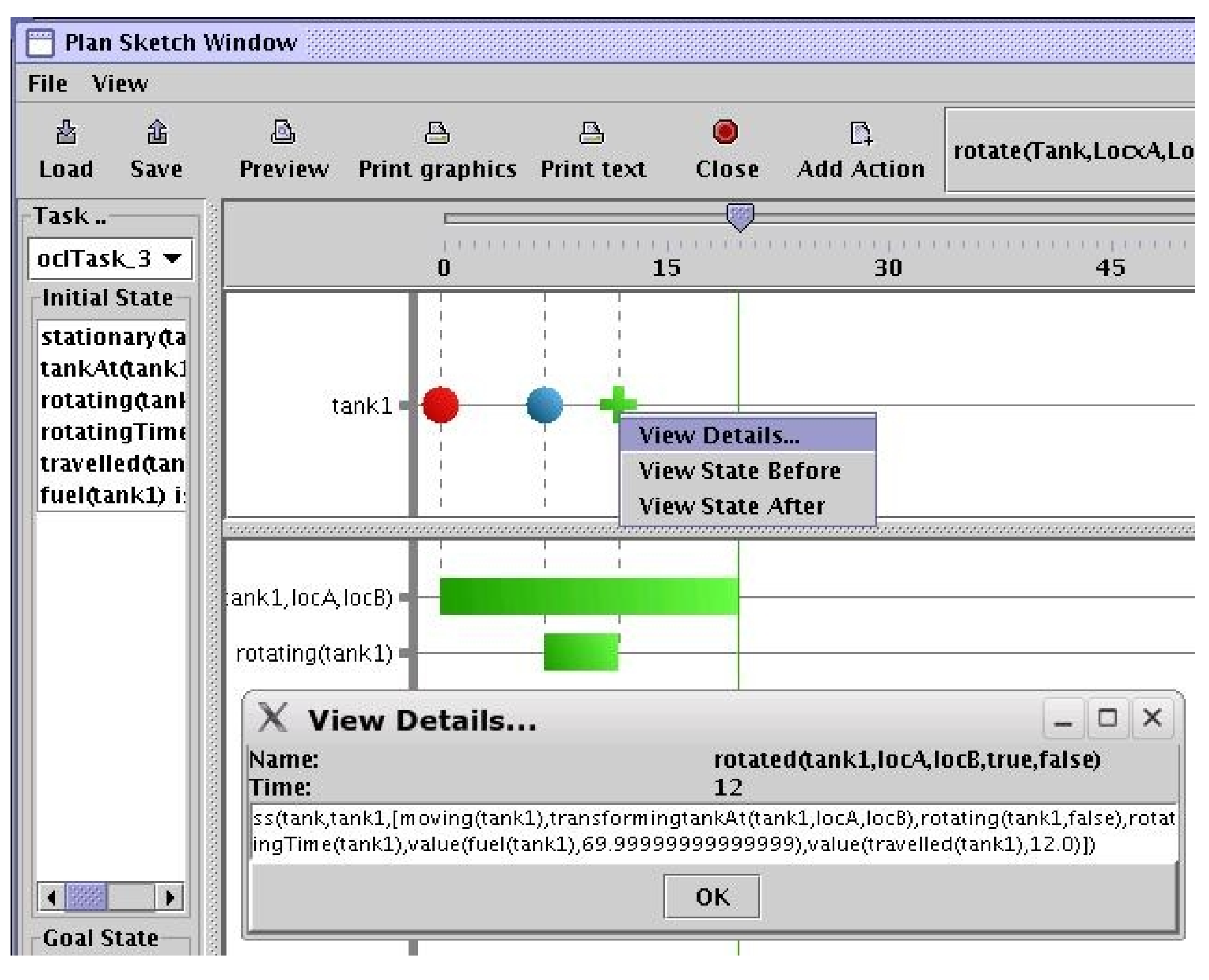
Container
The containers to be unloaded and stacked at their desired destinations
Property locatedAt - location

Track
The single line track used by robots. Each segment forms a named location. Track layout defined by directed graph "next"

Robot
Moves containers. can only be loaded and unloaded with the assistance of an adjacent crane.
Property locatedAt - location

Detail of state of container on stack hidden in package

Crane
Used to load/unload adjacent robots and container stacks
Property adjacentTo - location



Name: rotated(tank1,locA,locB,true,false)
Time: 12
ss(tank,tank1,[moving(tank1),transformingtankAt(tank1,locA,locB),rotating(tank1,false),rotatingTime(tank1),value(fuel(tank1),69.99999999999999),value(travelled(tank1),12.0)])

University of HUDDERSFIELD

GIPO http://scom.hud.ac.uk/planform/gipo