GTPyhop: A Hierarchical Goal+Task Planner Implemented in Python

Dana Nau[1], Yash Bansod[1], Sunandita Patra[1], Mark Roberts[2], Ruoxi Li[1]

[1]University of Maryland, College Park; [2]The US Naval Research Laboratory

GTPyhon extends Pyhop (2013) planner

Combines Hierarchical Task Network (HTN) and Goal Task Network (GTN) planning

- Totally-ordered version of GTN planning without sharing and task insertion
- May have any combination of task decomposition and goal decomposition
- Open source: https://github.com/dananau/GTPyhop

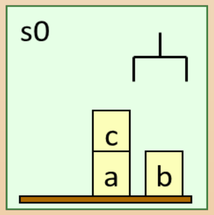
Pyhop (2013)

Simple HTN planner written in Python

- Implements a simple version of the SHOP algorithm
- Despite minimal publicity and no publication: Many citations, several forks
- Used mainly by non-AI researchers as an embedded planning system
- Open source: https://bitbucket.org/dananau/pyhop

Example State and Goal

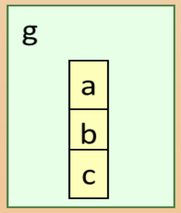
Initial State s0



- Python dictionary notation
- State object holds state-variable bindings

```
s0 = gtpyhop.State('example initial state')
s0.pos = {'a':'table', 'b':'table', 'c':'a'}
s0.clear = {'a':False, 'b':True, 'c':True}
s0.holding = {'hand':False}
```

Goal g



- State-like object gives desired objective
- Goal is state where s.pos[‘a’] == ‘b’
  and s.pos[‘a’] == ‘c’

```
g = gtpyhop.Multigoal('example goal')
g.pos = {'a':'b', 'b':'c'}
```

Task-Method for the task ('take’,x)

- pick up a clear block x, regardless of location
- to-do list depends on location of x

```
def m_take(s,x):
    if s.clear[x] == True:  #precondition
        if s.pos[x] == 'table’:
            return [('pickup', x)]
        else:
            return [('unstack',x,s.pos[x])]
gtpyhop.declare_task_methods('take',m_take)
```

Key:
Args: current state *s*, block *x*, goal *mgoal*
To-Do List
Declaration of actions, methods

Goal-Method for the task ('take’,x)

- pick up a clear block x, regardless of location
- Implements (Gupta & Nau 1992) block stacking algorithm

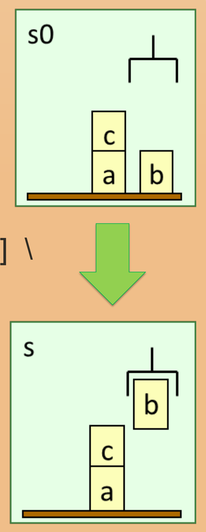
```
def m_moveblocks(s, mgoal):
    for x in all_clear_blocks(s):
        stat = status(x, s, mgoal)
        if stat == ’move-to-block’:
            where = mgoal.pos[x]
            return [(’take’,x), (’put’,x,where), mgoal]
        elif stat == ’move-to-table’:
            return [(’take’,x), (put,x,’table’), mgoal]
    for x in all_clear_blocks(s): # resolve deadlock
        if status(x, s, mgoal) == ’waiting’ and s.pos[x] != ’table’:
            return [(’take’,x), (’put’,x,’table’), mgoal]
    return [ ]    # no blocks need to be moved
gtpyhop.declare_multigoal_methods(m_moveblocks)
```

Example Action

- Blocks-world pickup action

```
def pickup(s,x):
    if s.pos[x] == 'table' and s.clear[x]  \
       and not s.holding['hand']:
        s.pos[x] = 'hand'
        s.clear[x] = False
        s.holding['hand'] = x
    return s

gtpyhop.declare_actions(pickup)
```



Discussion Points

```
method m_transport(r,x,c,y,z)
  Task: transport(c,y,z)
  Pre: loc(r) = x, cargo(r) = nil, loc(c) = y
  Sub: move(r,x,y), take(r,c,y), move(r,y,z), put(r,c,z)
```

- Pseudocode for a conventional HTN method
- Most planners use a specialized language for this
  - Method can have free variables (e.g., r, x), so planner can backtrack over other variable bindings if one fails
  - Planner knows in advance what the subtasks are, can reason about them to decide which method instance to use (e.g., heuristic functions)

```
def m_transport(c,y,z):
  if loc(r) == x and cargo(r) == nil and loc(c) == y:
    (r,x) = find_suitable_robot('transport',c,y,z)
    return [move(r,x,y), take(r,c,y),  move(r,y,z), put(r,c,z)]
```

- GTPyhop method (ordinary python function)
- Advantages:
  - Easy to embed arbitrary Python code
  - Users don’t need to learn a specialized planning language
- Problems:
  - Without free variables, hard to backtrack over other possibilities
  - Can we implement heuristic functions without knowing subtasks in advance?