STRIPS Planning in Infinite Domains
Caelan R. Garrett, Tomás Lozano-Pérez, Leslie P. Kaelbling
MIT CSAIL, {caelan,lpk,tlp}@csail.mit.edu

Task and Motion Planning

- Real world planning applications
- Continuous variables
- Non-linear dynamics
- Task and Motion Planning (TAMP)
- Collision, motion, kinematic, and discrete constraints
- STRIPS limited to finite domains
- We extend STRIPS to incorporate external procedures for modeling these domains
- Also see - “Sample-Based Methods for Factored Task and Motion Planning”
  - Transition system formulation
  - Probabilistic completeness

2 Domain-Independent Algorithms

- Reduce to a sequence of finite domain problems
  - Automatically compile finite domain to PDDL
  - Solve using off-the-shelf planner (FastDownward)

- Incremental algorithm
  - Call finite number of streams and replan
  - Problem - stream calls often expensive

- Focused algorithm
  - Plan using streams and actions to determine which streams could support a solution
  - Abstract objects stand in for real objects
  - Call streams along the plan and repeat

STRIPStream

- STRIPS (PDDL) + Streams
  - Predicates - fluent and static
  - Actions - params, preconditions, effects
- Types: O - BLOCK, P - POSE, G - GRASP, Q - CONF, T - TRAJ
- Streams (samplers)
  - Produce objects and certify static predicates
  - Conditional generator - external procedure

• Conditional Streams - stream with inputs

O, P, G
IsGrasp(O,G)
IK + Motion Planner
Q, T
IsKin(O,P,G,Q,T)

• Test Streams - no object outputs

O, P, T
Collision Checker
CFree(O,P,T)

• Future work - numerical and temporal planning

Task and Motion Planning in Python

- STRIPStream + Factored Transition System Software - https://github.com/caelan/stripstream

Types and Predicates
- Use derived predicates

# Types
CONF, TRAJ, REG = Type(), Type(), Type()
BLOCK, POSE, GRASP = Type(), Type(), Type()

# Fluent predicates
AtConfig = Pred(CONF)
HandEmpty = Pred()
AtPose = Pred(BLOCK, POSE)
Holding = Pred(BLOCK, GRASP)

# Static predicates
IsPose = Pred(BLOCK, POSE)
IsGrasp = Pred(BLOCK, GRASP)
IsKin = Pred(BLOCK, POSE, GRASP, CONF, TRAJ)
IsCollisionFree = Pred(BLOCK, POSE, TRAJ)
IsContained = Pred(REG, BLOCK, POSE)

# Derived predicates
Safe = Pred(BLOCK, TRAJ)
InRegion = Pred(BLOCK, REG)

# Parameters
O, P, G = Param(BLOCK), Param(POSE), Param(GRASP)
Q, G, T = Param(CONF), Param(Pose), Param(TRAJ)
OL, R = Param(BLOCK), Param(REG)

Actions and Axioms
- Safe axiom used to factor collision checking

actions = []
Action(name="pick", parameters=[O, P, G, T],
    condition=And(AtPose(O, P), HandEmpty(),
        IsKin(O, P, G, T), AtConfig(O),
        Forall(DOBox, Or(Equal(DOBox, OBBox), Safe(OBox, T)))),
    effect=And(Holding(O, G),
        Not(HandEmpty()), Not(AtPose(O, P)),
        condition=And(AtPose(O, P),
            Holding(O, G),
            IsKin(O, P, G, T), AtConfig(O),
            Forall(DOBox, Or(Equal(DOBox, OBBox), Safe(OBox, T))))),
    effect=And(AtPose(O, P),
        HandEmpty(),
        Not(Holding(O, G))))

axioms = []
Action(effect=InRegion(O, R), condition=Exists(PI),
    And(AtPose(O, P), IsContained(O, PI)))

TestStream(inputs=[O, P, T],
    condition=IsCollisionFree(0, P, T),
    test=collision_free)