Task and Motion Planning (TAMP)
- Robotic planning with discrete & continuous values
  - Discrete - holding, clean, cooked, etc...
  - Continuous - configs, poses, grasps, trajectories
- Example: movable objects A (in blue) and B (in green)
- Goal: object A in blue region & robot return to start
- State variables (x)
  - Robot config (q)
  - Pose A/B (p)
  - Holding (h)
- Control variables (u)
  - Trajectory (τ)

Factored Transition Systems
- TAMP high-dimensional but factorable
  - Can sample several variables at a time
  - Enables efficient search using AI planners
- Decompose transition relation into clauses
  - Move, Pick A/B, Place A/B, MoveH A/B
- Legal clause transitions given by set of constraints

Low Dimensional Constraints
- Plan skeleton produces constraint network
- Motion, Kin, Stable, Grasp low dimensional
- How do we sample their intersection?
- Example: sample (x, y) satisfying
  1. Sample y from C1
  2. Sample x from from C2 conditioned on y
  3. Reject (x, y) violating C2

Conditional Samplers
- Function from inputs to sampler
  - Placement sampler (Stable)
  - Inverse kinematic solver (Kin)
  - Motion planner (Motion)
- Compose conditional samplers

2 Domain-Independent Algorithms
- Meta-parameter: set of conditional samplers
- Probabilistically complete given sufficient samplers
- Incremental Algorithm \( \approx \) Probabilistic Roadmap (PRM)
  - Repeat:
    1. Compose and sample conditional samplers
    2. Search discretized problem for a plan
- Off-the-shelf AI planning algorithms for discrete search
  - Exploit factoring in their heuristics (e.g. FastDownward)

Focused Algorithm
- Lazy version of Incremental \( \approx \) Lazy PRM
  - Optimistically assume can get samples
  - Repeat:
    1. Create lazy (placeholder) samples
    2. Search sampled problem composed of real & lazy samples for a plan
    3. Sample values for lazy samples on plan

Experiments
- Code + STRIPS version - https://github.com/caelan/stripstream