Combining Planning and Motion Planning

Jaesik Choi* and Eyal Amir
University of Illinois at Urbana-Champaign
*Speaker
Example

• Goal: button (key1) is on (lit)
• Problem: Find a motion+logical plan that achieves Goal

We have two different representations
• A graph built by a motion planner (i.e. PRM) in CSpace
• Actions in PDDL (Planning Domain Description Language)

Two representations share predicates ($P_{key1}(.) = pushed_{key1}$)
Our Representation (Assumption)

• **Assumption: pair of shared predicates** \((P_i, P'_i)\) **are given.**
  - \(P_i: \text{CSpace} \rightarrow \{0,1\}\) and \(P'_i: \text{State Space of PDDL} \rightarrow \{0,1\}\)

In the example
  - \(P_2 = P_{\text{key1}}\) and \(P'_2 = \text{Pushed}_\text{key1}\)

\[\begin{align*}
P_1, P_2, \ldots, P_{M-1}, P_M \\
P'_1, P'_2, \ldots, P'_{M-1}, P'_M \\
S_1, S_2, S_3, \ldots, S_{K-1}, S_K
\end{align*}\]
Problem Definition

• **Given**
  – A CSpace* for a robot and objects
  – PDDL (Planning Domain Description Language)
  – Initial configuration (c_{init}) in CSpace
  – Initial/Goal conditions (q_{init}/q_{goal}) in PDDL
  – Predicates† map between PDDL and CSpace

• **Goal**
  – Finding a path from the c_{init} ∧ q_{init} to q_{goal}
    • The path is collision free in CSpace
    • Each action in the path changes states according to PDDL description.

*CSpace is the set of all the possible configurations.
† A predicate is a function which maps a configuration to a discrete value
Talk Outline

• Example
• Problem Definition
• Our Method & Details
• Computational Complexity
• Experimental Results
• Related Works
• Conclusion
• If have time: Why CPMP?
Algorithm Overview - I

1. Build a graph from a resolution-refutation motion planner
2. Project the graph into the states of shared predicates
3. Verify and Build primitive actions with the projection

\[ P_i(c) = 1, \ P_j(c) = 0 \]

**act\_key1**

Pre: \( \neg P_i \land \neg \text{On\_key1} \)

Eff: \( P_i \land \text{On\_key1} \)

\[ P_i(c) = 0, \ P_j(c) = 0 \]

**act\_bothkeys**

Pre: \( \neg P_i \land \neg P_j \land \ldots \)

Eff: \( P_i \land P_j \land \text{On\_key1} \land \ldots \)
Verify Actions

- Verify a PDDL action
  - Given a PDDL action (precondition: $\varphi_{\text{pre}}$, effect: $\varphi_{\text{eff}}$)
  - Take propositions only in predicate space ($\varphi'_{\text{pre}}$, $\varphi'_{\text{eff}}$)

- If there is no edge which realizes the projected PDDL action
  - Delete the action (i.e. $\text{act}_{\text{bothkeys}}$)

- Otherwise, build new PDDL action for a group of edges (which realizes the action)
  - $(\neg P'_i \land \neg P'_j, P'_i \land \neg P'_j)$ realizes (or is a model) of $(\neg P'_i, P'_i)$
  - For new actions, collects the set of edges which realize it

(precondition, effect) represents precondition and effect of an action

- $\text{act}_{\text{key1}}$
  - Pre: $\neg P'_i \land \neg \text{On}_{\text{key1}}$
  - Eff: $P'_i \land \text{On}_{\text{key1}}$

- $\text{act}'_{\text{key1}}$
  - Pre: $\neg P'_i \land \neg P'_j \land \neg \text{On}_{\text{key1}}$
  - Eff: $P'_i \land \neg P'_j \land \text{On}_{\text{key1}}$

- $\text{act}_{\text{bothkeys}}$
  - Pre: $\neg P'_i \land \neg P'_j \land \ldots$
  - Eff: $P'_i \land P'_j \land \text{pushed}_{\text{key1}} \land \ldots$
Build Actions

- Build PDDL actions
  - For each edge which is not realized* by a PDDL action
  - Build an action for the edge
    - For example

```
act_{unkey1}
Pre: P'_i \land \neg P'_j
Eff: \neg P'_i \land \neg P'_j
```

- Motion Planning complements PDDL description
  - Those actions complement the incomplete action descriptions
  - Thus, they reduce manual encodings

* realized can be replaced by subsumed in logic
Algorithm Overview - II

4. Find a plan (sequence of actions) with the modified PDDL
   • Using AI planner (i.e. Factored Planning, GraphPlan)

5. Find a path which passes through the PDDL actions in order

\[ \text{act}_{\text{key1}} \]
\[ \text{Pre: } \neg P_i \land \neg P_j \land \neg \text{pushed}_{\text{key1}} \]
\[ \text{Eff: } P_i \land \neg P_j \land \text{pushed}_{\text{key1}} \]

\[ \text{act}_{\text{unkey1}} \]
\[ \text{Pre: } P_i \land \neg P_j \]
\[ \text{Eff: } \neg P_i \land \neg P_j \]
## Computational Complexity

<table>
<thead>
<tr>
<th>C₁, C₂, C₃,...,Cₙ₋₁, Cₙ</th>
<th>P₁, P₂, ..., Pₘ₋₁, Pₘ</th>
<th>S₁, S₂, S₃,...,Sₖ₋₁, Sₖ</th>
</tr>
</thead>
</table>

- N (dimensions of CSpace), M (Shared Predicates), and K (Propositions in State Space of PDDL)

- **Computational Complexity of our algorithm (bottom up)**
  - \(O(\exp(N)+\exp(M+K))\)
    - Building a connectivity graph in CSpace: \(O(\exp(N))\)
    - Verifying actions with the Graph: \(O(\exp(M))\)
    - Find a Plan in PDDL: \(O(\exp(M+K))\)

- **Other Hybrid Planning with only projection**
  - \(O(\exp(N+M+K))\): Find a heuristic on a projected space, first
    - Planning in an abstract discrete space \(O(\exp(M+K))\)
      - It provide a heuristic, quickly
    - Planning in a huge CSpace: \(O(\exp(N+M+K))\)
      - The heuristic should be verified in the original space
Experimental Setting in Simulation

- **Initial**: no button is pushed
- **Goal**: Call A, B and C
- **Constraints encoded in actions**
  - **A_called**: Push Call after Push A
  - **B_called**: Push Call after Push B
  - **C_called**: Push Call after Push C
  - **Unlock**: Push unlock after On key1 and On key2
    - Unlock causes ¬On_{key1} and ¬On_{key2}
  - **Lock**: Push Call
    - Call causes Lock
Result Video
Related Works

• **Hybrid Representations** are used with Motion Planning:
  – **Hierarchical decomposition**: Alami, Laumond, Simeon, 1997; Alami, Chatila, Fleury, Ghallab, Ingrand, 1998; … (Manually determined)

• **Object Manipulations** are addressed in many works:
  – Manipulation Planning: Dacre-Write, Laumond, Alami, 1992; Cortes, 2003 (Assembly Planning); Stilman, Kuffner, 2005; T. Bretl, 2006 (Legged Locomotion); Stilman, 2007 (Heuristic search in higher level)
  – Learning Manipulation Task: Pardowitz, Zollner, Dilmann, 2006 …

• **Key difference is the direction of information**
  – Previous: Abstract Space (Discrete) → CSpace (Continuous)
  – This: CSpace (Continuous) → PDDL (Discrete)
Summary & Conclusions

• Key idea:
  – Verify and build PDDL actions from a motion plan

• More details in the paper
  – Tree decomposition of PDDL reduces computations
  – Decomposition with the set of reachable objects in CSpace

• Future Works:
  – Action formalism for situation calculus
  – Extension to more expressible logics (i.e. FOL, LTL)
Acknowledgment

• Thanks
  – Steven Lavalle, Tim Bretl, Seth Hutchinson, Benjamin Tovar, Han-Ul Yoon, and many others who gave valuable comments

• Funding Agencies
  – National Science Foundation under Grant No. 05-46663
  – UIUC/NCSA Adaptive Environmental Sensing and Information Systems (AESIS) initiative
Thank you!

Questions?
Motivation – Motion Planning Only

- **Strength of Motion planning algorithms**
  - Planning with **kinematic constraints** in CSpace

- **Weakness of Motion planning algorithms**
  - Planning with **non-kinematic constraints**
  - i.e. **To unlock the phone, key1 and key2 should be pushed**

- If we plan with non-kinematics constraints,
  - **Redundant**: In many case, the representation is not compact. A state ‘key1 is pushed’ and a state of the 1st joint appear in a configuration.
  - **Computationally inefficient**: Building a CSpace given a (non-kinematic) constraints needs computations which are exponentially proportional to the DOFs.

**Assumption: an encoded CSpace is given.**
Motivation – AI Planning Only

- **Strength of AI planning and description (PDDL)**
  - **Planning with logical constraints in state space**

- **Weakness of AI planning**
  - Executing (or actuating) real-world robots
    - i.e. How to push a button (key1)?

- If we plan with kinematics constraints,
  - **Incomplete (require details)**: In many cases, an action (i.e. push a button) is not straightforward to incorporate kinematic constraints into logic states. Moreover, there is no guarantee that there is a path for the action.
  - **Ambiguous**: there are many ways to interpret an action.

Assumption: executable actions are given.