# Fully Observable Nondeterministic HTN Planning – Formalisation and Complexity Results

School of Computing

The Australian National University

Dillon Chen, Pascal Bercher

Australian National University

# What is HTN Planning?

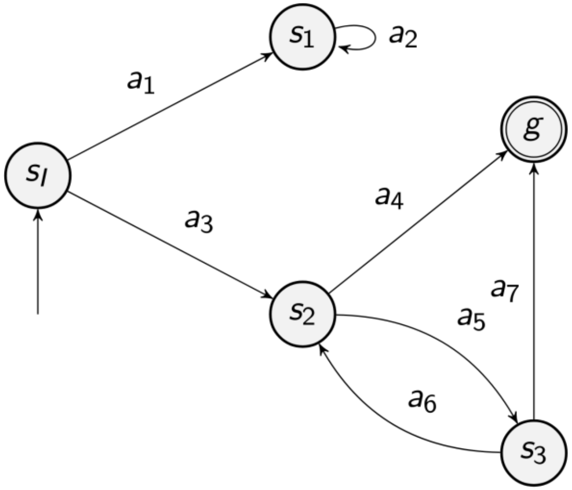
- Classical Planning:
  - Problem domain is a bunch of states and actions
  - Aim is to **reach a goal state**
  - Solution is a **sequence of actions**

# What is HTN Planning?

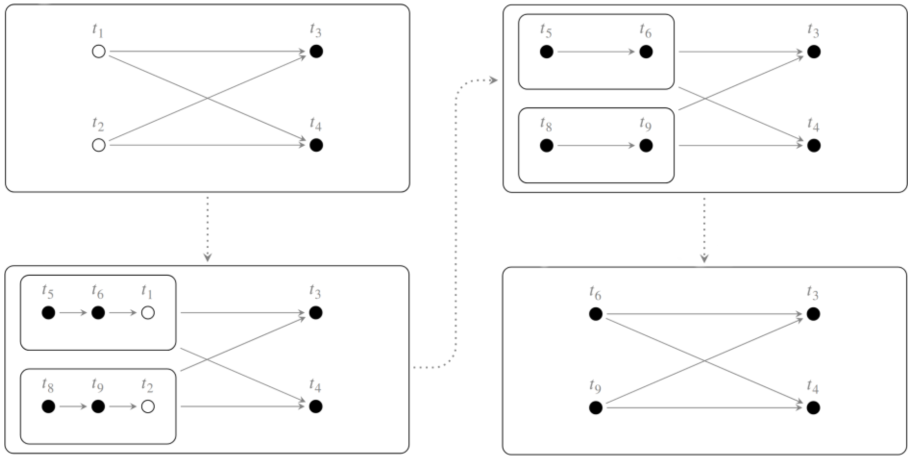
- Classical Planning:
  - Problem domain is a bunch of states and actions
  - Aim is to **reach a goal state**
  - Solution is a **sequence of actions**
- HTN Planning:
  - Problem domain is a bunch of states and (primitive and compound) tasks
  - Aim is to **execute a specified set of tasks**
  - Solution is
    1. a **refinement** of compound tasks followed by
    2. an **executable linearisation** of primitive tasks

# What is HTN Planning?

- Classical Planning




- HTN Planning



# Why HTN Planning?

- Expressive – complexity ranges from tractable to undecidable
  - Has a nice canonical compilation from classical planning
- Easy to encode domain dependent knowledge
- Levels of abstraction helpful for communicating with users

# Uncertainty in Planning

- Standard Planning:
  - Actions may have several effects, may be probabilistic
  - Solutions no longer a sequence but **policy of actions**
    - Notions of weak/strong
    - Probability of success

# Uncertainty in Planning

- Standard Planning:
  - Actions may have several effects, may be probabilistic
  - Solutions no longer a sequence but **policy of actions**
    - Notions of weak/strong
    - Probability of success
- HTN Planning:
  - Actions may have several effects, may be probabilistic
  - Solutions ???
    - Complications arise from compound tasks
    - Several possible formalisations available

# Possible Formalisations

*(Deterministic) Solution is*

1. *a **refinement** of compound tasks followed by*
2. *an **executable linearisation** of primitive tasks*

- Linearisation dependent solutions:

1. a refinement of compound tasks followed by
2. an `executable linearisation' for nondeterministic primitive tasks

# Possible Formalisations

*(Deterministic) Solution is*

1. *a **refinement** of compound tasks followed by*
2. *an **executable linearisation** of primitive tasks*

- Linearisation dependent solutions:

1. a refinement of compound tasks followed by
2. an `executable linearisation’ for nondeterministic primitive tasks

- Outcome dependent solutions:

1. a refinement of compound tasks followed by
2. a policy for nondeterministic primitive tasks

# Possible Formalisations

*(Deterministic) Solution is*

1. *a **refinement** of compound tasks followed by*
2. *an **executable linearisation** of primitive tasks*

- Linearisation dependent solutions:

1. a refinement of compound tasks followed by
2. an `executable linearisation' for nondeterministic primitive tasks

- Outcome dependent solutions:

1. a refinement of compound tasks followed by
2. a policy for nondeterministic primitive tasks

- Flexible solutions (future study):

❖ a policy involving both primitive and compound tasks

❖ Execution is difficult

# Complexity

- Classical Planning is PSPACE-complete
- Nondeterministic Planning is EXPTIME-complete

- General HTN planning is undecidable
- However, there exist HTN problem subclasses
  - range of complexities: tractable (P), NP, …
  - Same case for FOND HTN planning

# Complexity

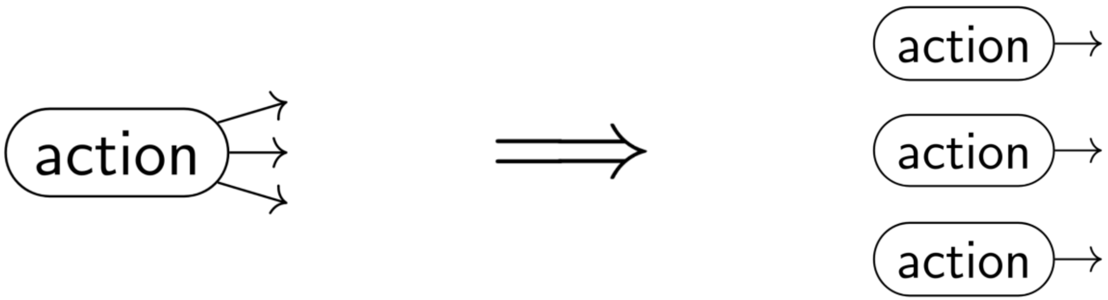
Key results:

- (almost all) weak FOND HTN problems can be compiled into deterministic problems
- totally ordered FOND HTN problems can be compiled into deterministic problems
- partially ordered FOND HTN problems made at least one class harder

# Complexity

- (almost all) weak FOND HTN problems can be compiled into deterministic problems

one action with $n$ effects $\rightarrow$ $n$ actions with one effect each



# Complexity

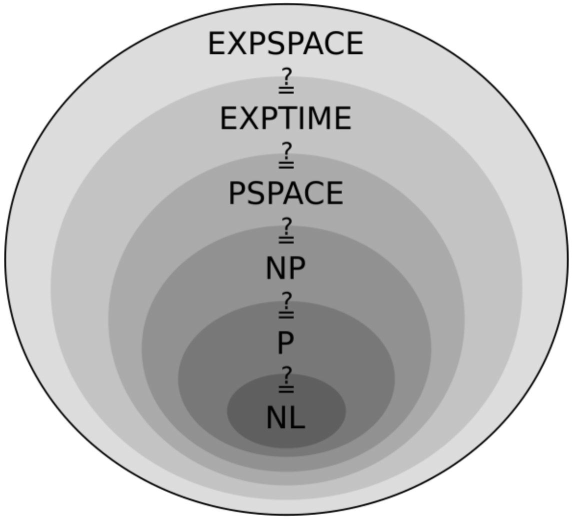
- totally ordered FOND HTN problems can be compiled into deterministic problems

$$\text{one action with } n \text{ effects} \rightarrow \text{one action with one effect}$$

$$\begin{aligned} \mathit{eff}_1 &= \{\text{add}_1, \text{del}_1\} \\ \mathit{eff}_2 &= \{\text{add}_2, \text{del}_2\} \\ \mathit{eff}_3 &= \{\text{add}_3, \text{del}_3\} \end{aligned} \Longrightarrow \mathit{eff} = \left\{\bigwedge \text{add}_i, \bigvee \text{del}_i\right\}$$

# Complexity

- partially ordered FOND HTN problems made at least one class harder



# Results

*See our paper for proofs!*

<table>
<tr><th rowspan="3">Hierarchy</th><th rowspan="3">Order</th><th rowspan="3">FOD</th><th colspan="3">FOND</th></tr>
<tr><th>Weak</th><th colspan="2">Strong</th></tr>
<tr><th></th><th>linearisation-dependent</th><th>outcome-dependent</th></tr>
<tr><td rowspan="2">primitive</td><td>total</td><td>$P^*$</td><td>NP [4.1]</td><td colspan="2">$P^*$ [4.8]</td></tr>
<tr><td>partial</td><td>$NP^{\alpha}$</td><td>NP [4.2]</td><td>NP [4.7]</td><td>PSPACE [5.1]</td></tr>
<tr><td rowspan="2">no recursion (acyclic)</td><td>total</td><td>$PSPACE^{\beta}$</td><td>PSPACE [4.4]</td><td colspan="2">PSPACE [4.8]</td></tr>
<tr><td>partial</td><td>$NEXPTIME^{\beta}$</td><td>NEXPTIME [4.4]</td><td>NEXPTIME [4.7]</td><td>$EXPSPACE^*$ [5.2]</td></tr>
<tr><td rowspan="2">regular</td><td>total</td><td>$PSPACE^{\alpha}$</td><td>PSPACE [4.5]</td><td colspan="2">PSPACE [4.8]</td></tr>
<tr><td>partial</td><td>$PSPACE^{\alpha}$</td><td>PSPACE [4.5]</td><td>PSPACE [4.7]</td><td>$EXPSPACE^*$ [5.3]</td></tr>
<tr><td rowspan="2">tail-recursion</td><td>total</td><td>$PSPACE^{\beta}$</td><td>PSPACE [4.4]</td><td colspan="2">PSPACE [4.8]</td></tr>
<tr><td>partial</td><td>$EXPSPACE^{\alpha,\beta}$</td><td>EXPSPACE [4.4]</td><td>EXPSPACE [4.7]</td><td>semidecidable$^*$ [3.1]</td></tr>
<tr><td rowspan="2">arbitrary recursion</td><td>total</td><td>$EXPTIME^{\beta}$</td><td>EXPTIME [4.4]</td><td colspan="2">EXPTIME [4.8]</td></tr>
<tr><td>partial</td><td>semi- & undecidable$^{\alpha,\gamma}$</td><td>semi- & undecidable [3.1]</td><td>semi- & undecidable [3.1]</td><td>semi- & undecidable [3.1]</td></tr>
</table>