Lecture 14

CS 6260 Automated Planning and Learning

HTN Planning 3

Department of Computer Science and Engineering

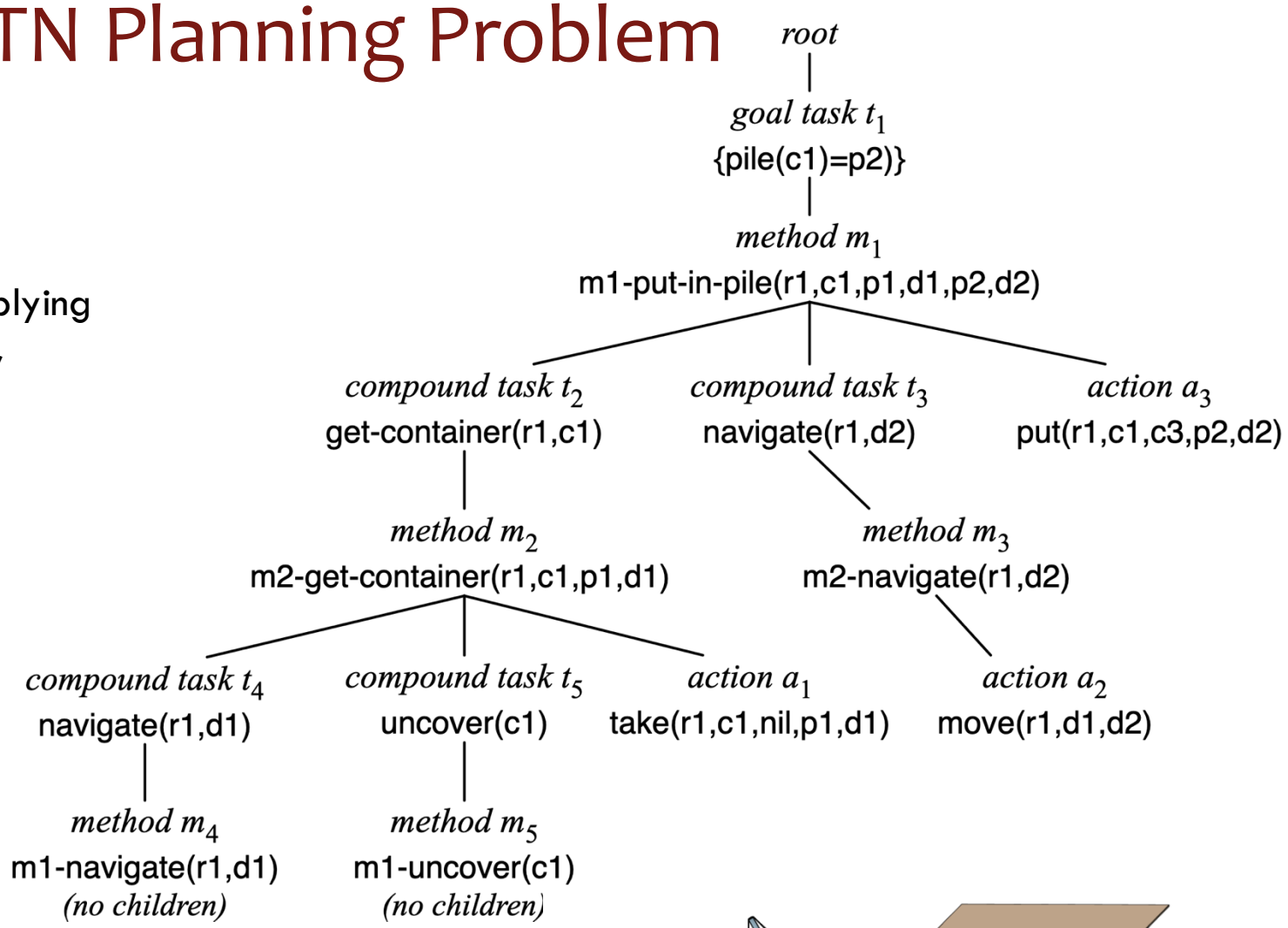
Indian Institute of Technology Madras

Some material based on lecture slides for Automated Planning and Acting by Nau/Roberts.

Recap: HTN Planning Solution

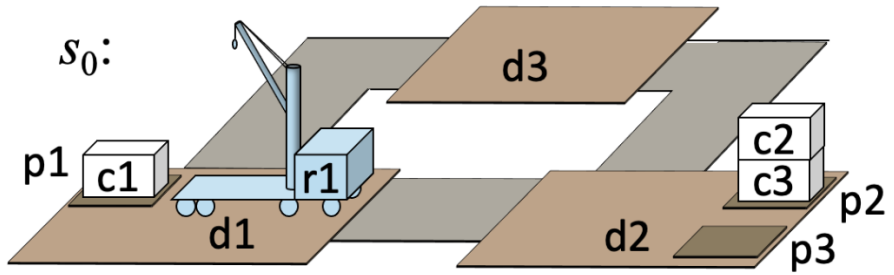
TOHTN Planning Problem

- *Planning problem:* $P = (\Sigma, s_0, T)$
- *Solution*
  - any executable plan produced by applying method instances to nonprimitive tasks, actions to primitive tasks



$P = (\Sigma, s_0, \langle\{\text{pile(c1)=p2}\}\rangle)$

$\pi = \langle$take(r1,c1,c2,p1,d1), move(r1,d1,d2), put(r1,c1,c3,p2,d2)$\rangle$



TOHTN Planning Domain (continued)

- Compound task navigate(*r*,*d*):
  - Get robot *r* from current location to *d*
  - May require several move actions
- These methods are just for illustration, I don't recommend using them
  - Use a route planner instead
- Method for the case where loc(*r*) = *d*

m1-navigate(r, d)
task: navigate(r, d)
pre: loc$(r) = d$
sub: — // *no subtasks*

- Method for cases where loc(*r*) is adjacent to *d*

m2-navigate(r, d', d)
task: navigate(r, d)
pre: adjacent(d', d), loc$(r) = d'$
sub: move(r, d', d)

- Method for cases where loc(*r*) isn't adjacent to *d*

m3-navigate(r, d', d)
task: navigate(r, d)
pre: loc$(r) \neq d$, ¬adjacent(loc(r), d), adjacent(loc(r), d')
sub: move$(r,$ loc$(r), d')$ // *primitive task*
navigate(r, d) // *compound task*

- Methods in $\mathcal{M}$:
  - m1-put-in-pile, m2-put-in-pile, m1-get-container, m2-get-container, m1-uncover, m2-uncover, m1-navigate, m2-navigate, m3-navigate
- Tasks in Σ:
  - Primitive: all instances of
    - move(*r*,*l*,*m*), take(*r*,*c*,*l*), put(*r*,*c*,*l*)
  - Compound: all instances of
    - put-in-pile(*c*,*p*), uncover(*c*), and navigate(*r*,*d*)
  - Goal tasks: all instances of goal(cargo(*r*) = *c*)

TOHTN Planning Problem

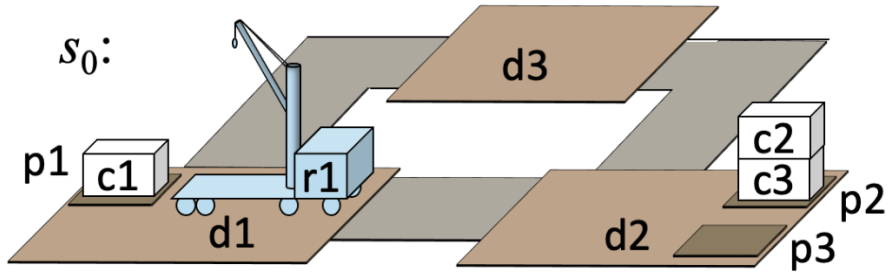
Planning problem: $P = (\Sigma, s_0, T)$

Solution

- If T is empty, then the empty plan $\langle\rangle$ is a solution for P.
- Else, let t be the first task of T, so that $T = t \cdot T'$
  where T' is a (possibly empty) sequence of tasks.
- If t is an action in Applicable($s0$), then for every solution π for the problem (Σ, $\gamma(s0, t)$, T'), the plan $t \cdot \pi$ is a solution for P.
- If t is a compound task or goal task and $m \in$ Methods($s0$, t, M), then every solution for the problem (Σ, $s0$, subtasks(m) $\cdot T'$) is also a solution for P.
- If t is a goal task and $a \in$ Actions($s0$, t), then for every solution π for the problem (Σ, $\gamma(s0, a)$, T'), the plan $a \cdot \pi$ is a solution for P.
- If t is a goal task and $s0 \models t$, then every solution for the problem (Σ, $s0$, T') is also a solution for P.

$P = (\Sigma, s_0, \langle\{\text{pile(c1)=p2}\}\rangle)$

$\pi = \langle$take(r1,c1,c2,p1,d1), move(r1,d1,d2), put(r1,c1,c3,p2,d2)$\rangle$



TOHTN Planning Problem

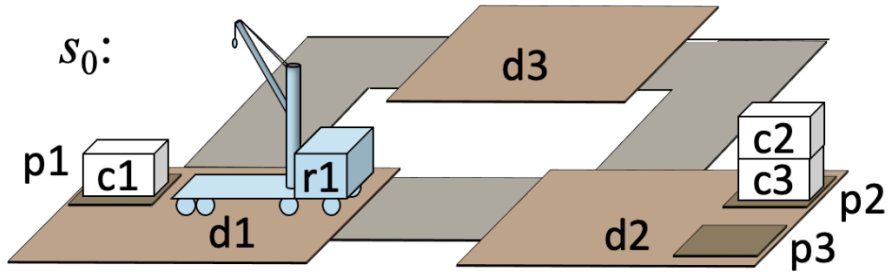
Planning problem: $P = (\Sigma, s_0, T)$

Solution

- If T is empty, then the empty plan $\langle\rangle$ is a solution for P.
- Else, let t be the first task of T, so that $T = t \cdot T'$
  where T' is a (possibly empty) sequence of tasks.
- If t is an action in Applicable($s0$), then for every solution π for the problem (Σ, $\gamma(s0, t)$, T'), the plan $t \cdot \pi$ is a solution for P.
- If t is a compound task or goal task and $m \in$ Methods($s0$, t, M), then every solution for the problem (Σ, $s0$, subtasks(m) $\cdot T'$) is also a solution for P.
- If t is a goal task and $a \in$ Actions($s0$, t), then for every solution π for the problem (Σ, $\gamma(s0, a)$, T'), the plan $a \cdot \pi$ is a solution for P.
- If t is a goal task and $s0 \models t$, then every solution for the problem (Σ, $s0$, T') is also a solution for P.

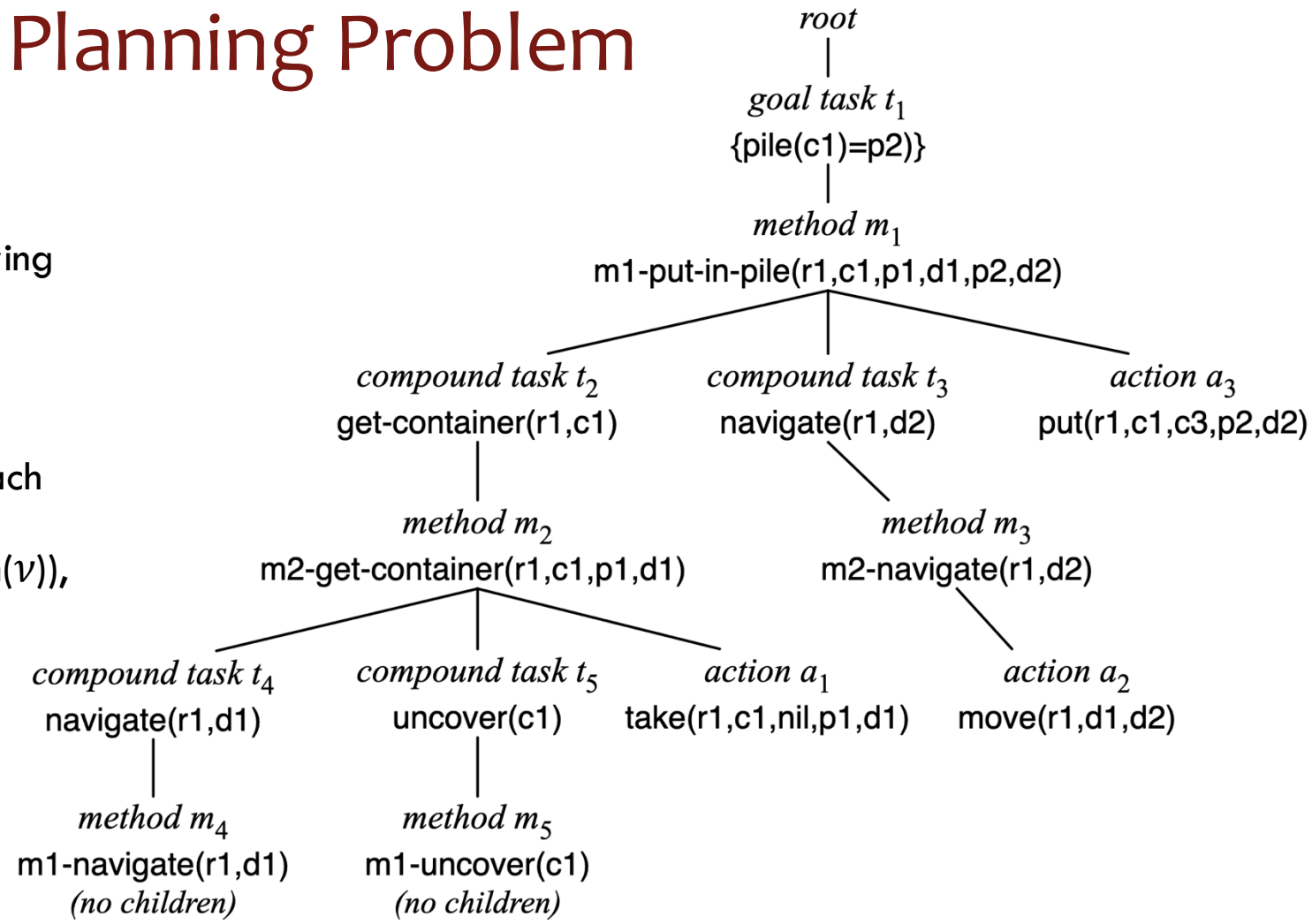
$P = (\Sigma, s_0, \langle\{\text{pile(c1)=p2}\}\rangle)$

$\pi = \langle$take(r1,c1,c2,p1,d1), move(r1,d1,d2), put(r1,c1,c3,p2,d2)$\rangle$



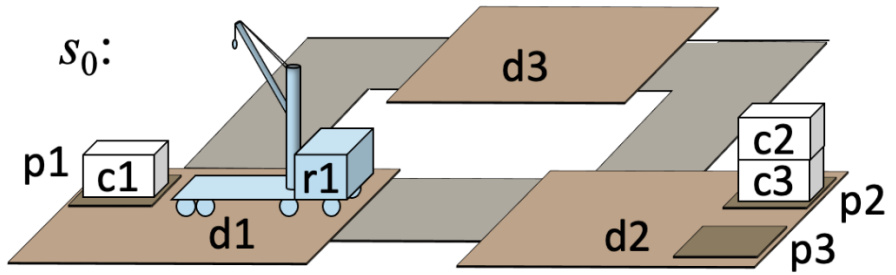
TOHTN Planning Problem

- *Planning problem:* $P = (\Sigma, s_0, T)$
- *Solution* any executable plan produced by applying method instances to nonprimitive tasks, actions to primitive tasks
- *Refinement tree*
  - A refinement tree for π is a tree in which each node is a tuple
    ν = (label(ν), content(ν), parent(ν), Children(ν)),
  - Nodes:
    root, compound tasks, goal tasks,
    method instances, actions
  - Edges:
    root → top-level tasks,
    task → method instance,
    method instance → subtasks



$P = (\Sigma, s_0, \langle\{\text{pile(c1)=p2}\}\rangle)$

$\pi = \langle$take(r1,c1,c2,p1,d1), move(r1,d1,d2), put(r1,c1,c3,p2,d2)$\rangle$

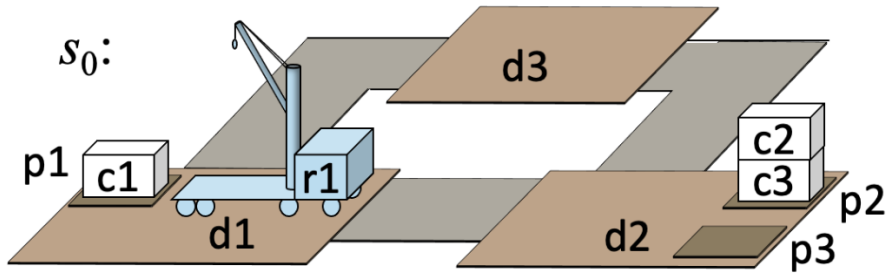


TOHTN Planning Problem

- *Planning problem:* $P = (\Sigma, s_0, T)$
- *Solution* any executable plan produced by applying method instances to nonprimitive tasks, actions to primitive tasks
- *Refinement tree*
  - A refinement tree for π is a tree in which each node is a tuple $v = (\text{label}(v), \text{content}(v), \text{parent}(v), \text{Children}(v))$,
  - Nodes:
    root, compound tasks, goal tasks,
    method instances, actions
  - Edges:
    root → top-level tasks,
    task → method instance,
    method instance → subtasks

$P = (\Sigma, s_0, \langle\{\text{pile(c1)=p2}\}\rangle)$

$\pi = \langle\text{take(r1,c1,c2,p1,d1), move(r1,d1,d2), put(r1,c1,c3,p2,d2)}\rangle$



Planning Algorithm

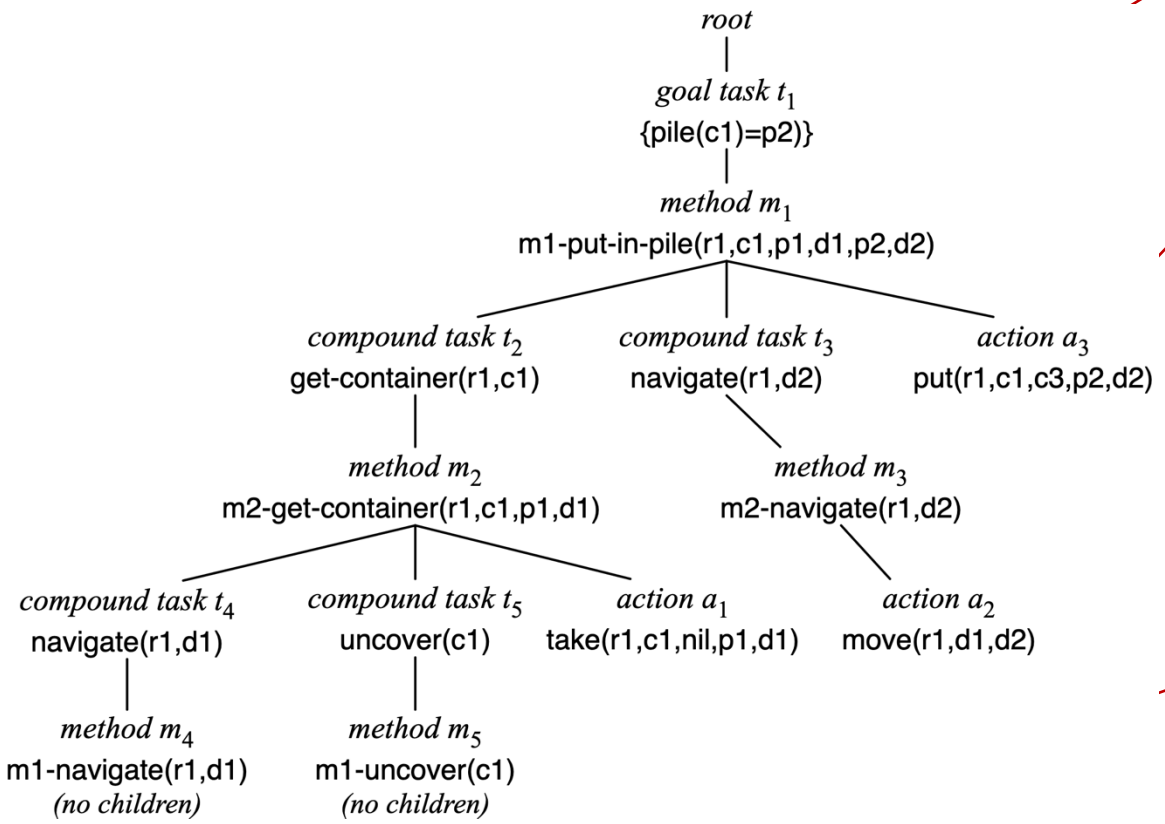
- Definitions

$$Achievers(s,t) = \{a \in Applicable(s) \mid \gamma(s,a) \models t\}$$

$$Ground(\mathcal{M}) = \{\text{all ground instances of methods in } \mathcal{M}\}$$

$$Refiners(s,t,\mathcal{M}) = \{m \in Ground(\mathcal{M}) \mid t \text{ is refinable by } m \text{ in } s\}$$

- Three cases: primitive, compound, goal task
  - Primitive task: apply action



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

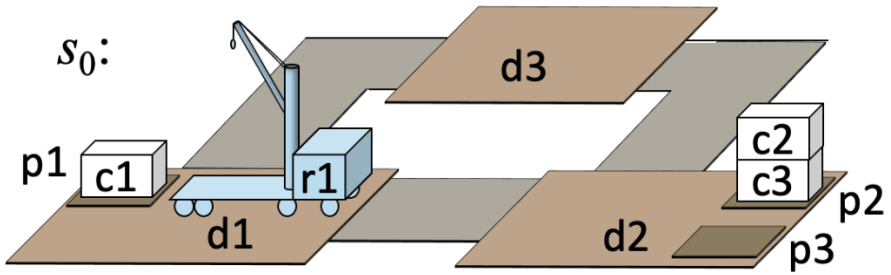
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow Refiners(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow Refiners(s,t,\mathcal{M}) \cup Achievers(s,t)$
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

T= ⟨{pile(c1) = p2}⟩



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

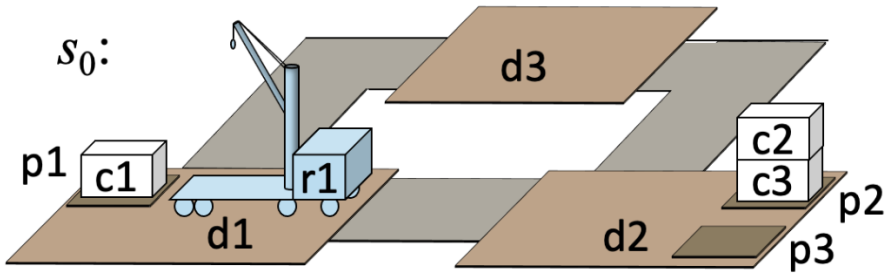
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7   **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

T= ⟨{pile(c1) = p2}⟩



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

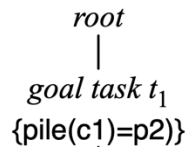
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow Refiners(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow Refiners(s, t, \mathcal{M}) \cup Achievers(s, t)$
7   **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

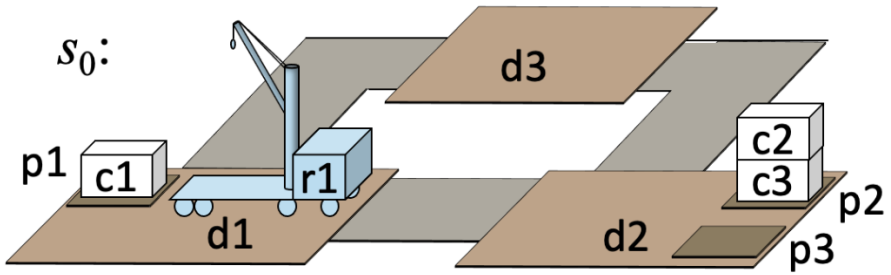
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩





TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

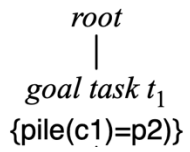
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

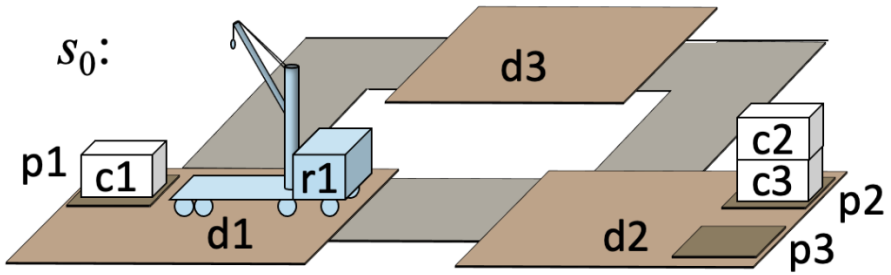
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}



M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, {pile(c1) = p2})



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s$, sub$(m) \cdot T'$)

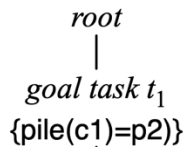
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

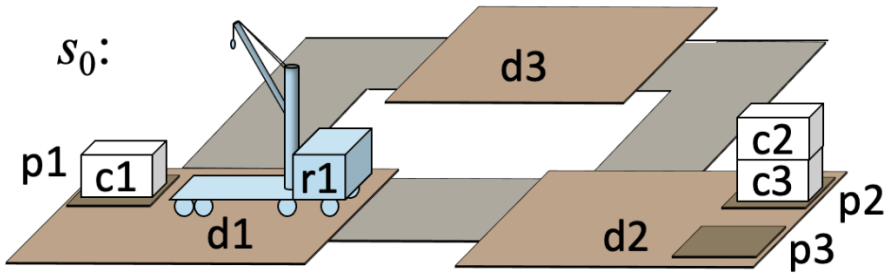
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}



M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, {pile(c1) = p2})



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

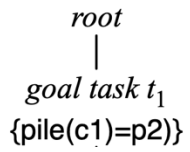
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

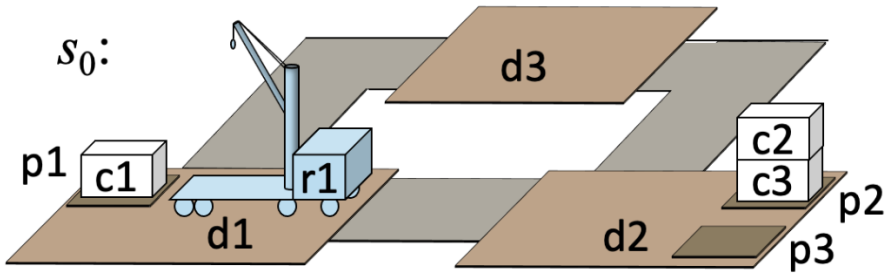
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}



M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, {pile(c1) = p2})



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

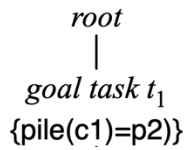
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
  case t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

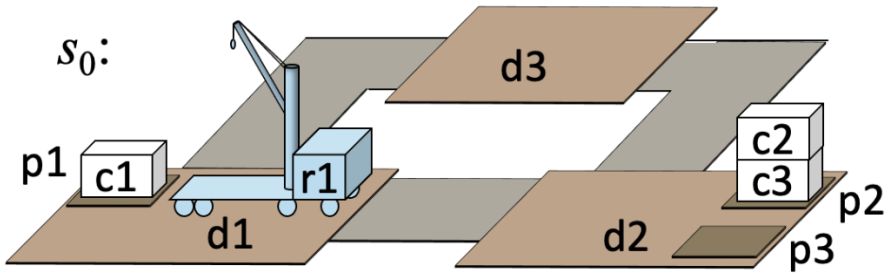
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}



M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, {pile(c1) = p2})



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

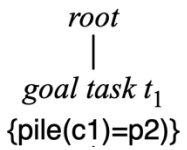
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

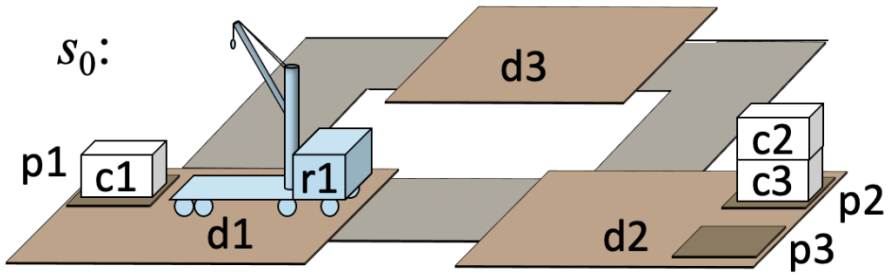
t= {pile(c1) = p2}



M = Refiners(s, {pile(c1) = p2}, $\mathcal{M}$) ∪
Achievers(s, {pile(c1) = p2})

$$Achievers(s,t) = \{a \in Applicable(s) \mid \gamma(s,a) \models t\}$$

$$Refiners(s,t,\mathcal{M}) = \{m \in Ground(\mathcal{M}) \mid t \text{ is refinable by } m \text{ in } s\}$$



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

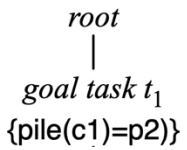
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow Refiners(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow Refiners(s,t,\mathcal{M}) \cup Achievers(s,t)$
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

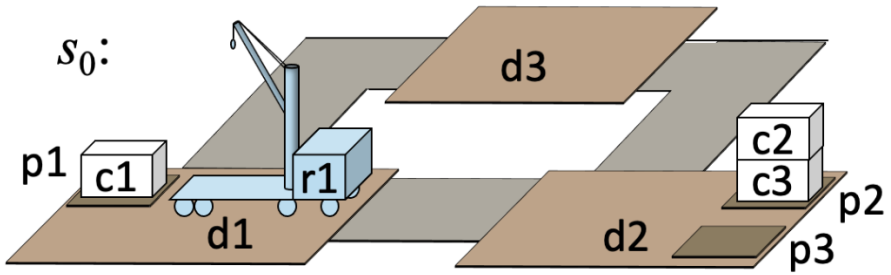
t= {pile(c1) = p2}



M = {m1-put-in-pile(r1,c1,p1,p2,d2)} ∪
∅

$$Achievers(s,t) = \{a \in Applicable(s) \mid \gamma(s,a) \models t\}$$

$$Refiners(s,t,\mathcal{M}) = \{m \in Ground(\mathcal{M}) \mid t \text{ is refinable by } m \text{ in } s\}$$



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

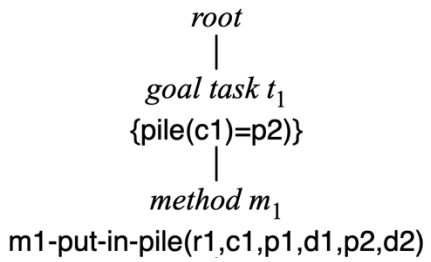
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow Refiners(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow Refiners(s,t,\mathcal{M}) \cup Achievers(s,t)$
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

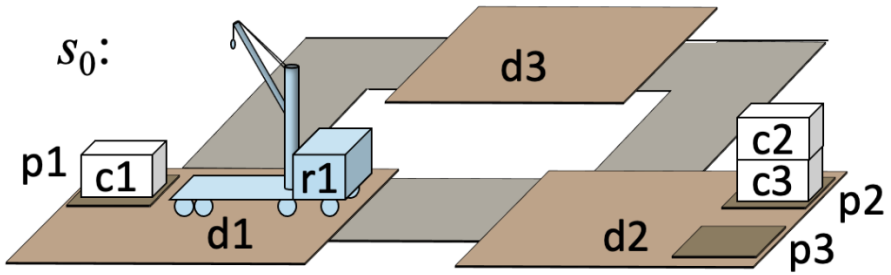
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩





TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

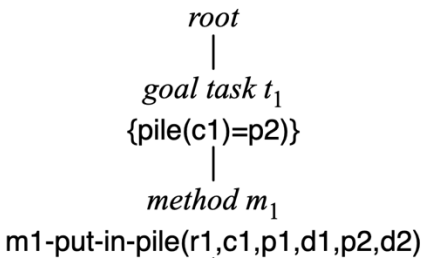
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

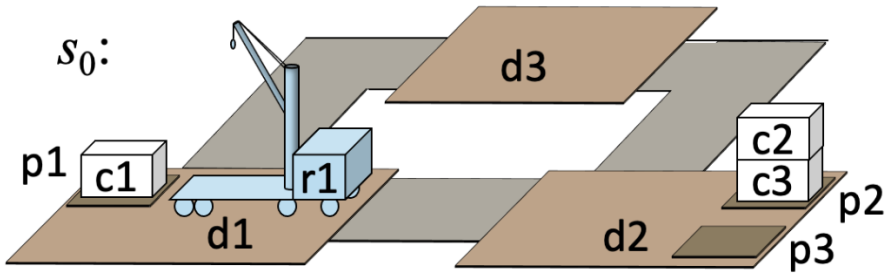
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩





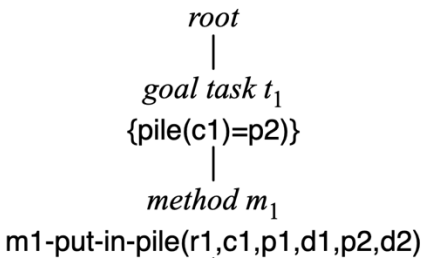

```
TO-HTN-Forward(Σc, M, s, T)
   if T is empty then return ⟨⟩
   t ← the first element of T;  T' ← the rest of T
1  M ← HTN-Get-Candidates(Σc, M, s, t)
   if M = ∅ then return failure
   nondeterministically choose m ∈ M
   switch m do
2     case m is an action do
         π ← TO-HTN-Forward(Σc, M, γ(s, m), T')
         if π ≠ failure then return m·π
         else return failure
3     case m is a method instance do
         return TO-HTN-Forward(Σc, M, s, sub(m)·T')

HTN-Get-Candidates(Σc, M, s, t)
   switch t do
4     case t is an action do
         if t is applicable in s then M ← {a}
         else M ← ∅
5     case t is a compound task do M ← Refiners(s, t, M)
6     case t is a goal task do
         M ← Refiners(s, t, M) ∪ Achievers(s, t)
7        if s ⊨ t then M ← M ∪ {null}
   return M
```

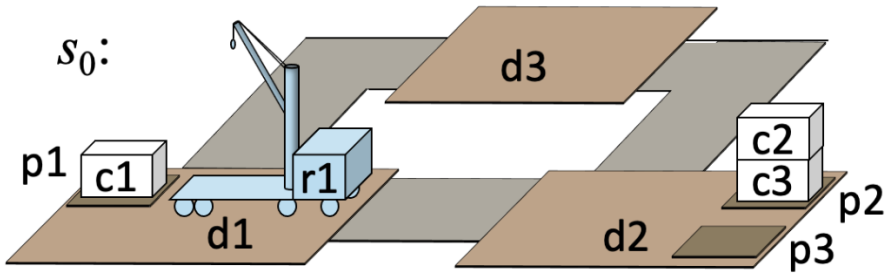
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩





TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s,$ sub$(m) \cdot T'$)

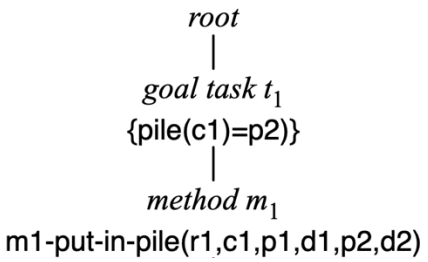
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

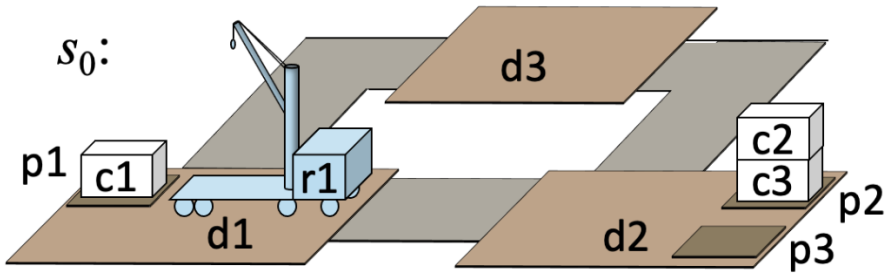
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩





TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
  case m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

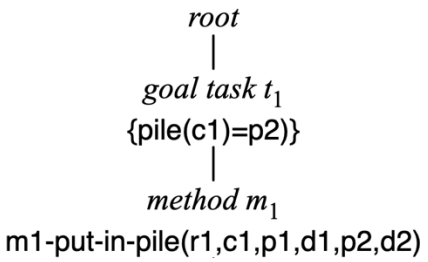
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

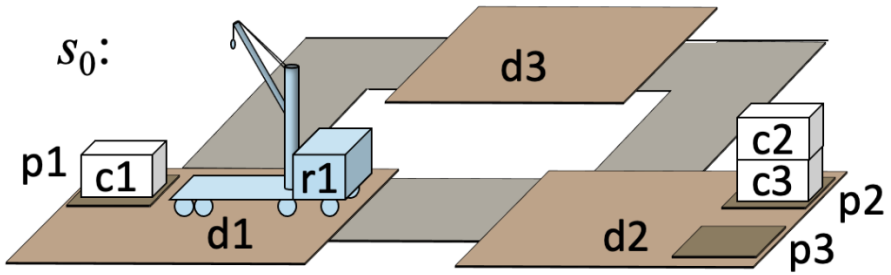
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩



return TO-HTN-Forward(Σ_c,$\mathcal{M}$, s_0, sub(m1-put-in-pile(r1,c1,d1,p2,d2)) ·T')



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s,$ sub$(m) \cdot T'$)

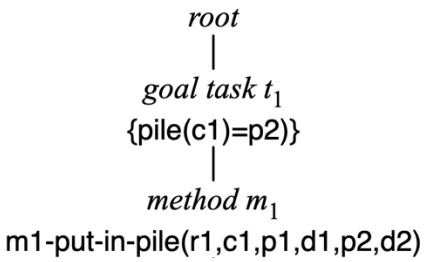
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

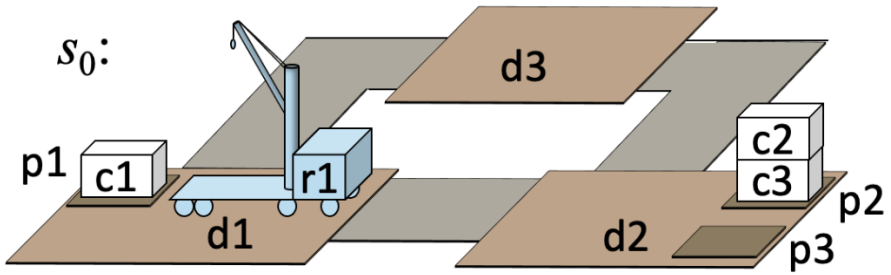
- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

t= {pile(c1) = p2}
T' = ⟨⟩



return TO-HTN-Forward(Σ_c,$\mathcal{M}$, s_0, ⟨get-container(r1,c1), navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩)



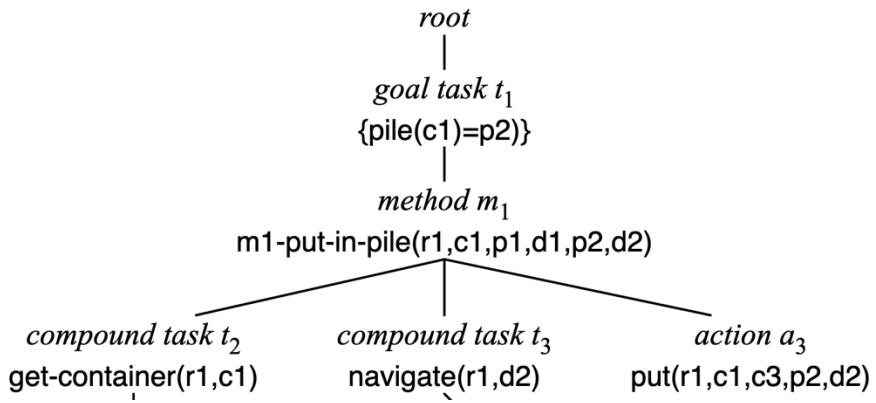
TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s, m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

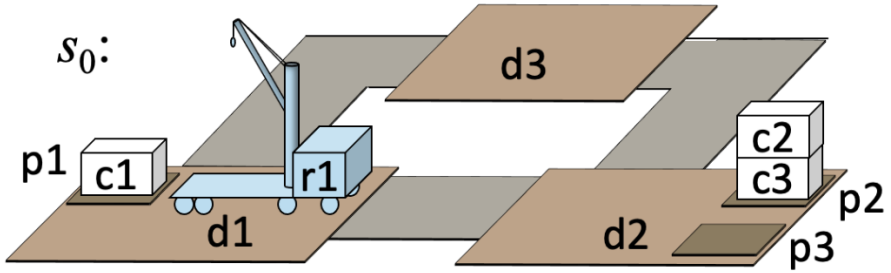
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$



T= ⟨get-container(r1,c1), navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩



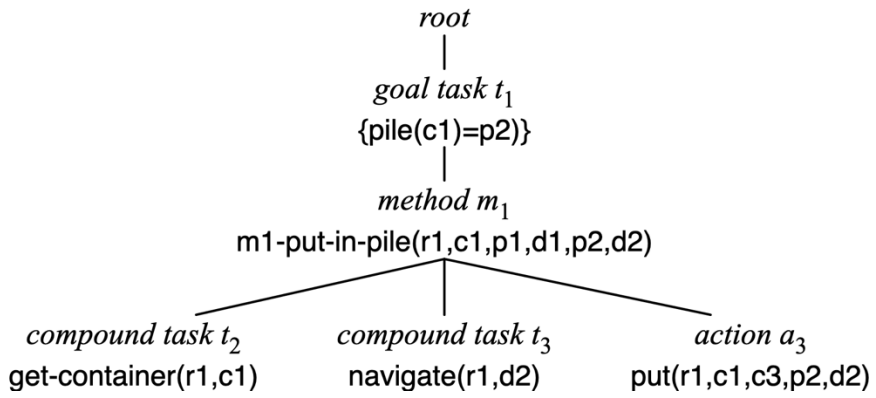
TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

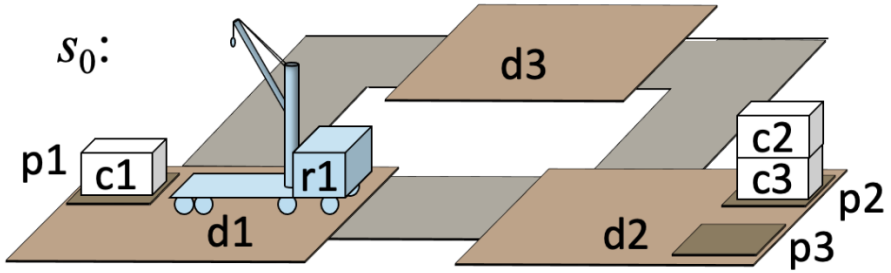
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$



t= get-container(r1,c1)
T'= ⟨navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩



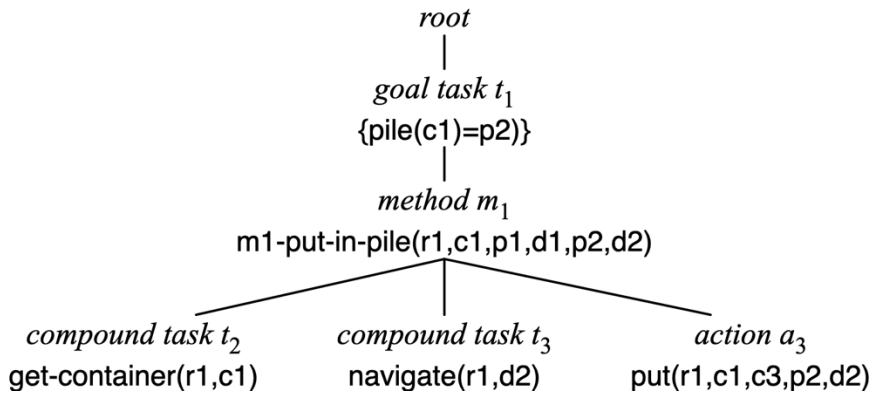
TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

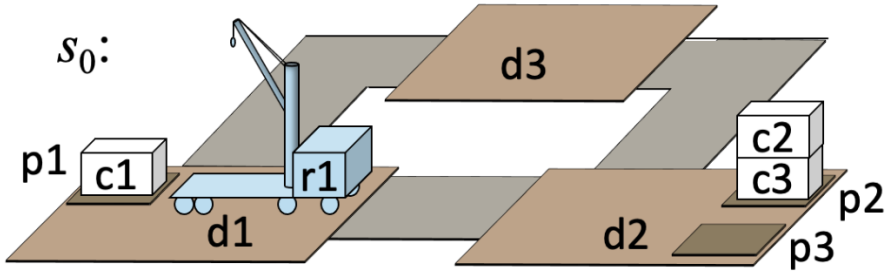
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$



t= get-container(r1,c1)
T'= ⟨navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

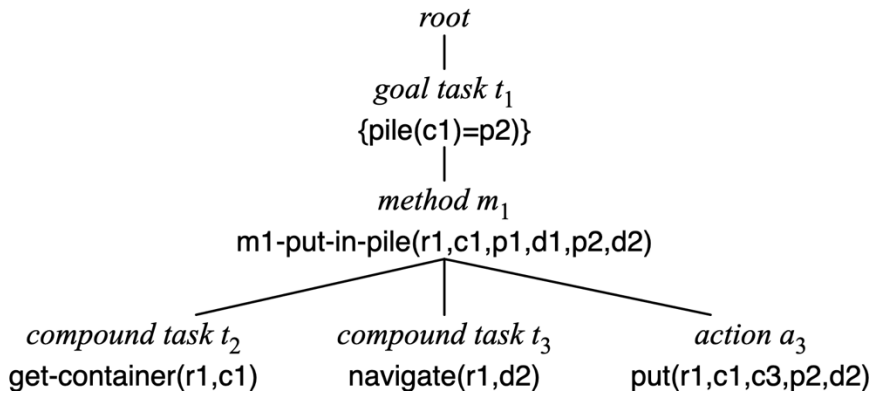
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

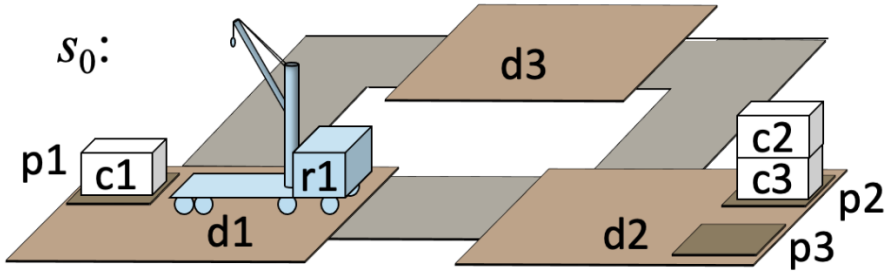
Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, get-container(r1,c1))



t= get-container(r1,c1)

T'= ⟨navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
  $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

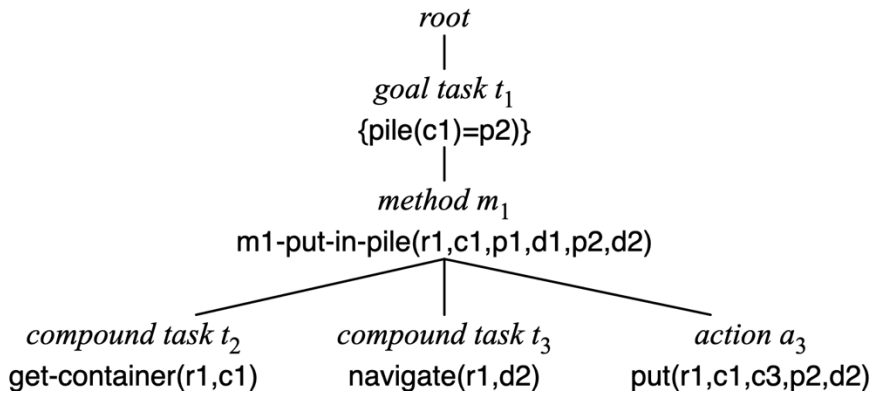
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

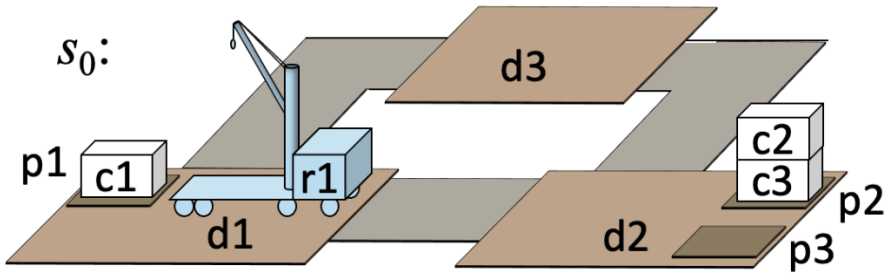
Planning Algorithm

- *Planning problem:* $P = (\Sigma_c, s_0, T)$

  Here, $P=(\Sigma_c, s_0, \langle\{\text{pile(c1)} = \text{p2}\}\rangle)$

  M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, get-container(r1,c1))





TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, \text{sub}(m) \cdot T'$)

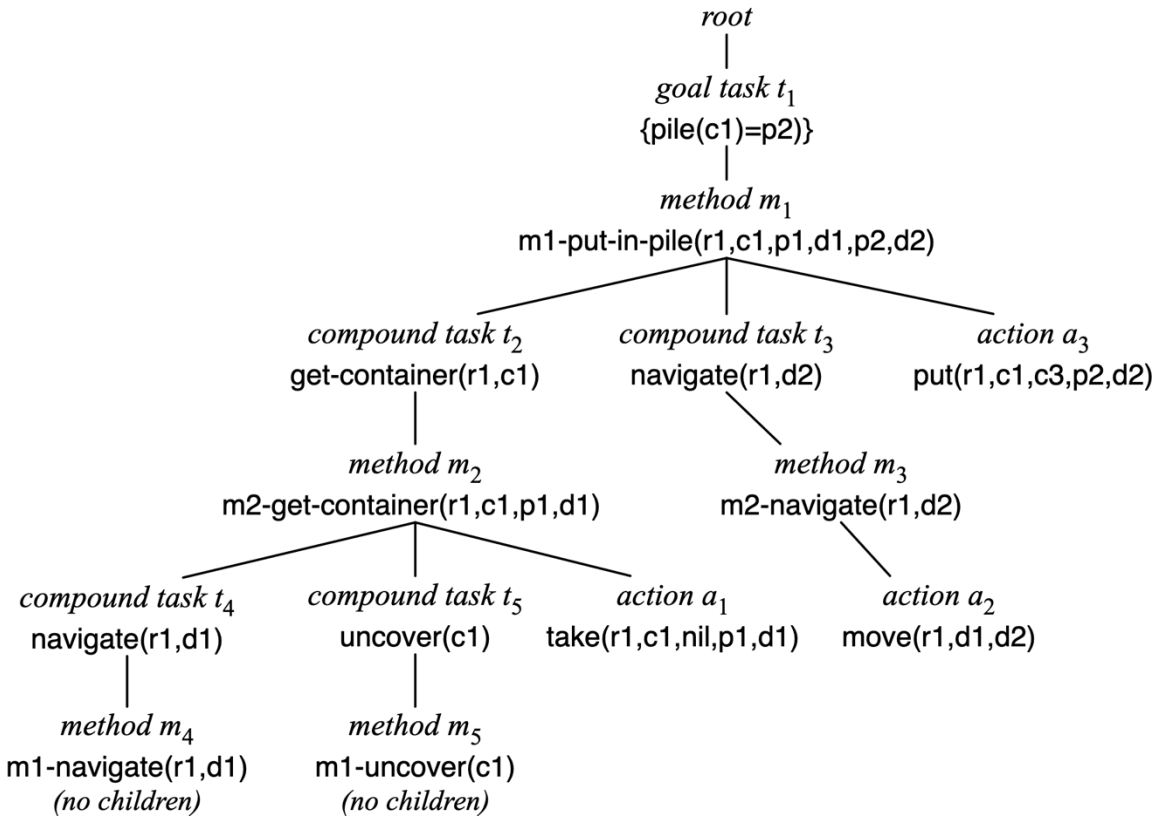
HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{\text{null}\}$
  return $\mathcal{M}$

Planning Algorithm

- Definitions

t= get-container(r1,c1)
T'= ⟨navigate(r1,d2), put(r1,c1,c3,p2,d2)⟩

M = HTN-Get-Candidates(Σ_c,$\mathcal{M}$, s_0, get-container(r1,c1))

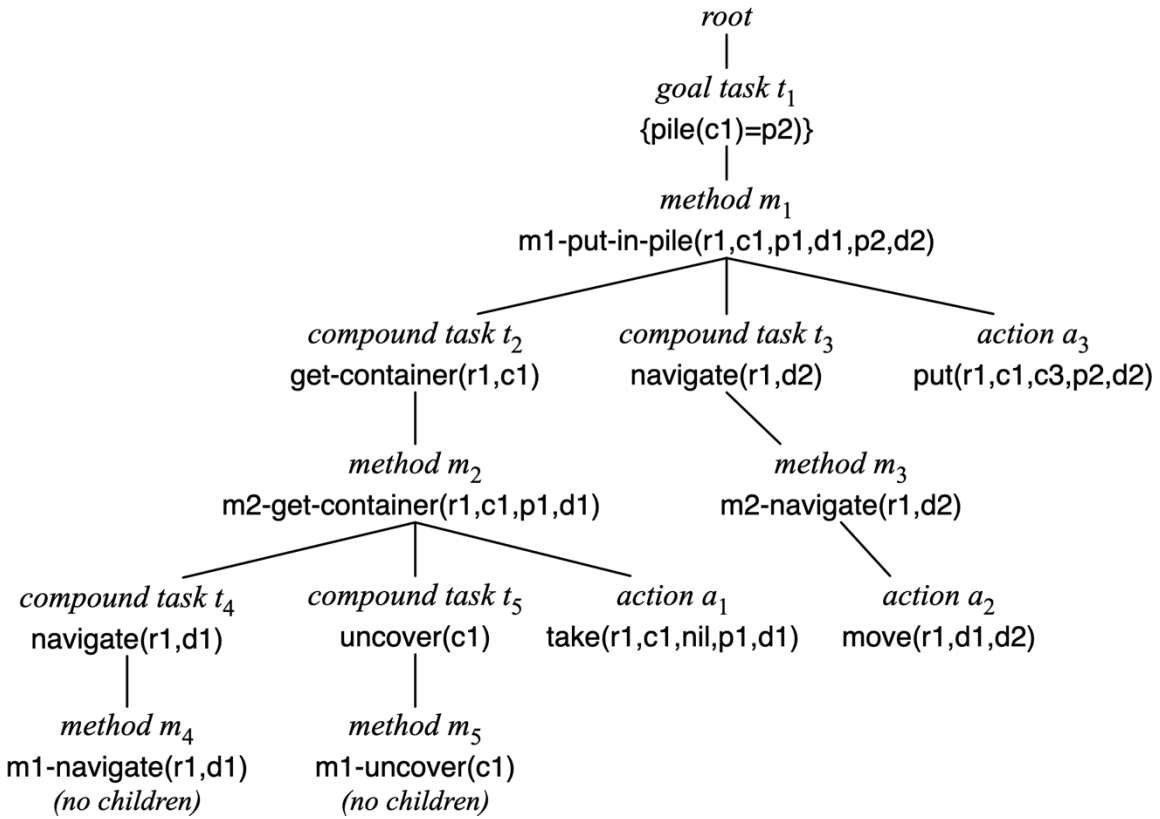


TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s$, sub$(m) \cdot T'$)

HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s,t,\mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s,t,\mathcal{M}) \cup$ *Achievers*(s,t)
7 **if** $s \models t$ **then** $M \leftarrow M \cup \{$null$\}$
  return $\mathcal{M}$

Planning Algorithm

- NOTE: Will update the slides till the portion of the algorithm covered in class.



TO-HTN-Forward($\Sigma_c, \mathcal{M}, s, T$)
  if T is empty **then return** $\langle\rangle$
  $t \leftarrow$ the first element of T; $T' \leftarrow$ the rest of T
1 $M \leftarrow$ HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  if $M = \varnothing$ **then return** failure
  nondeterministically choose $m \in M$
  switch m **do**
2 **case** m is an action **do**
    $\pi \leftarrow$ TO-HTN-Forward($\Sigma_c, \mathcal{M}, \gamma(s,m), T'$)
    if $\pi \neq$ failure **then return** $m \cdot \pi$
    else return failure
3 **case** m is a method instance **do**
    return TO-HTN-Forward($\Sigma_c, \mathcal{M}, s$, sub$(m) \cdot T'$)

HTN-Get-Candidates($\Sigma_c, \mathcal{M}, s, t$)
  switch t **do**
4 **case** t is an action **do**
    if t is applicable in s **then** $M \leftarrow \{a\}$
    else $M \leftarrow \varnothing$
5 **case** t is a compound task **do** $M \leftarrow$ *Refiners*$(s, t, \mathcal{M})$
6 **case** t is a goal task **do**
    $M \leftarrow$ *Refiners*$(s, t, \mathcal{M}) \cup$ *Achievers*(s, t)
7   **if** $s \models t$ **then** $M \leftarrow M \cup \{$null$\}$
  return $\mathcal{M}$