

Classical Planning via Plan-space search

COMP3431

Malcolm Ryan

Total-order planning

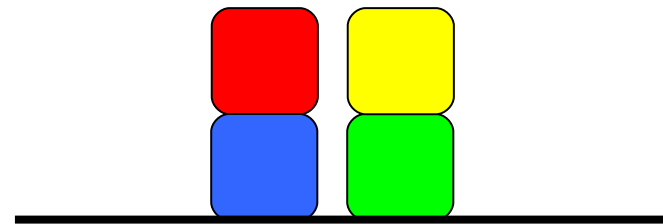
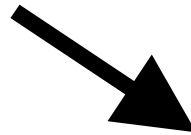
- The state-space planning technique produces **totally-ordered** plans, i.e. plans which consist of a strict sequence of actions.
- Often, however, there are many possible orderings of actions that have equivalent effects.

Example

- Consider the planning problem:



INITIAL



GOAL

Example

- There are many possible plans:

```
move(blue, red, table)
move(red, table, blue)
move(green, yellow, table)
move(yellow, table, green)
```

```
move(green, yellow, table)
move(yellow, table, green)
move(blue, red, table)
move(red, table, blue)
```

```
move(blue, red, table)
move(green, yellow, table)
move(red, table, blue)
move(yellow, table, green)
```

```
move(green, yellow, table)
move(blue, red, table)
move(red, table, blue)
move(yellow, table, green)
```

Example

- These plans share some common structure. In fact, they are all different **interleavings** of two separate plans:

```
move(blue, red, table)
```

```
move(red, table, blue)
```

```
move(green, yellow, table)
```

```
move(yellow, table, green)
```

- A **partial-order** plan is one which specifies only the necessary ordering information. One partial-order plan may have many total-orderings

Plan-space planning

- Plan-space planning is a kind of approach to planning that produces partial-order plans.
- It follows the **least-commitment principle**:
 - Do not add constraints (eg action ordering) to a plan until it becomes necessary to ensure the correctness of the plan.

Planning as plan-space search

- A search through the space of plans.
- Nodes in this search represent **incomplete plans** – plans with some steps missing.
- Edges represent **refinements** – additional actions or constraints that can be added to make new plans.

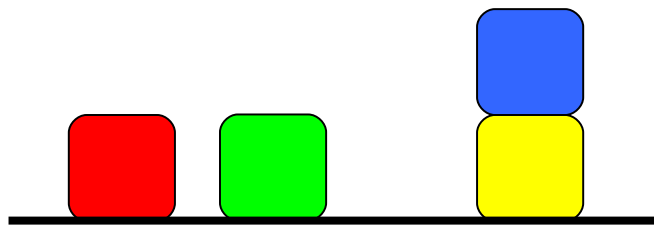
Partial Order Planning

Planning as search:

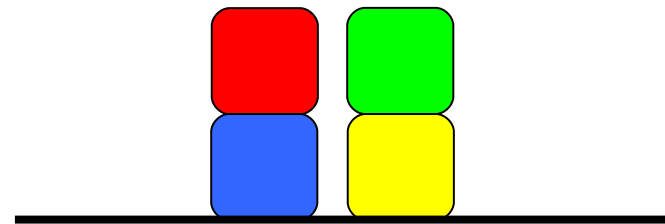
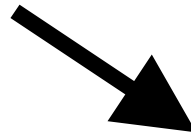
1. Start with the empty plan
2. While there are goals unsatisfied:
 - a. Pick an unsatisfied goal (**Generate**)
 - b. Add an action that satisfies it (**Select**)
 - c. Resolve conflicts (**Refine/Prune**)

Partial Order Planning Example

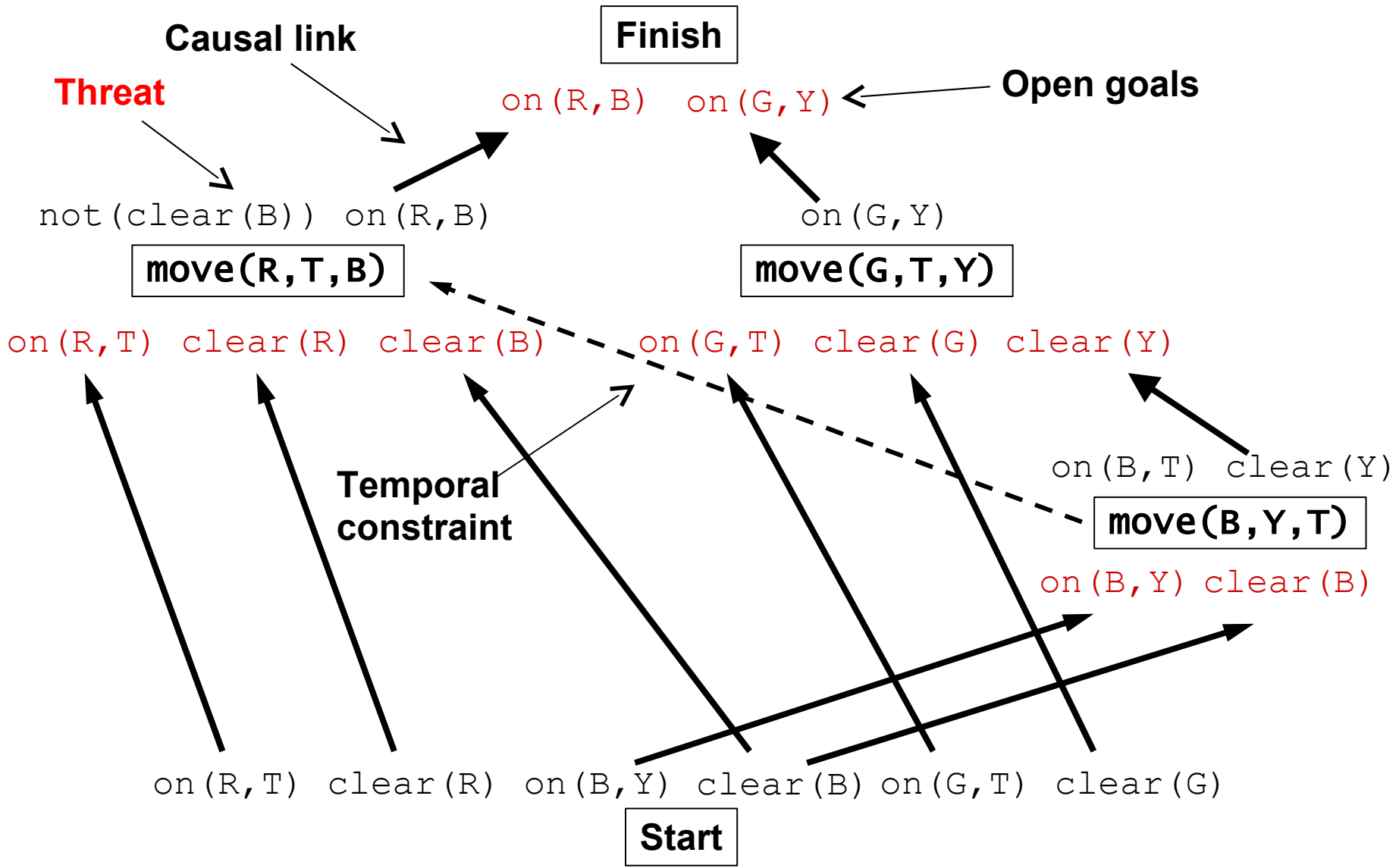
- Four block problem:



INITIAL



GOAL



The POP algorithm

POP($\pi, agenda$)

if $agenda = \emptyset$ **then** return π

select (a_k, p) from $agenda$

$relevant = \text{Providers}(p)$

if ($relevant = \emptyset$) **then** fail

choose a_i from $relevant$

add causal link (a_i, p, a_k) to π

The POP algorithm

add temporal constraint: $a_i < a_k$

if a_i is new **then**

 add constraint: $a_{start} < a_i < a_{finish}$

 add preconditions of a_i to *agenda*

for each new threat **do**

 choose a resolver r

 add r to π

return POP($\pi, agenda$)

Binding constraints

- Temporal ordering is not the only constraint we can consider relaxing
- We can also relax the variable binding procedure, only binding action parameters as they become necessary.

Hierarchical Task Networks

- Plan-space planning can be augmented to use hierarchy.
- We augment our planning system with a collection of abstract **tasks** which are described with preconditions and effects like actions
- We have a **plan library** which includes one or more **methods** for each task. A method is a partial-order plan.

Planning with HTNs

- When we plan with HTNs we extend the POP algorithm to include tasks and their associated methods.
- We can choose tasks as resolvers for open goals.
- We then have to choose a method to implement that task.

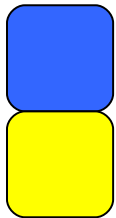
Example Task

- Example task in blocksworld:

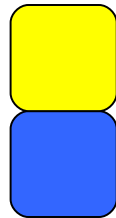
SWAP(*b1*, *b2*)

pre: **on** (**b1**, **b2**) , **clear** (**b1**)

add: **on** (**b2**, **b1**) , **clear** (**b2**)



Before

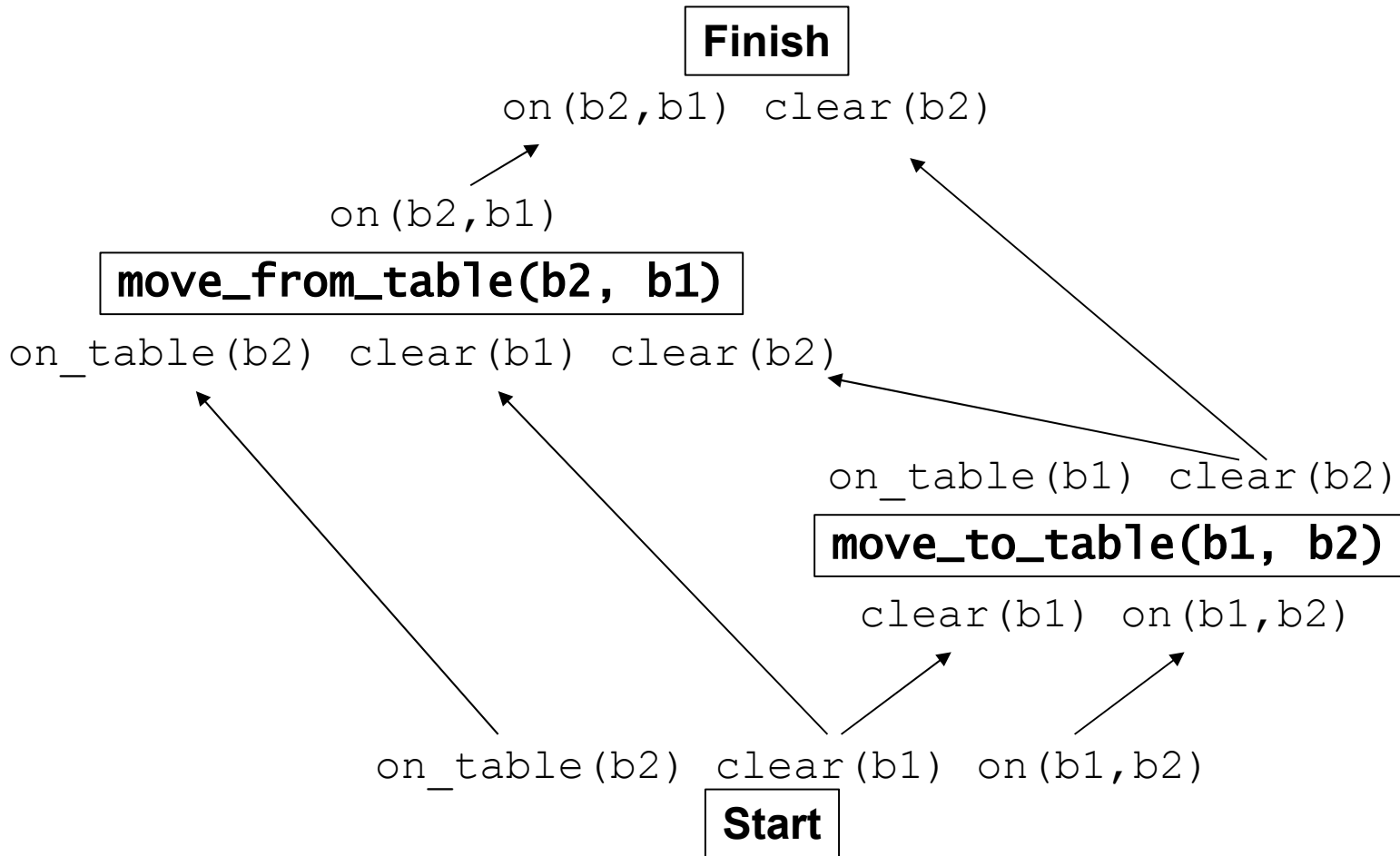


After

Example Methods

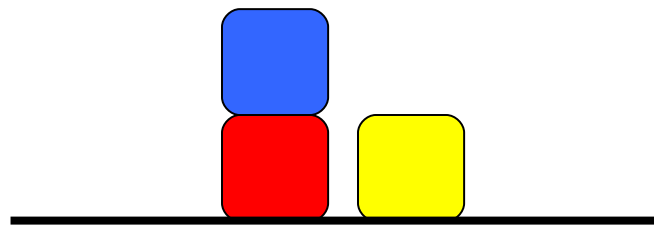
- `SwapWithTable(b1, b2)`
pre: `on(b1, b2), on_table(b2),`
`clear(b1)`
add: `on(b2, b1), clear(b2),`
`on_table(b1)`
- `SwapWithBlock(b1, b2, b3)`
pre: `on(b1, b2), on_table(b2)`
`clear(b1), clear(b3)`
add: `on(b2, b1), clear(b2),`
`on(b1, b3)`

SwapwithTable(b1, b2)



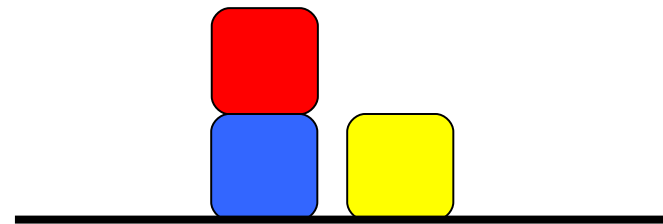
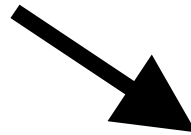
HTN Planning Example

- Consider the planning problem:



INITIAL

on (B, R) clear (B)
on (R, T) on (Y, T)
clear (Y)

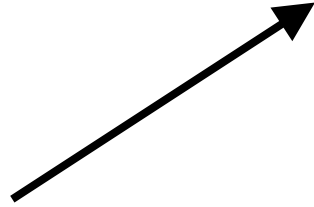


GOAL

on (R, B) on (Y, T) clear (Y)

Finish

on (R, B) on (Y, T) clear (Y)



on (R, B) clear (R)

Swap (B, R)

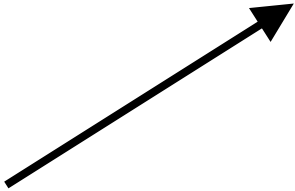
on (B, R) clear (B)

on (B, R) clear (B) on (R, T) on (Y, T) clear (Y)

Start

Finish

on (R, B) on (Y, T) clear (Y)



on (R, B) clear (R)

SwapWithBlock(B, R, Y)

on (B, R) clear (B) on (R, T) clear (Y)

on (B, R) clear (B) on (R, T) on (Y, T) clear (Y)

Start

Finish

on (R,B) on (Y,T) clear (Y)

SwapwithBlock(B, R, Y)

on (R,B)

move_from_table(R, B)

on (R,T) clear (B) clear (R)

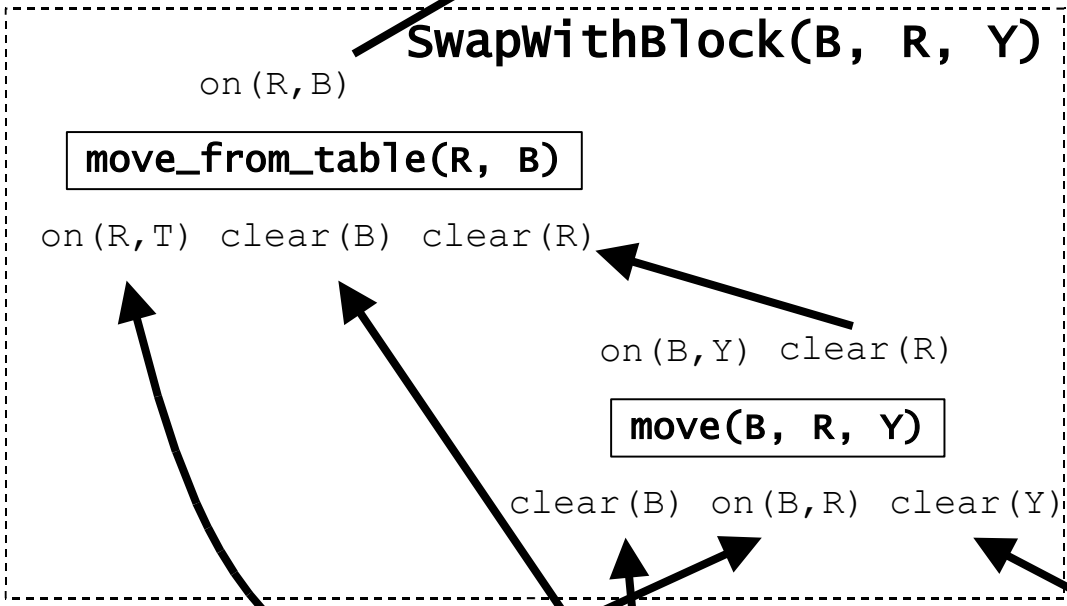
on (B,Y) clear (R)

move(B, R, Y)

clear (B) on (B,R) clear (Y)

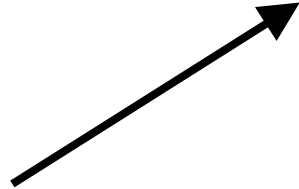
on (B,R) clear (B) on (R,T) on (Y,T) clear (Y)

Start



Finish

on (R, B) on (Y, T) clear (Y)



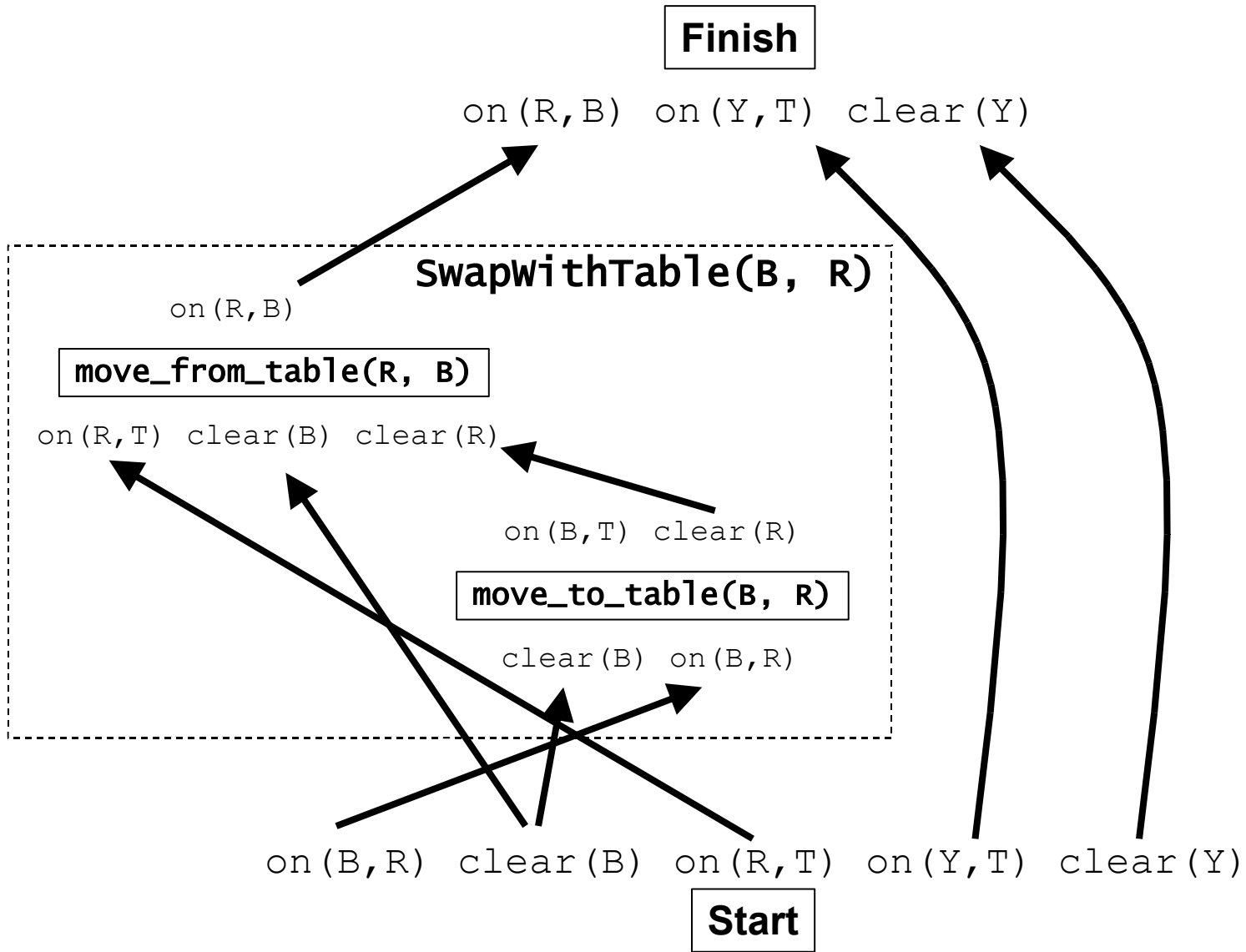
on (R, B) clear (R)

SwapWithTable(B, R)

on (B, R) clear (B) on (R, T)

on (B, R) clear (B) on (R, T) on (Y, T) clear (Y)

Start



Things to consider

- Interleaving actions from different methods
- Re-using actions across different methods
- Recursive HTNs