Ideas for Part 2

https://my.vanderbilt.edu/cs4269aiproject/ideas-for-project-part-2/

- Gamify the simulation
  - Embed simulators into game players
  - Protocol (e.g., Propose, Agree, Assert, Retract)
  - Game manager
    - Manage asynchronous or synchronous communication
    - Carry out changes in the world due to operators of plans being executed
    - Mediate encroachments/war, if relevant

- Learning
  - Learning macro operators
  - Learning weights
  - Reinforcement learning

First Order Planning

Representation of operators: A PREcondition list and an EFFects list

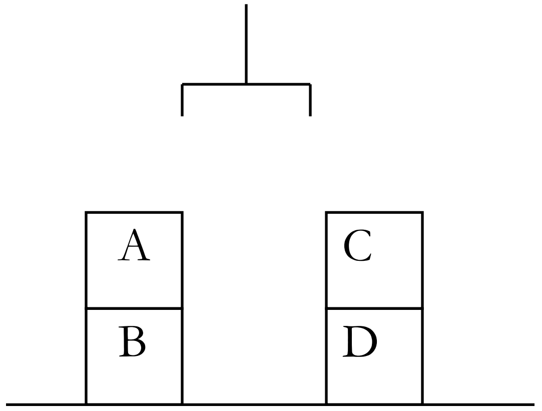
pickup(?x): PRE: ONTABLE(?x), CLEAR(?x), HANDEMPTY
EFF: ~ONTABLE(?x), ~CLEAR(?x), ~HANDEMPTY,
HOLDING(?x)

putdown(?x): PRE: HOLDING(?x)
EFF: ~HOLDING(?x),
ONTABLE(?x), CLEAR(?x), HANDEMPTY

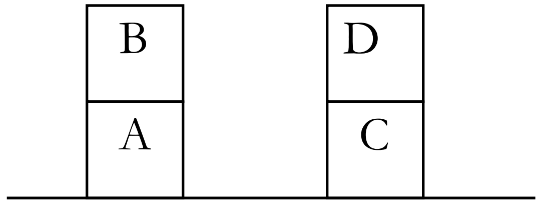
stack(?x, ?y): PRE: HOLDING(?x), CLEAR(?y)
EFF: ~HOLDING(?x), ~CLEAR(?y),
HANDEMPTY, ON(?x, ?y), CLEAR(?x)

unstack(?x, ?y): PRE: HANDEMPTY, CLEAR(?x), ON(?x,?y)
EFF: ~HANDEMPTY, ~CLEAR(?x), ~ON(?x,?y),
HOLDING(?x), CLEAR(?y)

A Planning Problem



Initial State



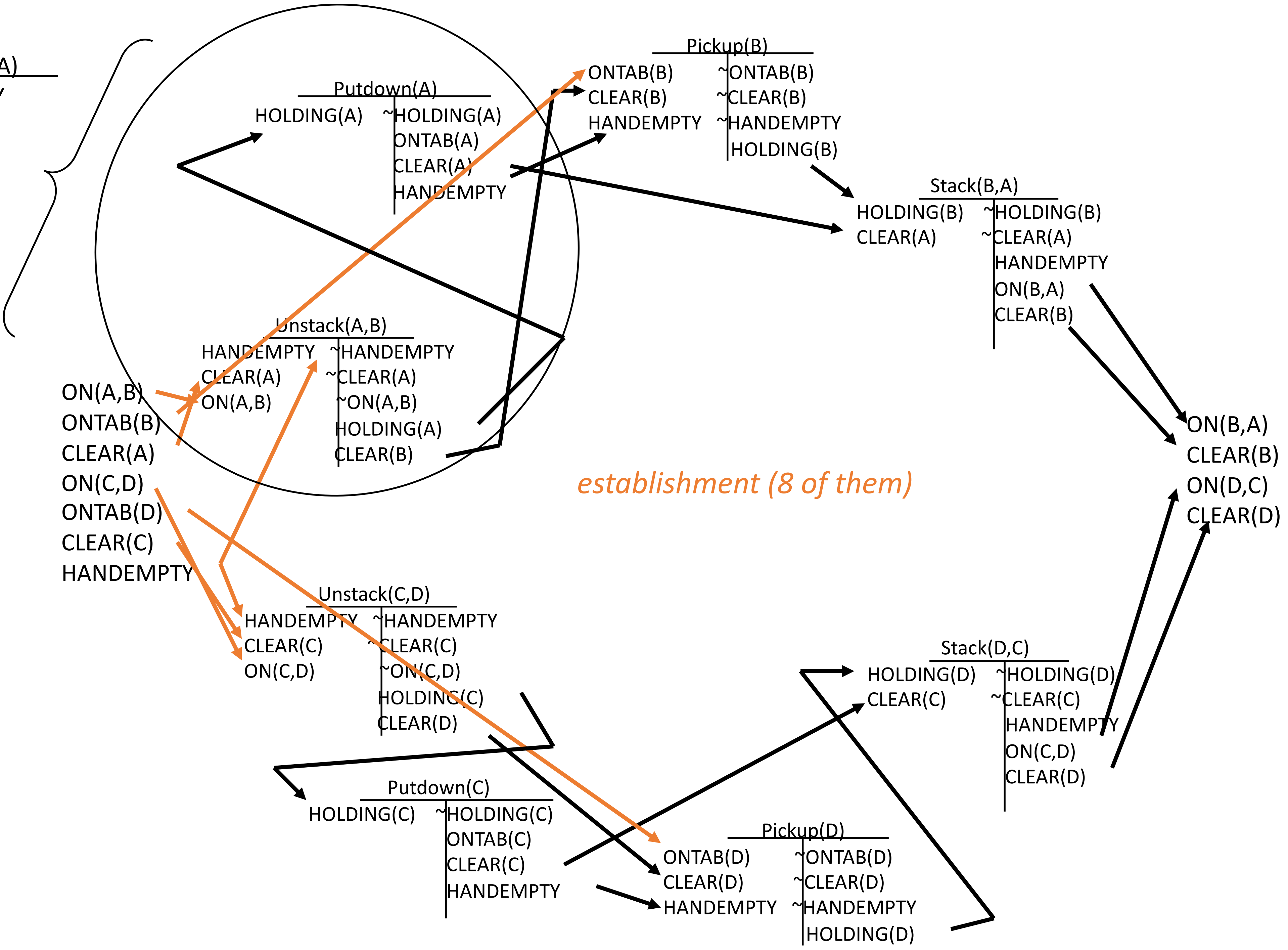
Goal spec

ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY

ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)

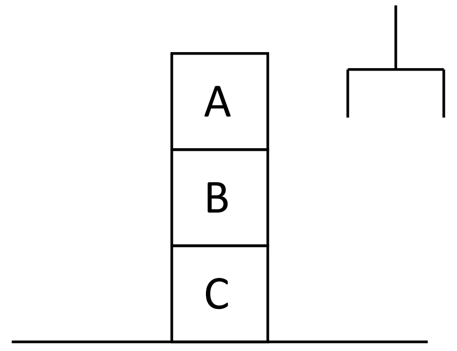
Unstack(A,B)→PutDown(A)	
HANDEMPTY	HANDEMPTY
CLEAR(A)	CLEAR(A)
ON(A,B)	~ON(AB)
	CLEAR(B)
	ONTAB(A)

A “macro” operator

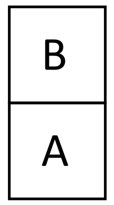


Learning macros: Given a plan, generalize the plan so that the generalized plan can be applied in a greater number of situations

Objective: reusing previously-developed generalized plans (aka macro-operators) will reduce the cost (improve the "speed") of subsequent planning



Start State



GoalSpec

Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

(Generalize) →

Unstack(?x1, ?y1) → Putdown(?x1) → Unstack(?y1, ?z1) → Stack(?y1, ?x1)

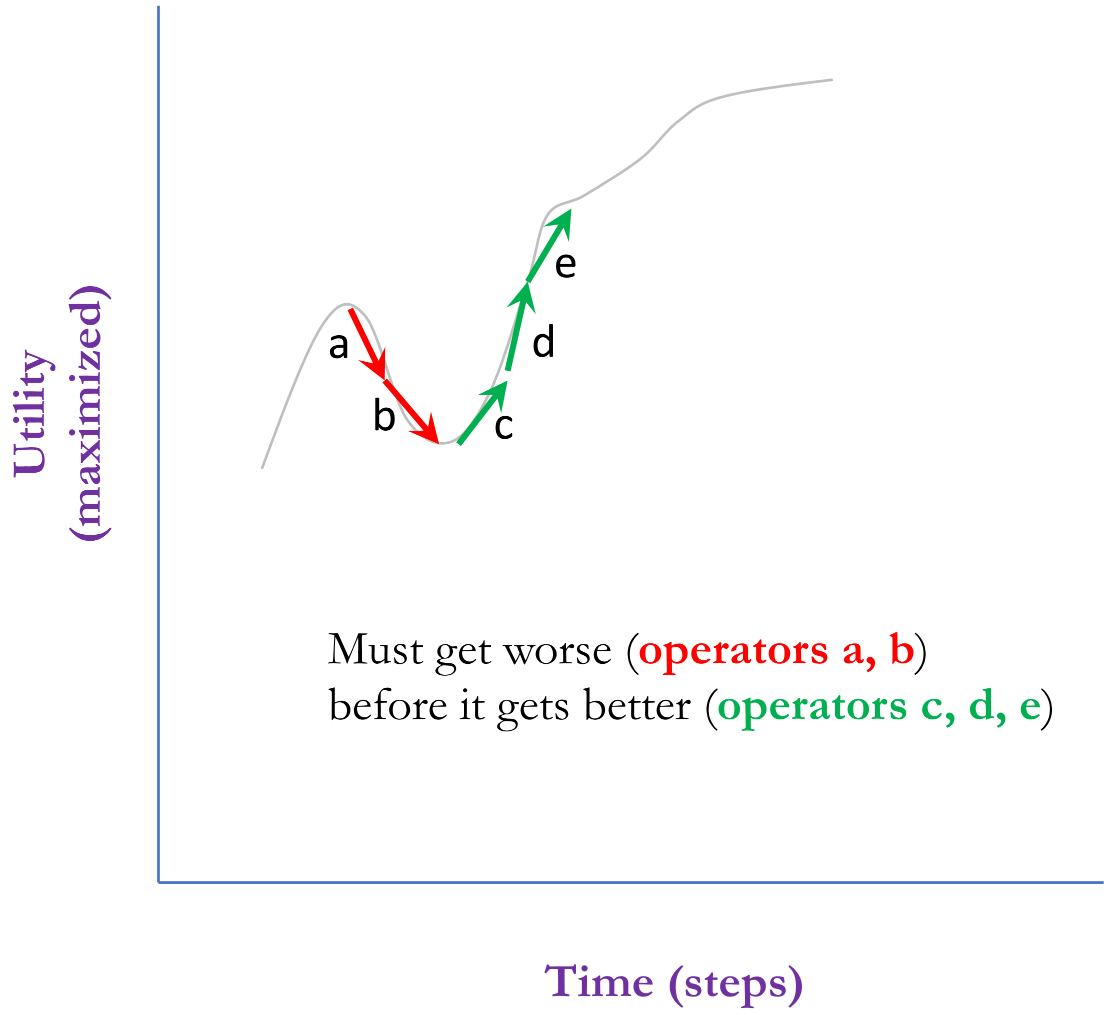
Why macros?

- decrease effective depth (good)
- but increase effective breadth (bad)
- so macros have to be accompanied by better search control – that is, better ways of choosing operators to apply

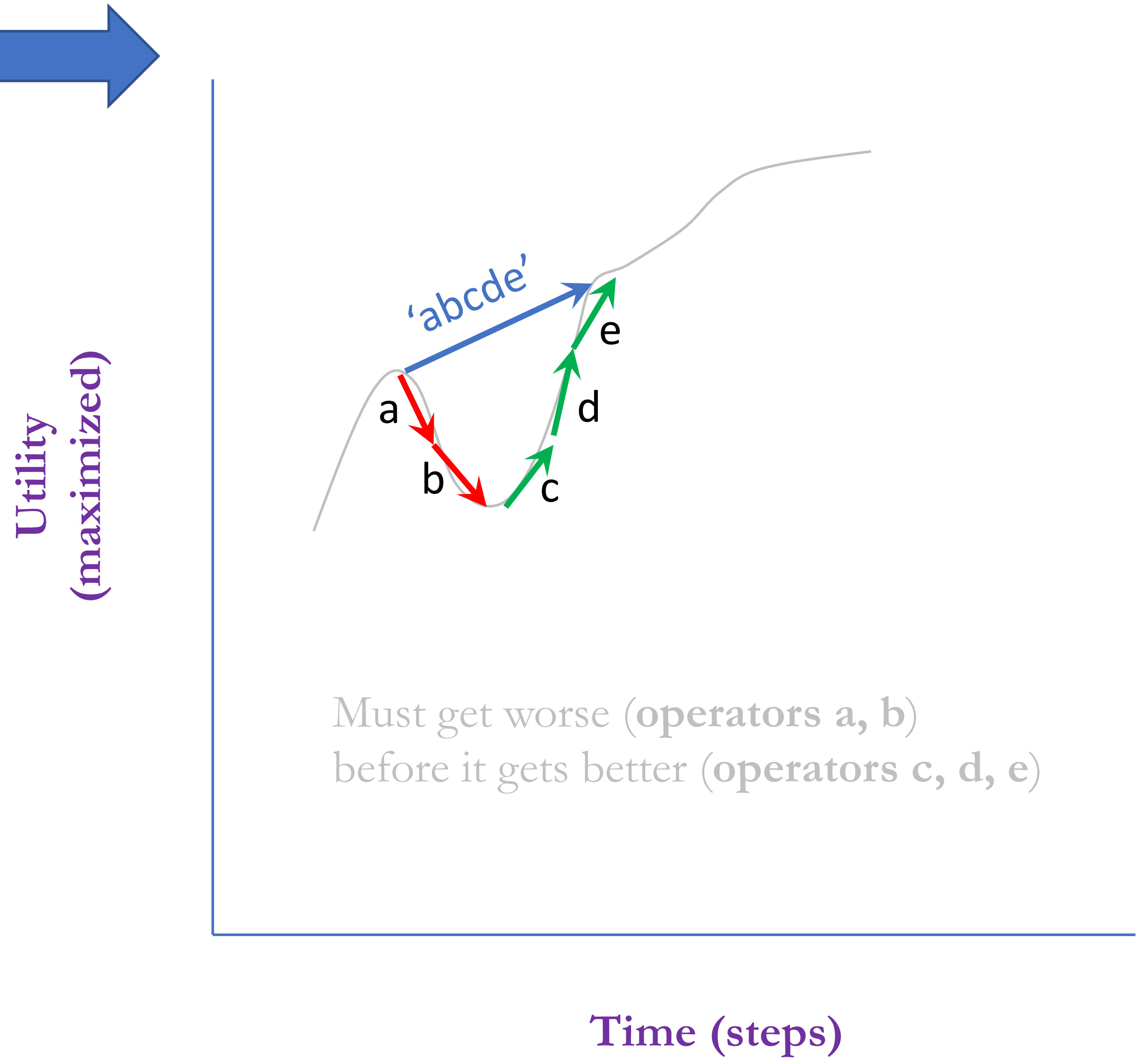
But what are the mechanisms by which an AI (or human) might learn macros?

- Form a macro from frequently applied operator sequences (the frequency reason)
- Form a macro that bridges a "misleading" region of the search space (the informedness reason) – **see graph next slide**
- These two reasons can be simultaneously important in a macro's formation

Add **macro 'abcde'** that bridges the search valley
(either "hand code" macro from human developer domain knowledge, and/or machine learn it with "experience")



Must get worse (**operators a, b**)
before it gets better (**operators c, d, e**)



Must get worse (**operators a, b**)
before it gets better (**operators c, d, e**)

Don't get rid of individual ops a, b, c, d, e – may need/want them in other circumstances

Trade macros at varying levels of specificity

You can "hand-code" this for Part 1, but can learn it for Part 2

```
(
  (TRANSFER ?C_i ?C_k (RESOURCES (?R_1j ?X_1j)))
  (TRANSFER ?C_k ?C_i (RESOURCES (?R_1l ?Y_1l)))
)

(
  (TRANSFER self ?C_k (RESOURCES (?R_1j ?X_1j)))
  (TRANSFER ?C_k self (RESOURCES (?R_1l ?Y_1l)))
)

(
  (TRANSFER self ?C_k (RESOURCES (Timber ?X_1j)))
  (TRANSFER ?C_k self (RESOURCES (MetallicElements ?Y_1l)))
)

(
  (TRANSFER self Camria (RESOURCES (Timber ?X_1j)))
  (TRANSFER Camria self (RESOURCES (MetallicElements ?Y_1l)))
)
```

From https://my.vanderbilt.edu/cs4269aiproject/project-description-and-logistics/

“**(TRANSFER C_i C_k ((R_1j X_1j) … (R_mj, X_mj)))** : C_i gives various amounts (X) of resources (R) to country C_k. These amounts are subtracted and added from the respective stores of resources in C_i and C_k. The TRANSFER of a list of resources is actually just shorthand for a list of one resource TRANSFERs, as shown below. That is there is no relationship between the amounts of different resources in a list.

The shorthand above can be written as a set of singleton transfers.

- **(TRANSFER C_i C_k ((R_1j X_1j)))**
- …
- **(TRANSFER C_i C_k ((R_mj, X_mj)))** ”

“For part 1, consider using only singleton TRANSFERs, with composite TRANSFERs potentially being used in Part 2 to implement macro operators (more later).”

A larger alliance

Packages of transfers can also be used to implement “alliances”. Two countries might collaborate to pool their resources so as to enable one country (the recipient) to make trades on behalf of itself and one or more other countries.

- **(TRANSFER ?C_i ?C_k (RESOURCES (Timber …)))**
- **(TRANSFER ?C_k self (RESOURCES (MetalicElements ….)))**
- **(TRANSFORM self (MetalicElements … to MetalicAlloys …)**
- **(TRANSFORM self (Timber … to Housing …)**
- **(TRANSFER self ?C_i (RESOURCES (MetalicAlloys …)))**
- **(TRANSFER self ?C_k (RESOURCES (Housing …)))**

In this example, C_i gives resources to C_k, which then transfers resources to C_p, which then transforms distributes resources back to C_i and C_k. There are other ways to write this, which in theory are equivalent.