COMP310 – Multi Agent Systems
Tutorial 3 –
Practical Reasoning

Lecturer: Dr T Carroll

Email: Thomas.Carroll2@Liverpool.ac.uk

Office: G.14

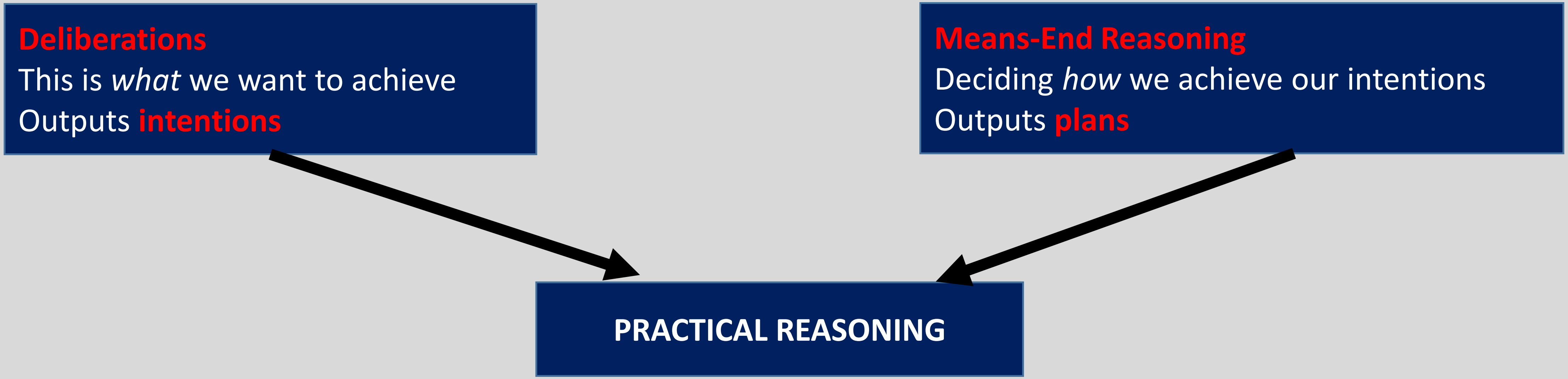
See vital for all material

Pro-active behaviour

- An intelligent agent is a computer system capable of flexible autonomous action in some environment, ie:
  - Reactive
  - Pro-active
  - Social
- We now deal with the **pro-active** bit, and show how we can program agents to have goal-directed behaviour

What is Practical Reasoning?

- Reasoning directed towards **actions**
  - Weighing up the pros and cons of competing options, which are provided by the agents values and beliefs
  - Eg: “Shall I get the bus or the train to work?”



Deliberations
This is *what* we want to achieve
Outputs **intentions**

Means-End Reasoning
Deciding *how* we achieve our intentions
Outputs **plans**

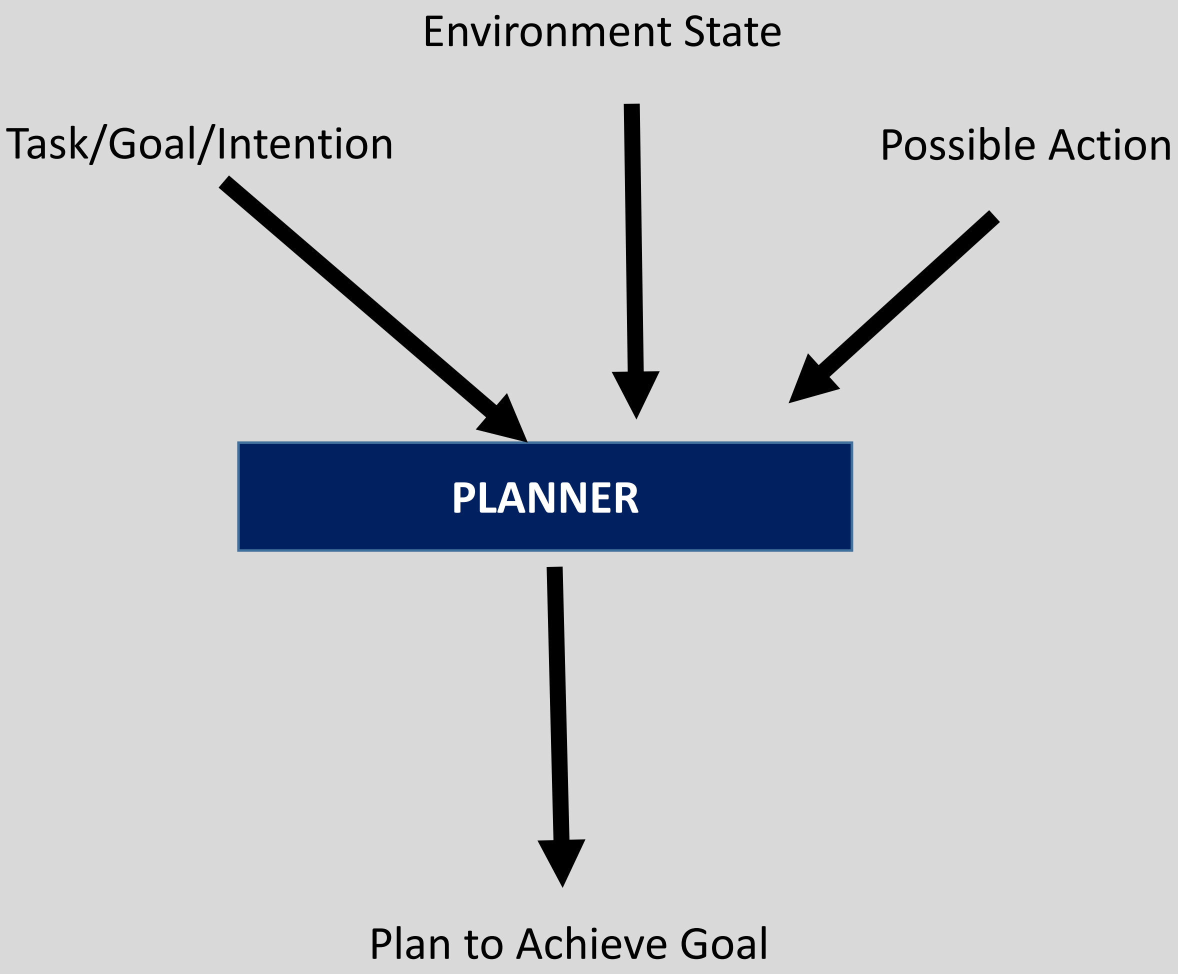
PRACTICAL REASONING

Intentions in Practical Reasoning

1. Intentions pose **problems**, agents must solve the problems
2. New intentions must not **conflict** with existing intentions
3. Agents track their **success** and may try again if they fail
4. Agents will adopt only intentions they think are **possible**
5. Agents do not believe they will **fail** at bringing about their intentions
6. Agents believe that, given the right circumstances, they **can achieve their intentions**
7. Agents need not *intend* to inflict the **side effects** of their intentions
8. Intentions are **stronger than** desires

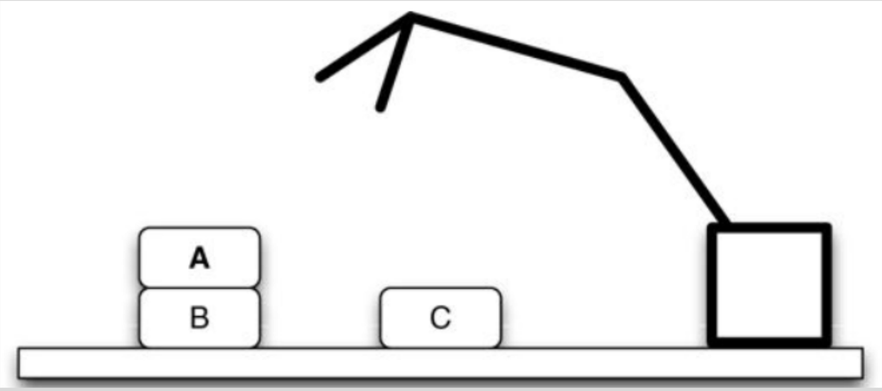
Means End Reasoning and Planning

- Planning is the design of a course of action to achieve a particular goal
- It requires:
  - Goal to achieve
  - Actions it can perform
  - The Environment
- It *automatically* generates a **plan**
- **STRIPS** can be used to represent all of this



Blocksworld

- We'll illustrate the technique using Blockswrld.
- Blocks World contains three blocks (A, B, and C) of equal size
- A robot arm capable of picking up and moving one block at a time, and a table top.
- The blocks may be placed on the table top, or may be placed one on top of another block
- The goal: all the blocks are on the table



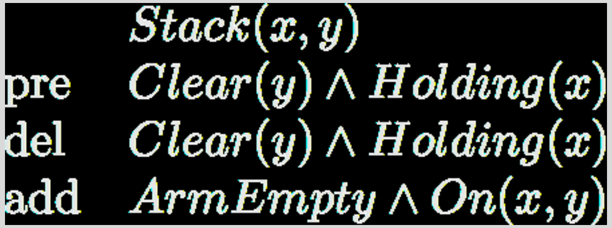
Blocksworld - Representation

- BlocksWorld predicates:
- *On(x,y)   object x on top of object y*
- *OnTable(x) object x is on the table*
- *Clear(x) nothing is on top of object*
- *Holding(x) arm is holding x*
- ArmEmpty   Arm is holding nothing

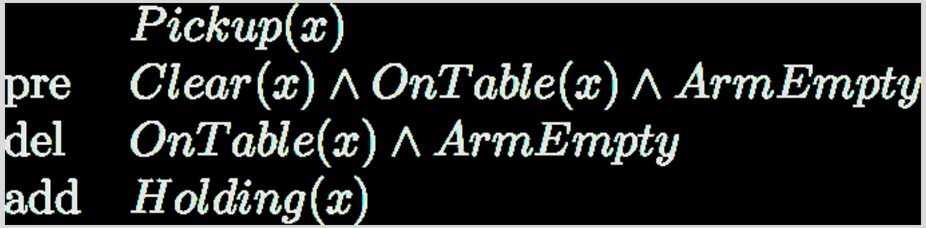
- Each action has:
  - a **name**: which may have arguments;
  - a ***pre-condition list***: list of facts which must be true for action to be executed;
  - a ***delete list***: list of facts that are no longer true after action is performed;
  - an ***add list***: list of facts made true by executing the action.

A plan is a sequence of actions

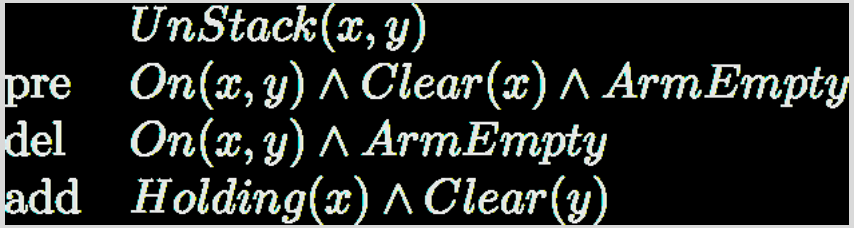
Blocksworld Actions



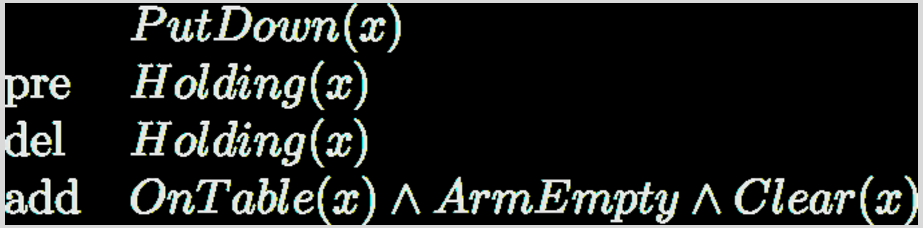
$Stack(x, y)$
pre $Clear(y) \wedge Holding(x)$
del $Clear(y) \wedge Holding(x)$
add $ArmEmpty \wedge On(x, y)$



$Pickup(x)$
pre $Clear(x) \wedge OnTable(x) \wedge ArmEmpty$
del $OnTable(x) \wedge ArmEmpty$
add $Holding(x)$



$UnStack(x, y)$
pre $On(x, y) \wedge Clear(x) \wedge ArmEmpty$
del $On(x, y) \wedge ArmEmpty$
add $Holding(x) \wedge Clear(y)$



$PutDown(x)$
pre $Holding(x)$
del $Holding(x)$
add $OnTable(x) \wedge ArmEmpty \wedge Clear(x)$

Task

- Initial State:

OnTable(A), OnTable(B), OnTable(C)

Clear(A), Clear(B), Clear(C)

ArmEmpty

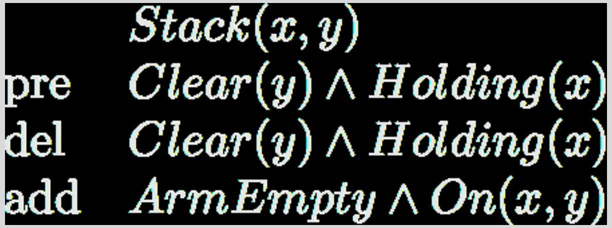
- Goal State:

On(A,B), On(B,C), OnTable(C)

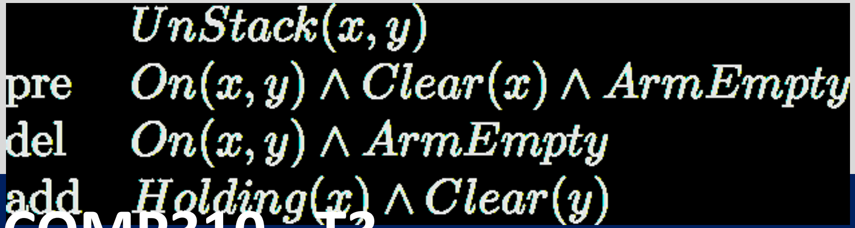
Clear(A)

ArmEmpty

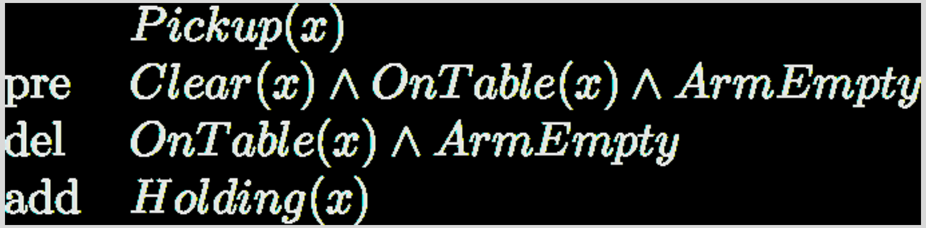
Come up with a **plan** to achieve this goal state



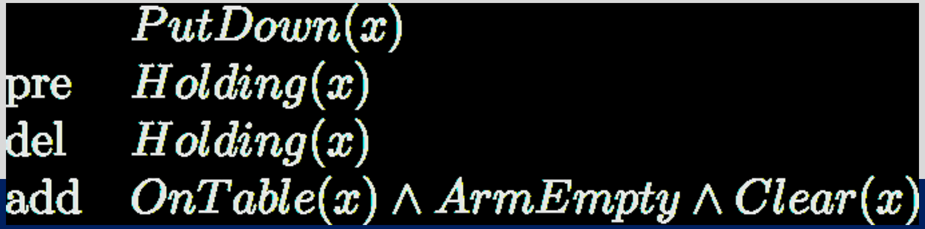
	$Stack(x,y)$
pre	$Clear(y) \land Holding(x)$
del	$Clear(y) \land Holding(x)$
add	$ArmEmpty \land On(x,y)$



	$UnStack(x,y)$
pre	$On(x,y) \land Clear(x) \land ArmEmpty$
del	$On(x,y) \land ArmEmpty$
add	$Holding(x) \land Clear(y)$



	$Pickup(x)$
pre	$Clear(x) \land OnTable(x) \land ArmEmpty$
del	$OnTable(x) \land ArmEmpty$
add	$Holding(x)$



	$PutDown(x)$
pre	$Holding(x)$
del	$Holding(x)$
add	$OnTable(x) \land ArmEmpty \land Clear(x)$