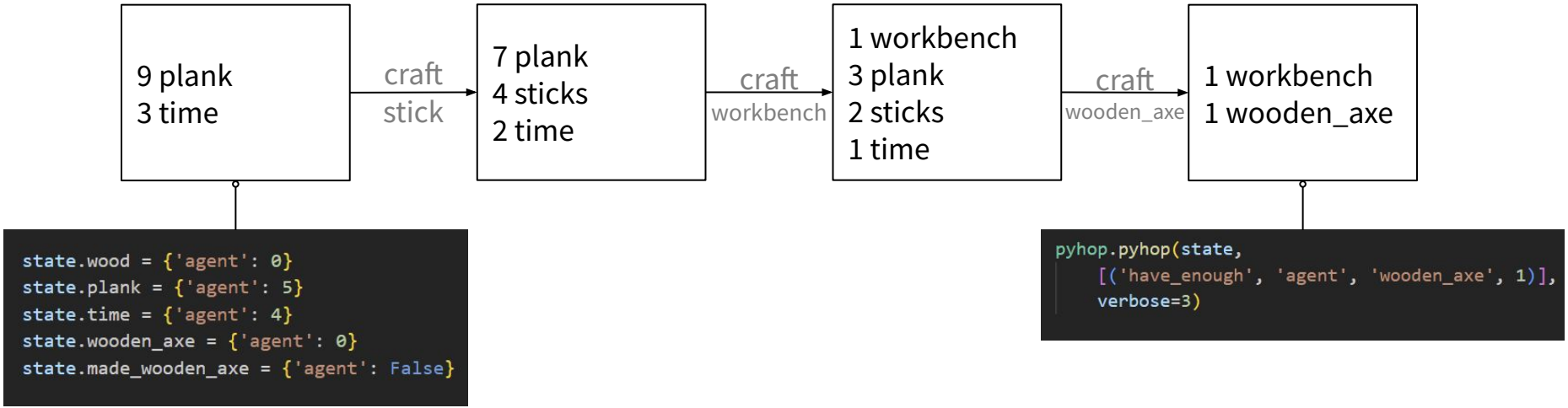
P4: Minecraft HTN

Winter 2025 - Game AI

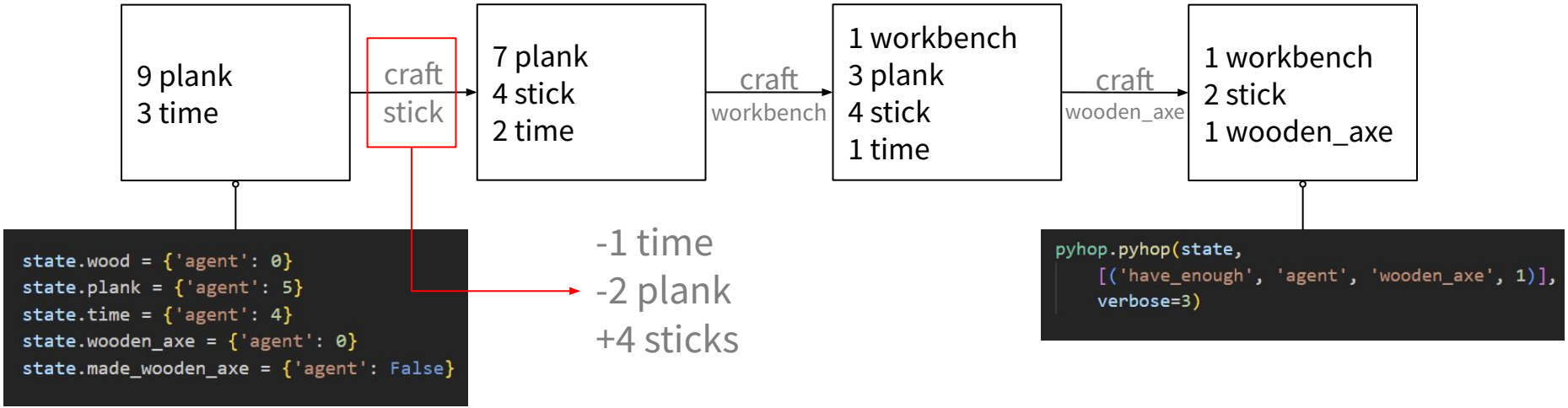
Assignment

Given some starting state, using Minecraft crafting recipes, can we reach an end state?



Assignment

Given some starting state, using Minecraft crafting recipes, can we reach an end state?



Manual HTN

Operators:

- If (have all the requirements - time, consumables, tools)
  - Consume any consumables
  - Consume time
  - Get the item
  - Return the new state
- Otherwise, return false (not possible)

At the end, **declare all the operators** in a single declare_operators call

Manual HTN

```python
def op_punch_for_wood (state, ID):
    if state.time[ID] >= 4:
        state.wood[ID] += 1
        state.time[ID] -= 4
        return state
    return False

def op_craft_wooden_axe_at_bench (state,
    if state.time[ID] >= 1 and state.benc
        state.wooden_axe[ID] += 1
        state.plank[ID] -= 3
        state.stick[ID] -= 2
        state.time[ID] -= 1
        return state
    return False

# your code here

pyhop.declare_operators (op_punch_for_woo
```

Function

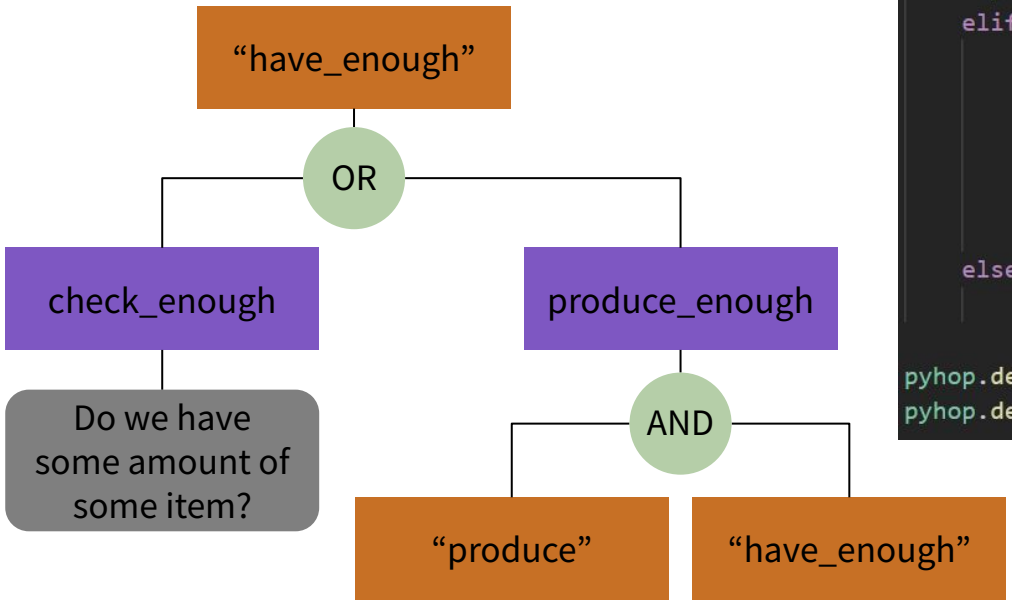
Operators

Pyhop Method

```python
def check_enough (state, ID, item, num):
    if getattr(state,item)[ID] >= num: return []
    return False

def produce_enough (state, ID, item, num):
    return [('produce', ID, item), ('have_enough', ID, item, num)]

def produce (state, ID, item):
    if item == 'wood':
        return [('produce_wood', ID)]
    # your code here
    elif item == 'wooden_axe':
        # this check to make sure we're not making multiple axes
        if state.made_wooden_axe[ID] is True:
            return False
        else:
            state.made_wooden_axe[ID] = True
        return [('produce_wooden_axe', ID)]
    else:
        return False

pyhop.declare_methods ('have_enough', check_enough, produce_enough)
pyhop.declare_methods ('produce', produce)
```

Manual HTN

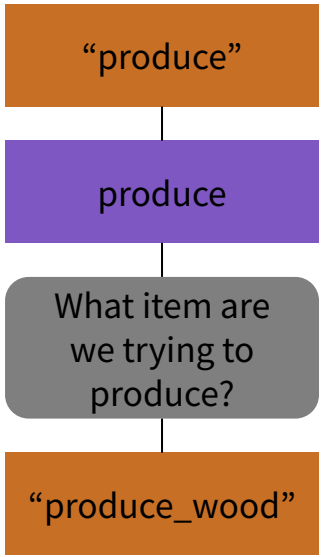
Methods




```python
def check_enough (state, ID, item, num):
    if getattr(state,item)[ID] >= num: return []
    return False

def produce_enough (state, ID, item, num):
    return [('produce', ID, item), ('have_enough', ID, item, num)]

def produce (state, ID, item):
    if item == 'wood':
        return [('produce_wood', ID)]
    # your code here
    elif item == 'wooden_axe':
        # this check to make sure we're not making multiple axes
        if state.made_wooden_axe[ID] is True:
            return False
        else:
            state.made_wooden_axe[ID] = True
        return [('produce_wooden_axe', ID)]
    else:
        return False

pyhop.declare_methods ('have_enough', check_enough, produce_enough)
pyhop.declare_methods ('produce', produce)
```

Manual HTN

Methods




```python
def check_enough (state, ID, item, num):
    if getattr(state,item)[ID] >= num: return []
    return False

def produce_enough (state, ID, item, num):
    return [('produce', ID, item), ('have_enough', ID, item, num)]

def produce (state, ID, item):
    if item == 'wood':
        return [('produce_wood', ID)]
    # your code here
    elif item == 'wooden_axe':
        # this check to make sure we're not making multiple axes
        if state.made_wooden_axe[ID] is True:
            return False
        else:
            state.made_wooden_axe[ID] = True
        return [('produce_wooden_axe', ID)]
    else:
        return False

pyhop.declare_methods ('have_enough', check_enough, produce_enough)
pyhop.declare_methods ('produce', produce)
```

Manual HTN

Methods - Part 1:

- Define for each item, which “produce_???” to call
- For tools, check if we already have that tool

```
def check_enough (state, ID, item, num):
    if getattr(state,item)[ID] >= num: return []
    return False

def produce_enough (state, ID, item, num):
    return [('produce', ID, item), ('have_enough', ID, item, num)]
```

```
def produce (state, ID, item):
    if item == 'wood':
        return [('produce_wood', ID)]
    # your code here
    elif item == 'wooden_axe':
        # this check to make sure we're not making multiple axes
        if state.made_wooden_axe[ID] is True:
            return False
        else:
            state.made_wooden_axe[ID] = True
        return [('produce_wooden_axe', ID)]
    else:
        return False
```

```
pyhop.declare_methods ('have_enough', check_enough, produce_enough)
pyhop.declare_methods ('produce', produce)
```

Manual HTN

Methods - Part 2:

Define specific production methods

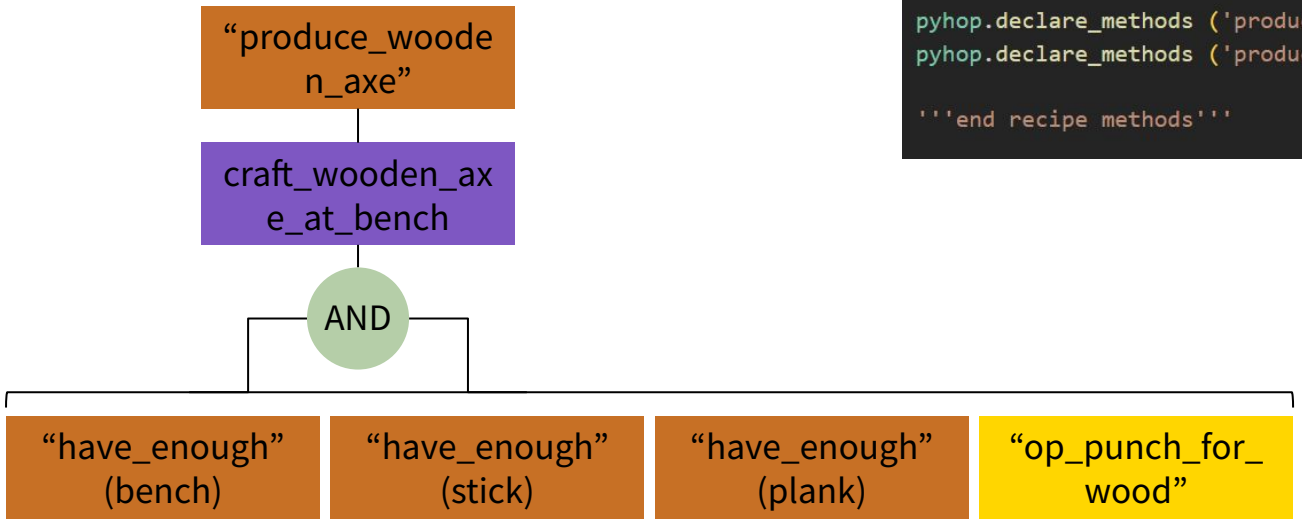
- have_enough [requirement]
- have_enough [requirement]
- …
- op_that_item

Declare **each method** in Pyhop

```
'''begin recipe methods'''

def punch_for_wood (state, ID):
    return [('op_punch_for_wood', ID)]

def craft_wooden_axe_at_bench (state, ID):
    return [('have_enough', ID, 'bench', 1), ('have_enough', ID, 'stick',

# your code here

pyhop.declare_methods ('produce_wood', punch_for_wood)
pyhop.declare_methods ('produce_wooden_axe', craft_wooden_axe_at_bench)

'''end recipe methods'''
```

Manual HTN

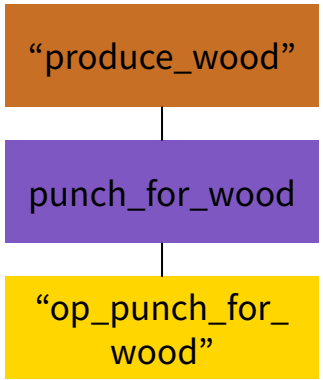
Methods




```
'''begin recipe methods'''

def punch_for_wood (state, ID):
    return [('op_punch_for_wood', ID)]

def craft_wooden_axe_at_bench (state, ID):
    return [('have_enough', ID, 'bench', 1), ('have_enough', ID, 'stick',

# your code here

pyhop.declare_methods ('produce_wood', punch_for_wood)
pyhop.declare_methods ('produce_wooden_axe', craft_wooden_axe_at_bench)

'''end recipe methods'''
```

Manual HTN

Methods




```
'''begin recipe methods'''

def punch_for_wood (state, ID):
    return [('op_punch_for_wood', ID)]

def craft_wooden_axe_at_bench (state, ID):
    return [('have_enough', ID, 'bench', 1), ('have_enough', ID, 'stick',

# your code here

pyhop.declare_methods ('produce_wood', punch_for_wood)
pyhop.declare_methods ('produce_wooden_axe', craft_wooden_axe_at_bench)

'''end recipe methods'''
```

Manual HTN

Solve and submit a solution for the following task:

Given {}, achieve {“wood”: 12} [time <= 46]

1. Define any operators for any item think you need (look at recipes in crafting.json, think about which one you need), declare all of them
2. For each item, declare a method and define a function, map to “have_enough” checks and the operator you defined
3. Add the item inside the “produce” function
4. Update the goal and initial state and run the program

Auto HTN

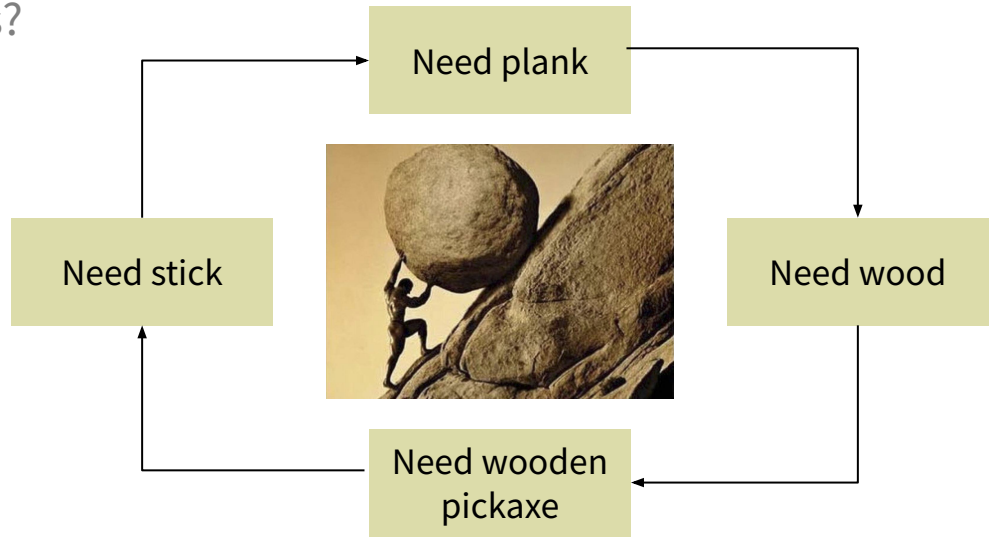
Automatically read crafting.jason. For each rule:

- Declare operators
  - Create operator function in make_operator
  - Declare the operator
- Declare methods
  - Create method function in make_method
  - Declare method as “produce_???”
- Set initial state and goal in set_up_state and set_up_goals
- Heuristic

Auto HTN

Heuristic:

- How to fix this?



How to eliminate unwanted branches for faster execution and preventing infinite loops?