Impasses and Substates

[20 min]

Soar Tutorial

July 2016

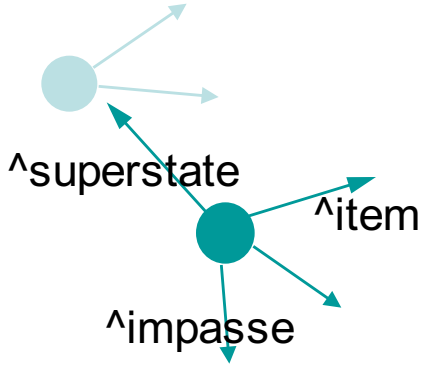
Impasses

What if insufficient or conflicting knowledge for selecting an operator? Called an impasse in Soar

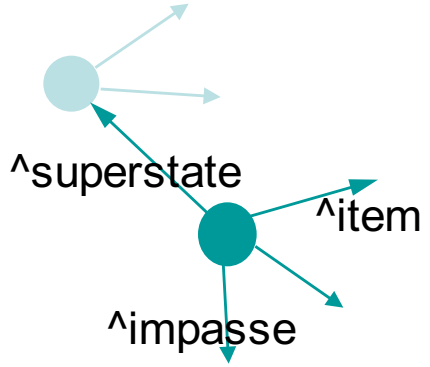
- when no operator is proposed.
  - **[state no-change]**
- when multiple operators are proposed by insufficient preferences to select between them
  - **[tie]**
- when an operator is selected, but it can't be applied by a single rule
  - **[operator no-change]**

Substates

- Substate is created if there is an impasse
- Substate has structures that define impasse
  - **`^superstate`**
  - **`^impasse - no-change, tie, conflict, …`**
  - **`^item - tied or conflicted operators`**
  - …
- Substate is context for deliberate reasoning and accessing additional knowledge sources
  - Long-term memories
  - External environment
  - Internal reasoning
- Select and apply operators in substate
  - Access superstate information through
  - **`(<s> ^superstate <ss>)`**

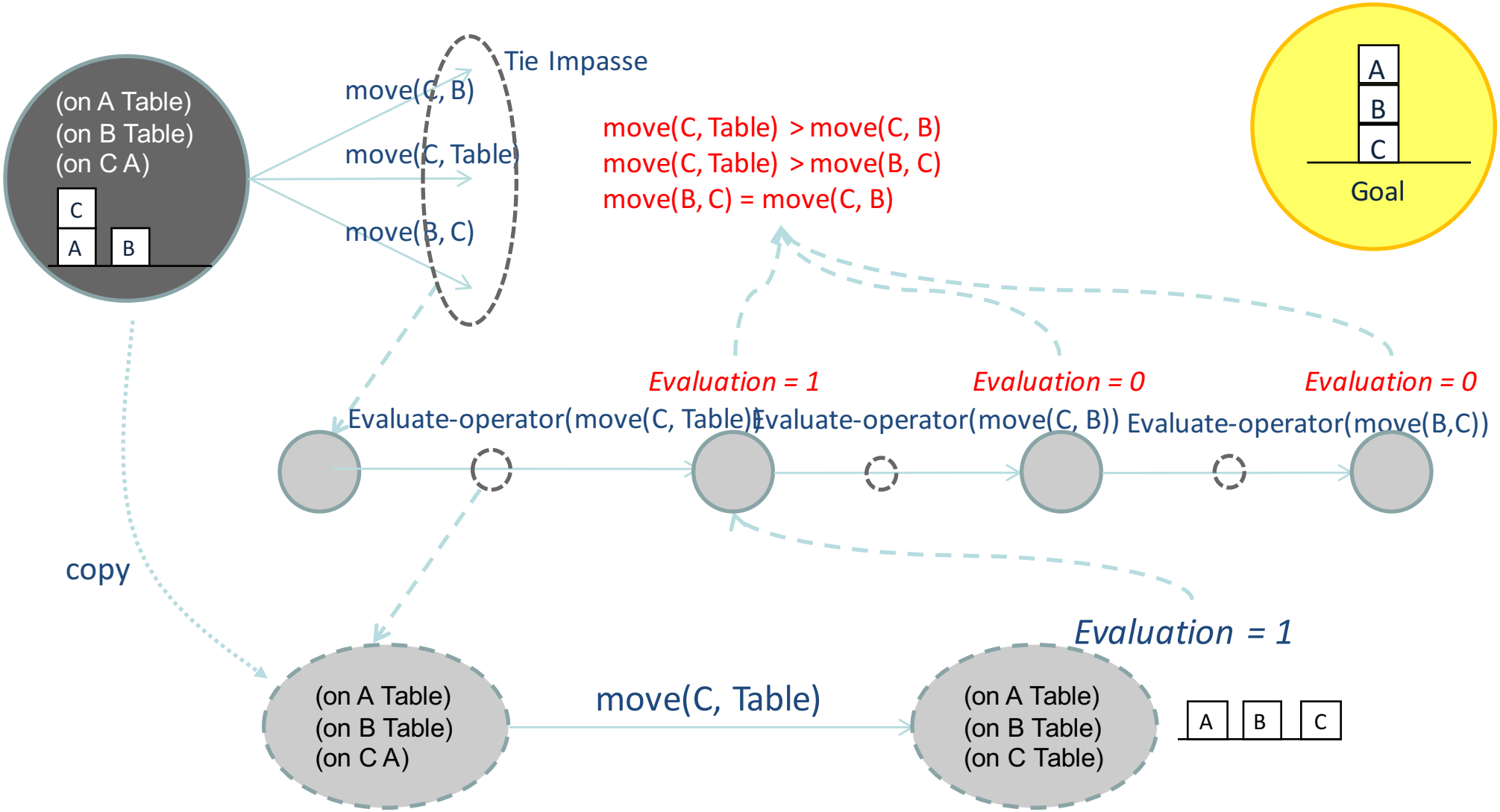


Substates



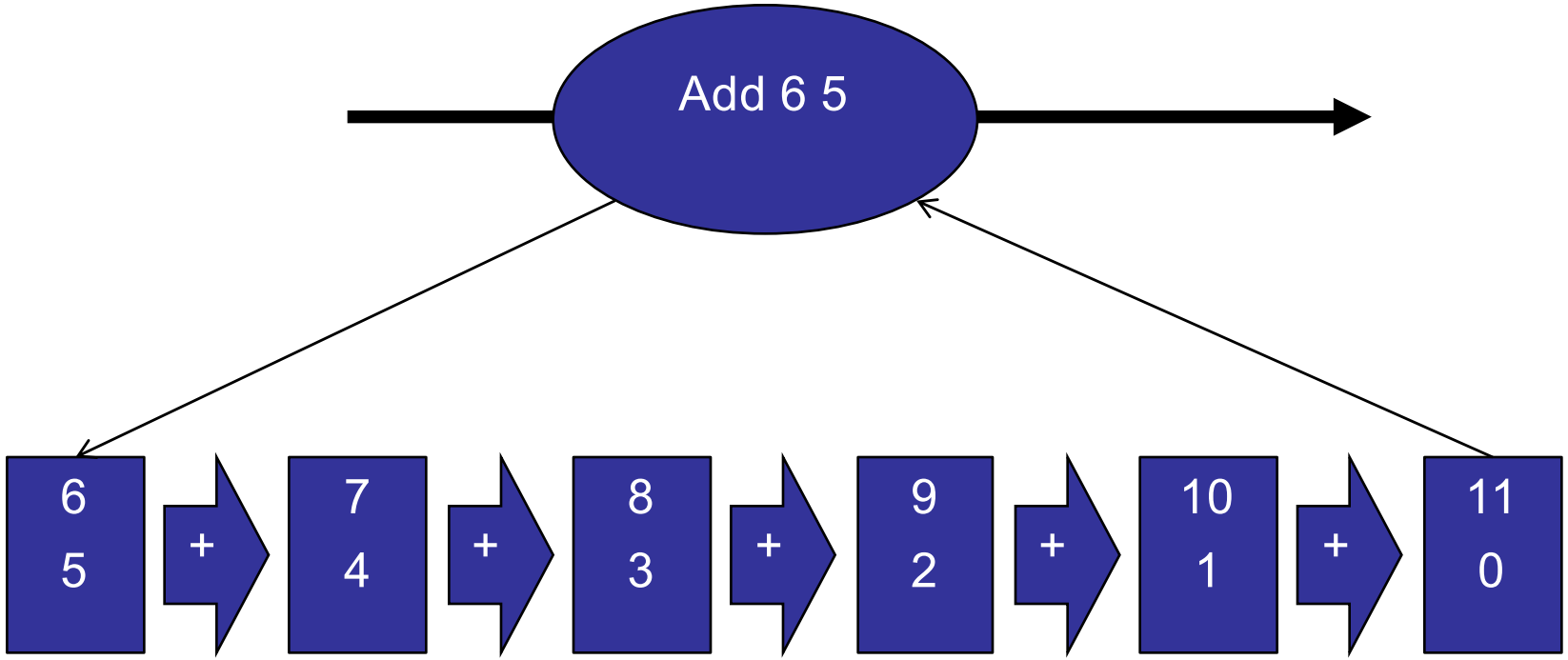
- Substates are removed when the impasse is resolved
  - A new operator is selected.
- Results are structures created in substate but linked to the superstate
- Can have recursive stack of impases/substates

Overview One-step Look-ahead Using Selection Problem Space

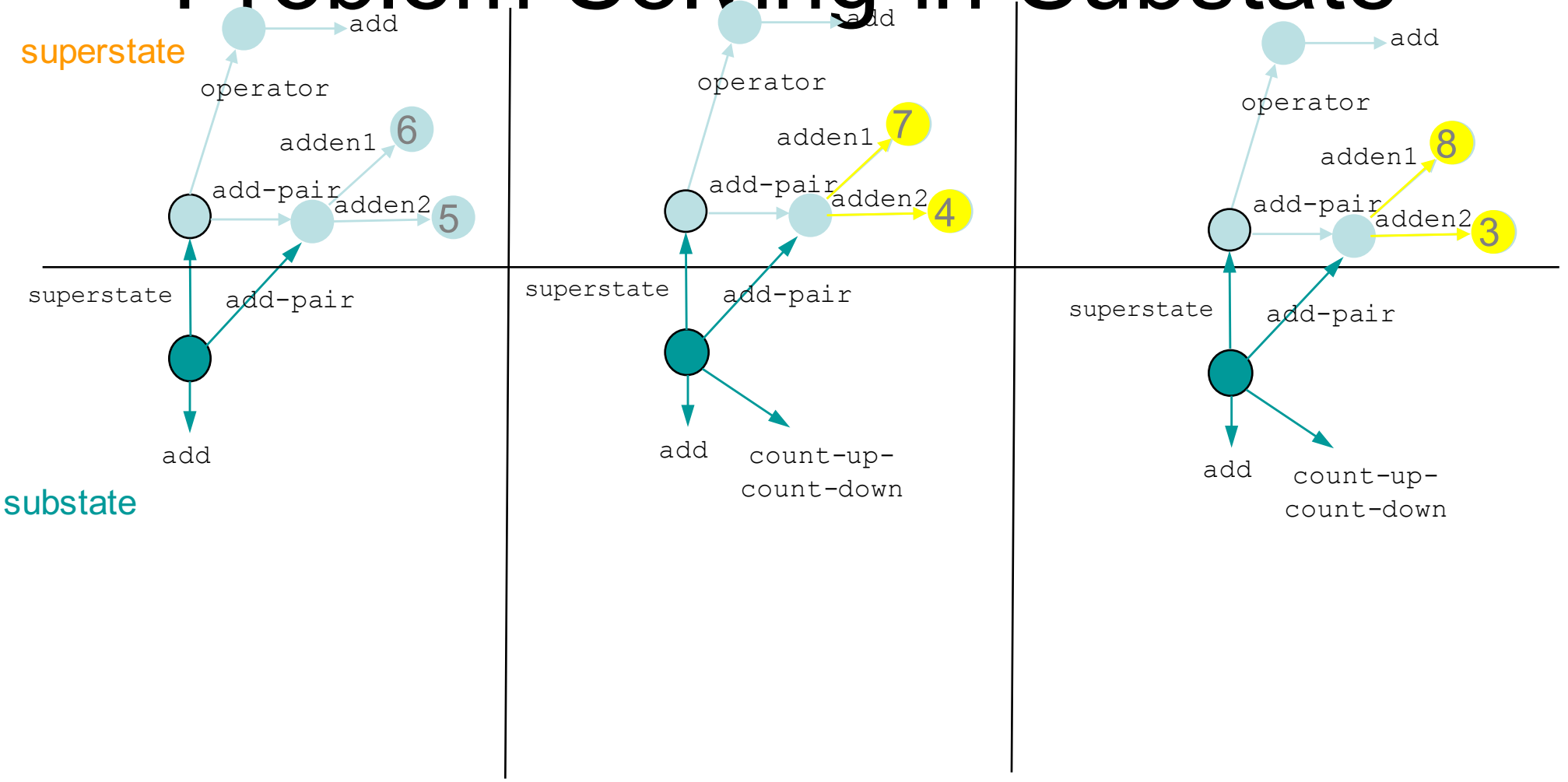


Operator Implementation

- Add two numbers by counting up and down.



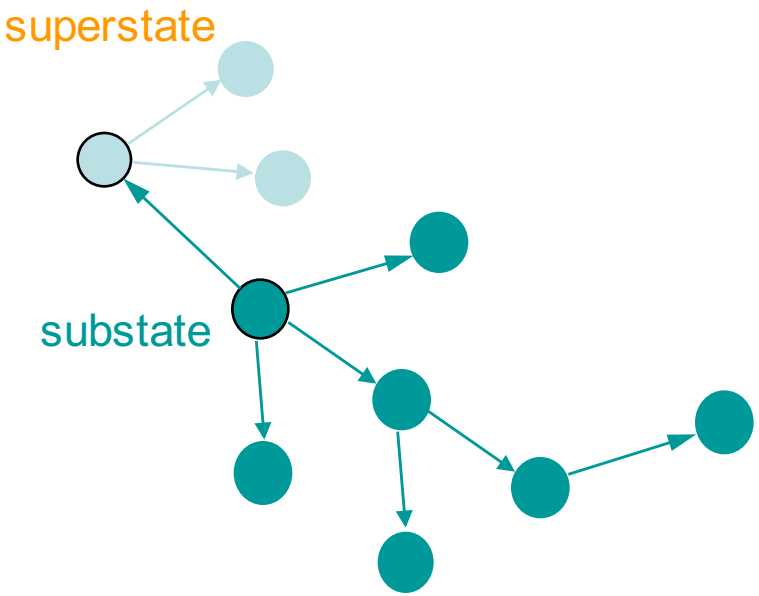
Problem Solving in Substate

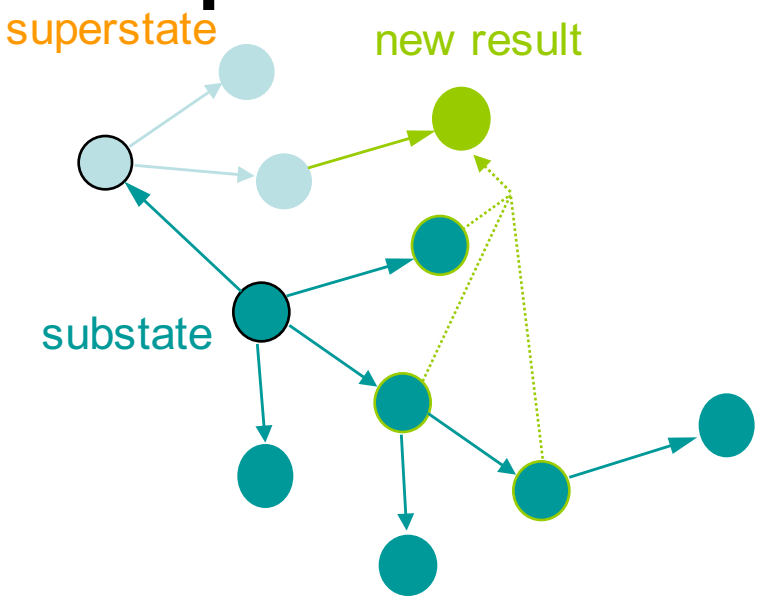


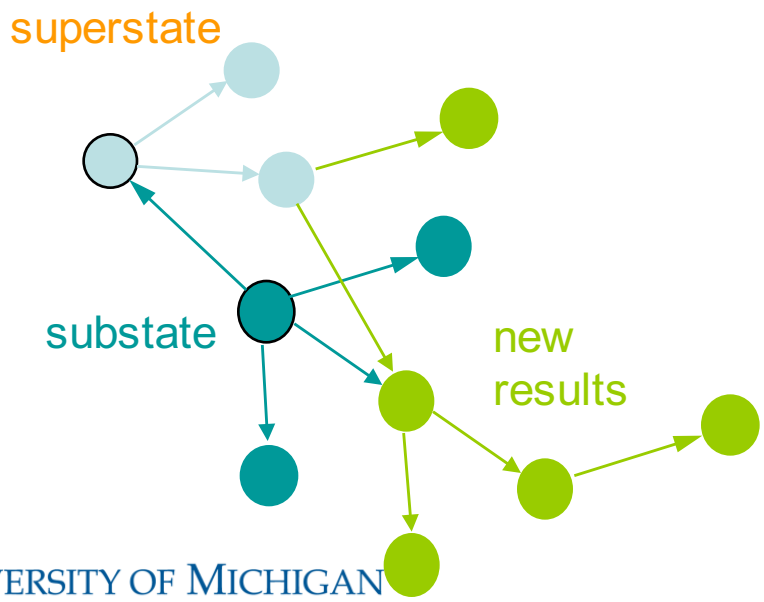
Results

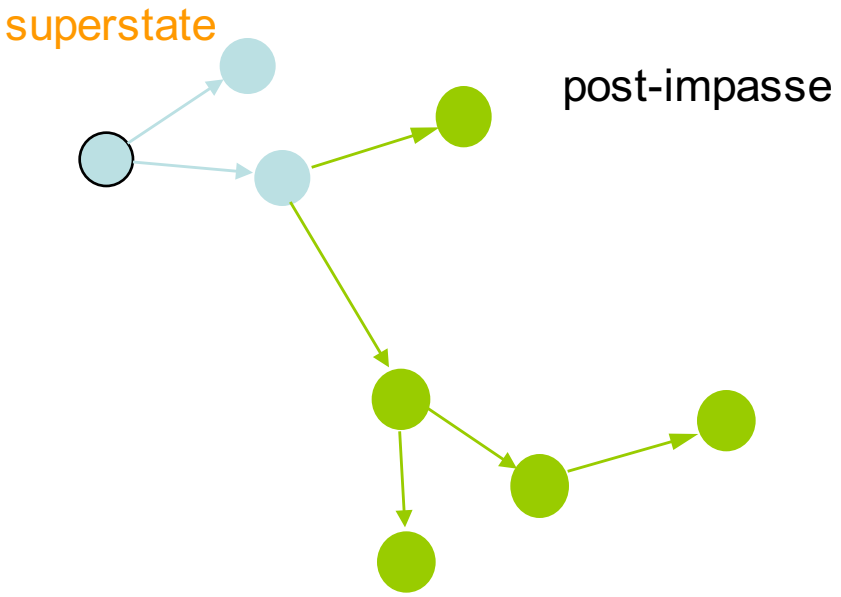
- No explicit marking of results.
- Structures created by rules that match structures in the substate, but create structures linked to a superstate.
- Side effect of working memory structure.

Result Examples

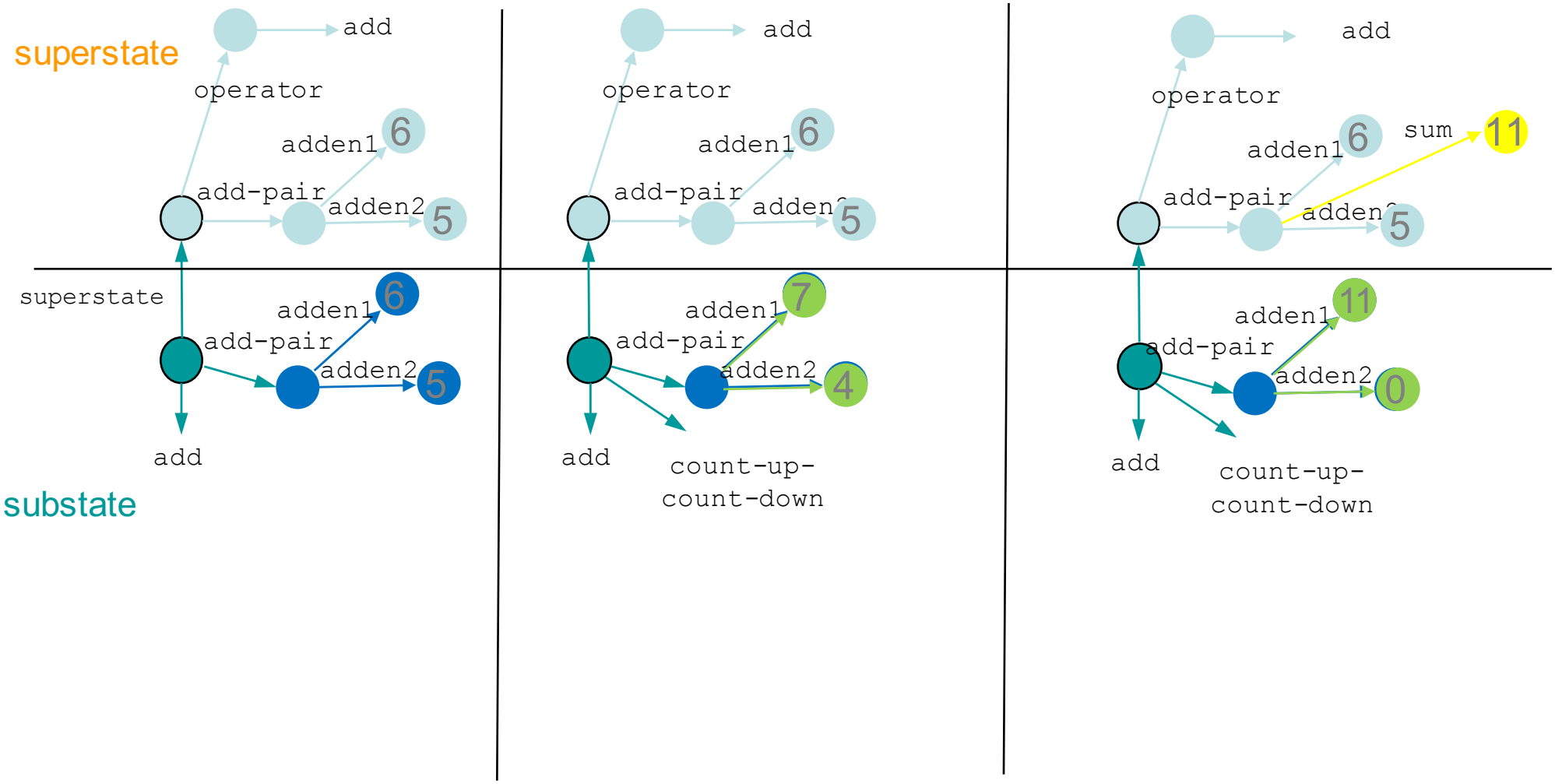








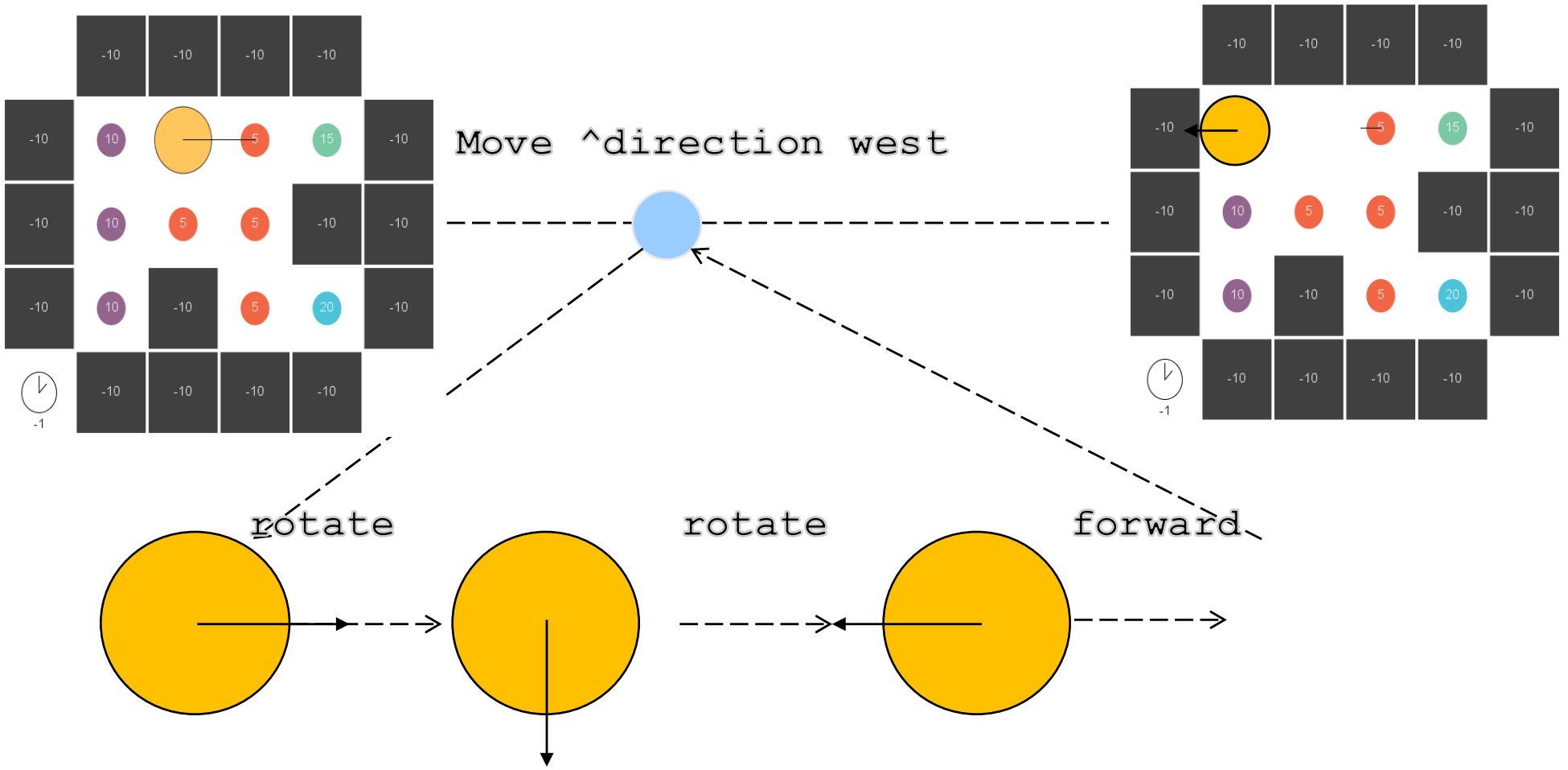
Problem Solving in Substate



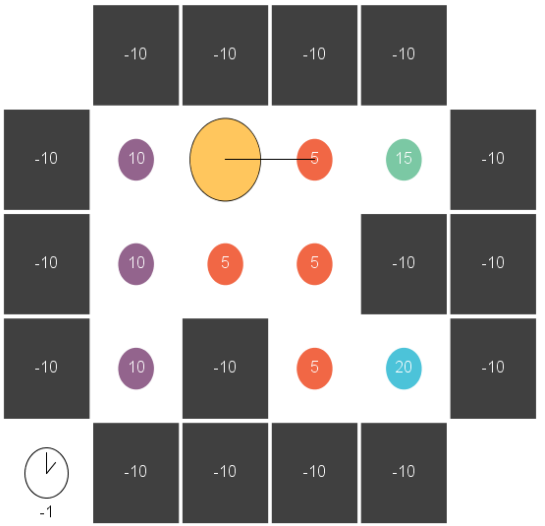
Operator Implementation

- Add operator to move in a cardinal direction:

```
(<o> ^name move ^direction << north south east west >> )
```



Move Operator

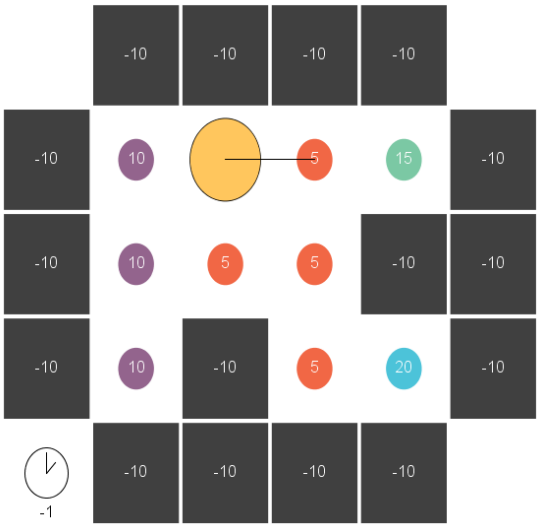


- Need only a proposal.
- Apply will be by operators in substate.

```
If name eater and
   in direction <dir> there is a non-wall
then
   propose move in direction <dir>
```

Cardinal directions don't "blink" during rotate, only during forward

Propose move



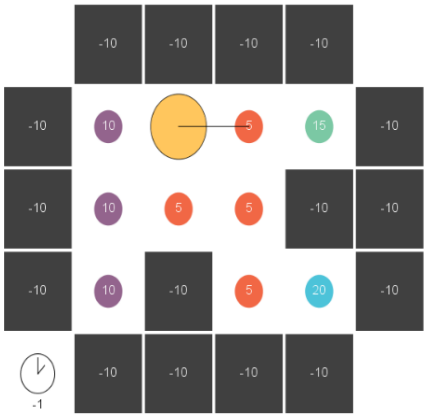

```
If name eater and
    in direction <dir> there is a non-wall
then
    propose move in direction <dir>

sp {eater*propose*move
   (state <s> ^name eater
              ^io.input-link <input>)
   (<input> ^{ << west east north south >> <dir> } <> wall)
-->
   (<s> ^operator <op> + =)
   (<op> ^name move
         ^direction <dir>)}
```

Substate Structure

```
(s1 ^superstate nil
    ^type state
    ...)
(s9 ^attribute operator
    ^choices none
    ^impasse no-change
    ^superstate s1
    ^type state
    ^smem …
    ...)
```

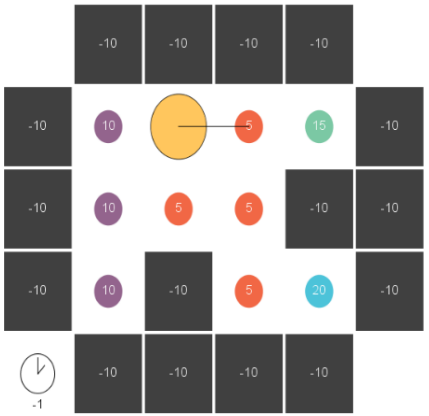
No ^io structure in substate

Rotate in Substate



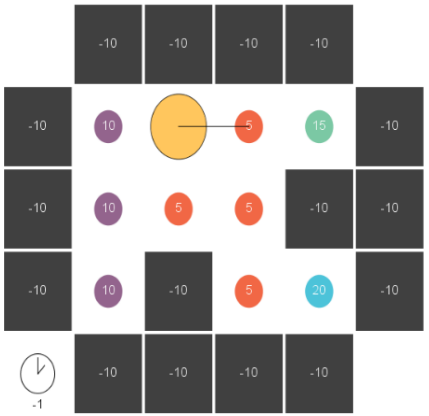

```
# If not facing direction of move (in superstate), propose rotate.
# When rotate, will retract this proposal (orientation blinks)
# Move operator will not retract (directions don't blink on rotate)

sp {move*propose*rotate
   (state <s> ^superstate <ss>)
   (<ss> ^operator <o>
         ^io.input-link.orientation <dir>)
   (<o> ^name move
        ^direction <> <dir>)
-->
   (<s> ^operator <op> + =)
   (<op> ^name rotate)}


sp {apply*rotate
   (state <s> ^operator.name rotate
              ^superstate.io.output-link <out>)
 -->
   (<out> ^rotate <r>)}
```

Forward in Substate




```
# If facing direction of move (in superstate), propose forward.

sp {move*propose*forward
   (state <s> ^superstate <ss>)
   (<ss> ^operator <o>
         ^io.input-link.orientation <dir>)
   (<o> ^name move
        ^direction <dir>)
-->
   (<s> ^operator <op> + =)
   (<op> ^name rotate)}


sp {apply*forward
   (state <s> ^operator.name forward
              ^superstate.io.output-link <out>)
-->
   (<out> ^forward <r>)}
```

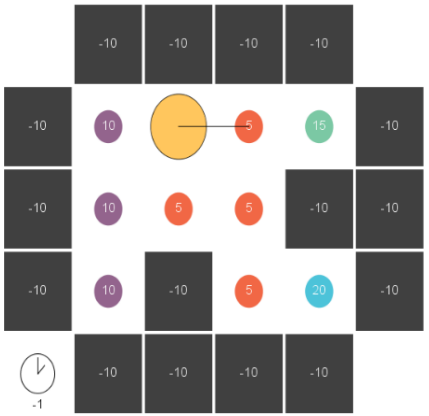
Adding Selection Knowledge



Select operators based on value of food they will consume.

1. Maintain in working memory information on what each color is worth.
2. Propose operators with value they will get
3. Use that values to select operators.

Values in working memory




```
sp {hierarchical*elaborate*map-object*reward
   (state <s> ^name eater)
-->
   (<s> ^color-values <r>)
   (<r> ^wall -10
        ^empty 0
        ^red 5
        ^purple 10
        ^green 15
        ^blue 20)}
```

Numeric Indifferent Rule

```
sp {eater*propose*move
   (state <s> ^name eater
              ^io.input-link <input>)
   (<input> ^{ << west east north south >> <dir> }
                                      { <> wall <color>)
-->
   (<s> ^operator <o> + =)
   (<o> ^name move
        ^direction <dir>
        ^color <color>)}

sp {eater*select*move*operator*indifferent
   (state <s> ^operator <o> +
              ^color-values.<color> <value>)
   (<o> ^name move
        ^color <color>)
-->
   (<s> ^operator <o> = <value>)}
```

Comparison Rule

```
sp {eater*select*move*operator
   (state <s> ^name eater
              ^operator <o1> +
              ^operator { <> <o1> <o2> } +
              ^color-values <cv>)
(<cv> ^<color1> <value1>
      ^<color2> < <value1>)
(<o1> ^color <color1>)
(<o2> ^color <color2>)
-->
(<s> ^operator <o1> > <o2>)}
```

Hierarchical RL

- Use RL for move operators

```
sp {eater*propose*move
   (state <s> ^name eater
              ^io.input-link <input>)
   (<input> ^{ << west east north south >> <dir>}
                  { <> wall <color>})
-->
   (<s> ^operator <o> +)
   (<o> ^name move
        ^direction <dir>
        ^color <color>)}

gp {eater*select*move
   (state <s> ^name eater
              ^operator <o> +)
   (<o> ^name move
        ^color [ purple red blue green empty ] )
-->
   (<s> ^operator <o> = 0)}
```

Hierarchical RL

- Use RL for move operators

```
sp {RL*elaborate*state                  # same as in normal RL
   (state <s> ^name eater
              ^reward-link <rl>
              ^io.input-link.score-diff <d>)
-->
   (<rl> ^reward.value <d>)}
```