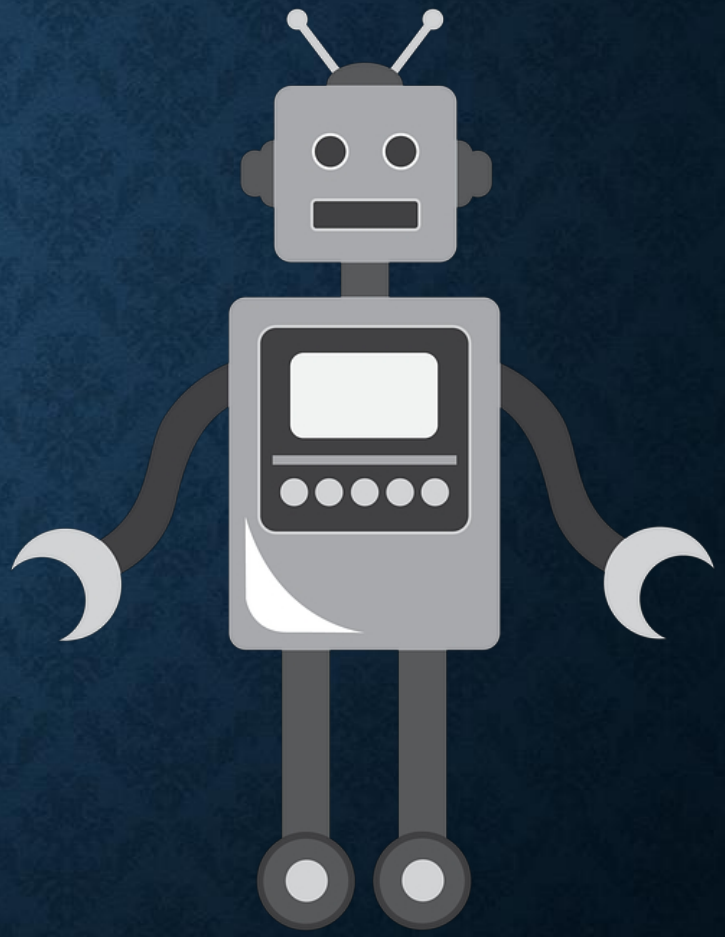
SOAR

ESCAPE ROOM



Presentation of the third part of the Artificial Intelligence Course and Laboratory 2021/2022

University of Turin - Department of Computer Science

Alessandro Saracco - Leonardo Magliolo - Mattia Marra

ROUTE RSO

DESCRIPTION INITIAL

STRUCTURE AND STRATEGIES

RULES AND ACTIONS

FLOWCHART

DEMO

CONCLUSIONS

INITIAL DESCRIPTION

INITIAL DESCRIPTION (THE AGENT)

- Agent Soar is a robot imprisoned in a room where the only way out is A glass window.

- The features and functionality of Soar are as follows:
  - It is 150 cm high
  - It has two pockets in which to store collected items
  - It is capable :
    - See
    - Walk
    - Collect and combine different objects in the room
  - Non è in grado di:
    - Break down objects after he has combined them

- Soar will have to use what he to figure out how to break and Reach the window in order to escape from the room in which he is imprisoned

INITIAL DESCRIPTION (THE ROOM)

- The room is the environment in which Soar is to operate in which he will have the following items:
  - Spring
  - Sprig
  - Pebbles
  - Tire
  - 2 Logs (from a 1 m height each)

- The room will have a wet floor that may Slide the agent while walking

- The window in the room is set **3.5 meters** high and, if hit with a slingshot, likely to be broken on the first attempt

INITIAL DESCRIPTION (THE SETUP)

- Initially, the Soar agent knows only the objects he has available and their possible combinations viz:
  - Slingshot (twig spring)
  - Hammer (twig-stone)
  - Necklace (elastic-stone)
  - Pneumatic-Elastic
  - Pneumatic-Stone
  - Tire-Ramet
- But he doesn't know will be the best combination and strategy to be able to break the window and escape
- In the course of his attempts to interact, he will learn reinforcements that will enable him to make the right (and optimal) decisions in order to escape

INITIAL DESCRIPTION (THE REINFORCEMENTS)

- But what are the reinforcements he will learn?
- Soar, while trying to find the solution, Will learn from the following rewards:

- Rompere la finestra e fuggire (SUCCESSO) → 
- Non riuscire a fuggire (FALLIMENTO) → 

STRUCTURE AND STRATEGIES

HOW TO STRUCTURE THE PROBLEM?

- For the implementation of the project we defined a **Soar Production** that initializes the initial space (described earlier) by declaring all the elements, present in the room, with which Soar will interact

- Subsequently, they were defined:
  - The SPs that define the agent's actions (***setup-rules.soar***)
  - SPs defining RL-rules and rewards. Of the agent (***rl-reward_rules.soar***)

HOW TO STRUCTURE SUCCESS AND FAILURE?

- It is important to note that the agent has 4 cases of success, described later in the section: "*flowchart*"

- While the agent fallirà when he will create a different combo from "*Slingshot*" as it would not allow him to break the window

WHAT STRATEGIES?

- For execution of the project, the following were adopted
  The following Soar strategies:

- **Strategia di Esplorazione ➡ epsilon-greedy**

- Set with the command: "*decide indifferent-selection*," this policy is particularly suitable for projects that make use of RL- rules and is one of the strategies, for action selection, most used .
- In fact, the default strategy set, i.e., *Softmax* not prove so effective in this project.

WHAT STRATEGIES?

- The following were adopted for execution of the project.
  following Soar strategies:

- **Strategia di Learning ➡ *sarsa***

- Set with the command: "*decide rl -s learning-policy*"
- Sarsa, being a variation of Q-Learning, proved to be better than latter, in that it is an on-policy, i.e., the value of the function is updated using a correction factor provided by selected actions with the current policy, i.e., the same one used for estimating value functions.
- Q-Learning, on the other hand, uses two different policies: one is used to estimate value functions, another is used to control the improvement process. Converges first, with less optimal results
- Sarsa's goal is to have the agent's long-term reward maximized.

RULES AND ACTIONS

```
# si sposta con il pavimento asciutto
sp {escape*propose*cammina*stanza*asciutta
   (state <s> ^name escape
      ^oggetti <ogg_1> <ogg_2>
      ^vicino {<ogg_2> <> <ogg_1>}
      -^vicino <ogg_1>
      -^preso <ogg_1>)
   (<st> ^name pavimento ^stato asciutto)
  -->
   (<s> ^operator <op>)
   (<op> ^name cammina-asciutta
      ^to <ogg_1>
      ^from <ogg_2>)
}

sp {escape*apply*cammina*stanza*asciutta
   (state <s> ^name escape
      ^operator <op>
      ^oggetti <ogg_1> <ogg_2>
      ^vicino {<> <ogg_1> <ogg_2>}
      ^azione-svolta <a>)
   (<op> ^name cammina-asciutta
         ^to <ogg_1>
         ^from <ogg_2>)
   (<ogg_1> ^name <n>)
   (<ogg_2> ^name <n2>)
   (<st> ^name pavimento ^stato asciutto)
  -->
   (<s> ^vicino <ogg_2> - <ogg_1> +)
   (<s> ^azione-svolta <a> - cammina-asciutta
   (write (crlf) (crlf) |Ho camminato verso:
}
```

WALKING WITH A DRY FLOOR

- **Precondizioni**:
  - The object to which you want to move:
    - He is not close to the agent
    - He is not taken by the agent
  - The floor of the room is dry

- **Azioni**:
  - The agent will no longer be close to the current object but will move toward the desired object

```
# si sposta con il pavimento bagnato
sp {escape*propose*cammina*stanza*bagnata
   (state <s> ^name escape
      ^oggetti <ogg_1> <ogg_2>
      ^vicino {<ogg_2> <> <ogg_1>}
      -^vicino <ogg_1>
      -^preso <ogg_1>)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<s> ^operator <op>)
   (<op> ^name cammina-bagnata
      ^to <ogg_1>
      ^from <ogg_2>)
}

sp {escape*apply*cammina*stanza*bagnata
   (state <s> ^name escape
      ^operator <op>
      ^oggetti <ogg_1> <ogg_2>
      ^vicino {<> <ogg_1> <ogg_2>}
      ^azione-svolta <a>)
   (<op> ^name cammina-bagnata
         ^to <ogg_1>
         ^from <ogg_2>)
   (<ogg_1> ^name <n>)
   (<ogg_2> ^name <n2>)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<s> ^vicino <ogg_2> - <ogg_1> +)
   (<s> ^azione-svolta <a> - cammina-bagnata)
   (write (crlf) (crlf) |Sono scivolato mentre camminavo ve
}
```

WALKING ON A WET FLOOR

- **Precondizioni**:
  - The object to which you want to move:
    - He is not close to the agent
    - He is not taken by the agent
  - The floor of the room is wet

- **Azioni**:
  - The agent will no longer be close to the current object but will move toward the desired object

- **N.B.**: When the agent executes this rule it will simulate that it has slipped. This rule, over iterations, will be discarded by the agent because it has a lower reward than the previous one

```
# asciuga il pavimento della stanza
sp {escape*propose*asciuga
   (state <s> ^name escape
      ^stanza <st>)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<s> ^operator <op>)
   (<op> ^name asciuga)
}

sp {escape*apply*asciuga
   (state <s> ^operator <op>
     ^azione-svolta <a>
     ^stanza <st>)
   (<op> ^name asciuga)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<st> ^stato bagnato - asciutto +)
   (<s> ^azione-svolta <a> - asciuga)
   (write (crlf) (crlf) |HO ASCIUGATO IL PAVIMENTO!
}
```

DRY THE FLOOR OF THE ROOM

- **Precondizioni**:
  - The floor of the room is wet

- **Azioni**:
  - The agent dries the floor and the state of the floor changes from wet dry

```
sp {escape*propose*prendi
   (state <s> ^name escape
      ^vicino <ogg>
      ^oggetti <ogg>
      ^tasche <z>
      -^preso <ogg>)
   (<z> ^tiene nil)
   -(<ogg> ^name finestra)
  -->
   (<s> ^operator <op>)
   (<op> ^name prendi
      ^oggetti <ogg>
      ^tasche <z>)
}

sp {escape*apply*prendi
   (state <s> ^operator <op>
      ^vicino <ogg>
      -^preso <ogg>
      ^azione-svolta <a>)
   (<op> ^name prendi
      ^oggetti <ogg>
      ^tasche <z>)
   (<z> ^tiene nil)
   (<ogg> ^name <n>)
  -->
   (<s> ^preso <ogg>)
   (<z> ^tiene nil - <ogg>)
   (<s> ^azione-svolta <a> - prendi)
   (write (crlf) (crlf) |Ho preso e messo in
}
```

PICKING UP AN OBJECT AND PUTTING IT IN YOUR POCKET

- **Precondizioni**:
  - The object to which you want to move:
    - He is close to the officer
    - It is not yet taken by the agent
    - One of the two pockets is empty
    - It is not the window (for obvious reasons)

- **Azioni**:
  - The agent will have taken the object
  - One of the two pockets will contain the item taken

```
gp {escape*propose*togli
   (state <s> ^name escape
      ^preso <ogg>
      ^tasche (<z> ^tiene <ogg>))
   (<ogg> ^name [elastico rametto pietra tronco pneumatico])
  -->
   (<s> ^operator <op>)
   (<op> ^name togli
      ^oggetti <ogg>
      ^tasche <z>)
}

sp {escape*apply*togli
   (state <s> ^name escape
      ^azione-svolta <a>
      ^operator <op>)
   (<op> ^name togli
      ^oggetti (<ogg> ^name <n>)
      ^tasche <z>)
  -->
   (<z> ^tiene <ogg> - nil)
   (<s> ^preso <ogg> -)
   (<s> ^azione-svolta <a> - togli)
   (write (crlf) (crlf) |Ho tolto dalla tasca e posato a ter
}
```

REMOVE AN ITEM FROM THE POCKET AND PLACE IT ON THE GROUND

- **Precondizioni**:
  - One of the two agent pockets contains the object
  - The only object he cannot remove is the window (for obvious reasons)

- **Azioni**:
  - The agent will no longer have poossession of the item previously taken
  - One of the two pockets does not will contain more of the object

```
sp {escape*propose*combinazione
   (state <s> ^name escape
      ^tasche <t1> <t2>
      ^combinazione <c>)
   (<t1> ^tiene <ogg_1> ^is tasca_sx)
   (<t2> ^tiene <ogg_2> ^is tasca_dx)
   (<c> ^ingredienti <ogg_1> {<ogg_2> <> <ogg_1
   (<finestra> ^name finestra ^rotta no)
  -->
   (<s> ^operator <op>)
   (<op> ^name combinazione
      ^oggetti <c>
      ^ingredienti <ogg_1> <ogg_2>)
}

sp {escape*apply*combinazione
   (state <s> ^operator <op>
      ^tasche <t1> <t2>
      ^azione-svolta <a>)
   (<op> ^name combinazione
      ^oggetti <c>
      ^ingredienti <ogg_1> <ogg_2>)
   (<c> ^name <n>)
   (<t1> ^tiene <ogg_1> ^is tasca_sx)
   (<t2> ^tiene <ogg_2> ^is tasca_dx)
   (<finestra> ^name finestra ^rotta no)
  -->
   (<s> ^oggetti <c> ^preso <c>)
   (<t1> ^tiene <ogg_1> - <c>)
   (<t2> ^tiene <ogg_2> - nil)
   (<s> ^azione-svolta <a> - combinazione)
   (write (crlf) (crlf) |SONO RIUSCITO A COSTRU
}
```

COMBINE TWO OBJECTS

- **Precondizioni**:
  - The agent's two pockets are not empty
    - The left pocket contains the first item to be combined
    - The right pocket contains the second item to be combined
  - The window is not broken
  -
- **Azioni**:
  - The agent creates the combination based on the combo
  - possible with the two items it possesses
  - You free up one of the two pockets
  - The other pocket is occupied by the combination created

    One of the two pockets will contain the item taken

```
sp {escape*propose*uso*fionda-lontano
   (state <s> ^name escape
      ^tasche <t1> <t2>
      ^oggetti <finestra>
      ^altezza <> 3.5)
   (<finestra> ^name finestra ^rotta no)
   (<t1> ^tiene (<c> ^name fionda))
   (<t2> ^tiene.name pietra)
  -->
   (<s> ^operator <op>)
   (<op> ^name uso
      ^oggetti <c>)
}

sp {escape*apply*uso*fionda-lontano
   (state <s> ^operator <op>
      ^oggetti <finestra>
      ^azione-svolta <a>
      ^tasche (<z> ^tiene <ogg>)
      ^altezza <> 3.5)
   (<ogg> ^name pietra)
   (<op> ^name uso ^oggetti <c>)
   (<finestra> ^name finestra ^rotta no)
   (<c> ^name fionda)
  -->
   (<finestra> ^rotta no -)
   (<finestra> ^rotta (ifeq (rand-int 5) 1 si no))
   (<z> ^tiene <ogg> - nil)
   (<s> ^preso <ogg> -)
   (<s> ^azione-svolta <a> - uso-fionda-lontano)
   (write (crlf) (crlf) |HO PROVATO A COLPIRE UN PUNTO DELLA
}
```

USE THE SLINGSHOT FROM A DISTANCE

- **Precondizioni**:
  - The agent's two pockets are not empty
    - The left pocket contains the combo: "Slingshot"
    - The right pocket contains a stone
  - The window is not broken
  - The agent is far from the window (i.e., has not could still climb the logs)
- **Azioni**:
  - Officer tries to use slingshot to break up the window glass
  - The probability of it breaking is only 20% (because it is far from it).
  - After throwing, the stone is no longer in the pocket of the agent

```
sp {escape*propose*uso*fionda-vicino
   (state <s> ^name escape
      ^tasche <t1> <t2>
      ^oggetti <finestra>
      ^altezza [<ht> 3.5]
      ^vicino <finestra>)
   (<finestra> ^name finestra ^rotta no)
   (<t1> ^tiene (<c> ^name fionda))
   (<t2> ^tiene.name pietra)
  -->
   (<s> ^operator <op>)
   (<op> ^name uso
      ^oggetti <c>)
}

sp {escape*apply*uso*fionda-vicino
   (state <s> ^operator <op>
      ^oggetti <finestra>
      ^azione-svolta <a>
      ^tasche (<z> ^tiene <ogg>)
      ^altezza [<ht> 3.5]
      ^vicino <finestra>)
   (<ogg> ^name pietra)
   (<op> ^name uso ^oggetti <c>)
   (<finestra> ^name finestra ^rotta no)
   (<c> ^name fionda)
  -->
   (<finestra> ^rotta no -)
   (<finestra> ^rotta (ifeq (rand-int 3) 1 si no))
   (<z> ^tiene <ogg> - nil)
   (<s> ^preso <ogg> -)
   (<s> ^azione-svolta <a> - uso-fionda-vicino)
   (write (crlf) (crlf) |HO PROVATO A COLPIRE IL PUNTO DEBOLE DEL
}
```

USE THE SLINGSHOT AT CLOSE RANGE

- **Precondizioni**
  - :
    - The agent's two pockets are not empty
      - The left pocket contains the combo: "Slingshot"
      - The right pocket contains a stone
    - The window is not broken
    - The officer is near the window (i.e., he was able to climb the logs)
- **Azioni**
  - :
    - The officer tries to use the slingshot to break the window glass
    - The probability of it breaking down is 33% (because it is close to it).

```
sp {escape*propose*uso*tronco*stanza*asciutta
   (state <s> ^name escape
      ^tasche.tiene (<t> ^name tronco)
      ^oggetti <t>
      ^vicino <finestra>)
   (<finestra> ^name finestra)
   (<st> ^name pavimento ^stato asciutto)
  -->
   (<s> ^operator <op>)
   (<op> ^name uso
      ^oggetti <t>)
}

sp {escape*apply*uso*tronco*stanza*asciutta
   (state <s> ^operator <op>
      ^altezza <aa>
      ^tasche (<z> ^tiene <t>)
      ^azione-svolta <a>
      ^vicino <finestra>)
   (<finestra> ^name finestra)
   (<op> ^name uso
      ^oggetti <t>)
   (<t> ^name tronco ^altezza <at>)
   (<st> ^name pavimento ^stato asciutto)
  -->
   (<s> ^altezza <aa> -)
   (<s> ^altezza (+ <aa> <at>))
   (<z> ^tiene <t> - nil)
   (<s> ^azione-svolta <a> - uso-tronco-asciutta)
   (write (crlf) (crlf) |HO POSIZIONATO UN TRONCO CON IL P
}
```

PLACE A LOG WITH A DRY FLOOR

- **Precondizioni**:
  - The log is in an agent's pocket.
  - The officer stands near the window
  - The floor of the room is dry
  - The desired height has not yet been reached

- **Azioni**:
  - The desired height changes by increasing By 1 meter from the previous value
  - The agent's pocket no longer contains the trunk

```
sp {escape*propose*uso*tronco*stanza*bagnata
   (state <s> ^name escape
      ^tasche.tiene (<t> ^name tronco)
      ^oggetti <t>
      ^vicino <finestra>)
   (<finestra> ^name finestra)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<s> ^operator <op>)
   (<op> ^name uso
      ^oggetti <t>)
}

sp {escape*apply*uso*tronco*stanza*bagnata
   (state <s> ^operator <op>
      ^altezza <aa>
      ^tasche (<z> ^tiene <t>)
      ^azione-svolta <a>
      ^vicino <finestra>)
   (<finestra> ^name finestra)
   (<op> ^name uso
      ^oggetti <t>)
   (<t> ^name tronco ^altezza <at>)
   (<st> ^name pavimento ^stato bagnato)
  -->
   (<s> ^altezza <aa> -)
   (<s> ^altezza (+ <aa> <at>))
   (<z> ^tiene <t> - nil)
   (<s> ^azione-svolta <a> - uso-tronco-bagnata)
   (write (crlf) (crlf) |HO POSIZIONATO UN TRONCO CON IL PA
}
```

PLACE A LOG WITH A WET FLOOR

- :
  - The log is in an agent's pocket.

Precondizioni

  - The officer stands near the window
  - The floor of the room is wet
  - The desired height has not yet been reached
- :
  - The desired height changes by increasing By 1 meter from the previous value
  - The agent's pocket no longer contains the trunk

Azioni

- : When the agent executes this rule, it is simulated that a troco may slip because of the wet floor. This rule, over the course of iterations, will be discarded by the agent because it has a lower reward than the previous one

N.B.

```
sp {escape*propose*escape
   (state <s> ^name escape
              ^oggetti <finestra>
              ^altezza [<ht> 3.5])
   (<finestra> ^name finestra ^rotta si)
  -->
   (<s> ^operator <op>)
   (<op> ^name escape)
}

sp {escape*apply*escape
   (state <s> ^name escape
              ^operator <op>)
   (<op> ^name escape)
  -->
   (<s> ^fuggito si)
}
```

ESCAPE

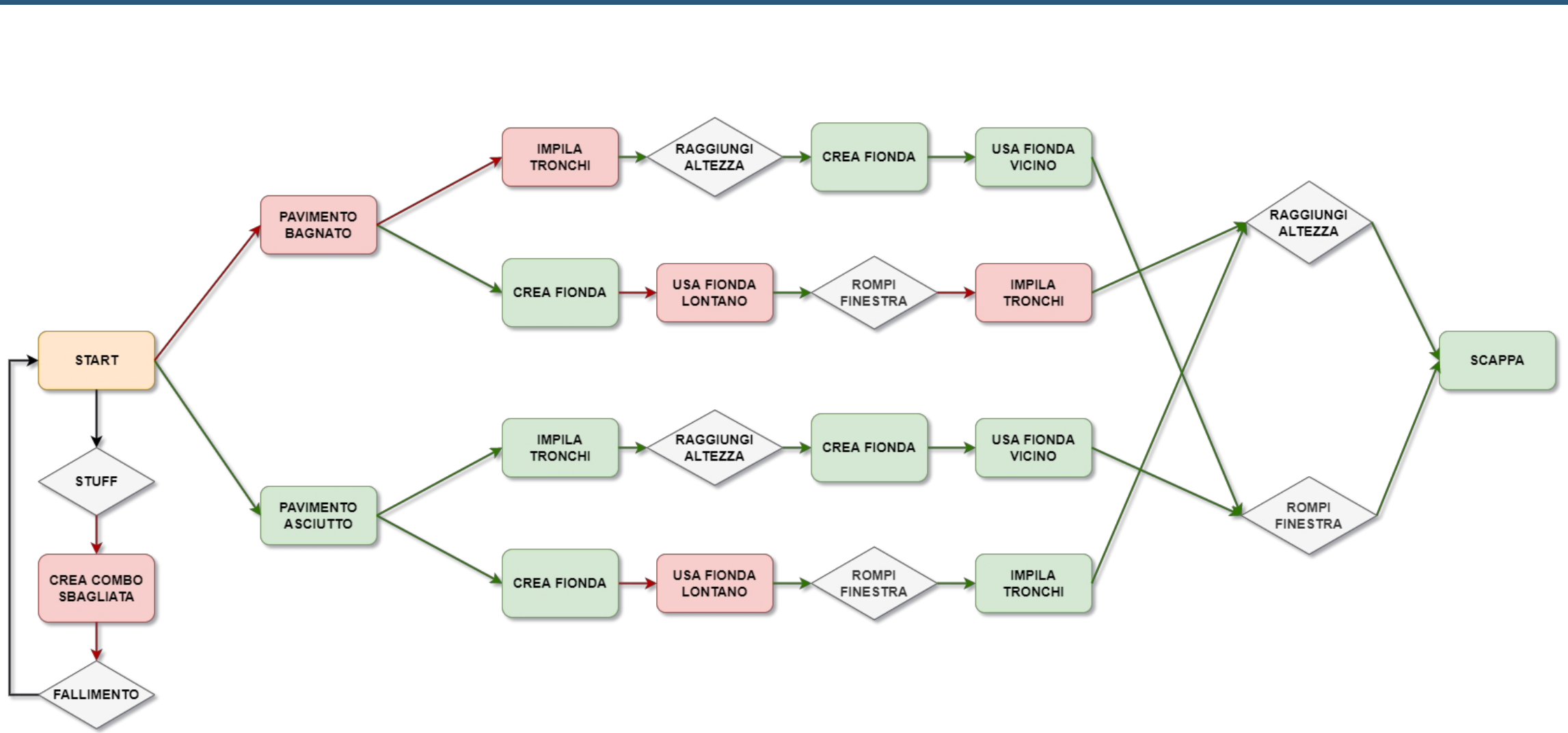
- **Precondizioni**:
  - agent reached the desired height (i.e., stacked the two logs)
  - The window glass is broken
- **Azioni** :
  - The agent escapes from the room

```
sp {escape*elaborate*finito*fallimento
   (state <s> ^name escape
              ^reward-link <r>
              ^oggetti <c> <finestra>
              ^tasche <z>
              ^combinazione <c>)
   (<z> ^tiene <c>)
   (<c> ^name <n> <> fionda)
   (<finestra> ^name finestra ^rotta no)
   (<r> ^reward <rr>)
  -->
   (write (crlf) (crlf) |HO FALLITO, CON: | <n> | NON POSSO ROMPERE LA FINESTRA..|)
   (halt)
}
```

FAILURE

- **Precondizioni**:
  - The agent did not create the combo: "Slingshot."
  - The window glass is not broken
- **Azioni** :
  - Officer fails to escape from room

FLOWCHART


START
STUFF
CREA COMBO SBAGLIATA
FALLIMENTO
PAVIMENTO BAGNATO
PAVIMENTO ASCIUTTO
IMPILA TRONCHI
RAGGIUNGI ALTEZZA
CREA FIONDA
USA FIONDA VICINO
CREA FIONDA
USA FIONDA LONTANO
ROMPI FINESTRA
IMPILA TRONCHI
IMPILA TRONCHI
RAGGIUNGI ALTEZZA
CREA FIONDA
USA FIONDA VICINO
CREA FIONDA
USA FIONDA LONTANO
ROMPI FINESTRA
IMPILA TRONCHI
RAGGIUNGI ALTEZZA
ROMPI FINESTRA
SCAPPA

DEMO

CONCLUSIONS

CONSLUSIONS

Test su 10.000 runs

SP	Sarsa	
	N di firing	Valore
escape	6439	3.84
cammina*stanza*bagnata	275	-2.39
cammina*stanza*asciutta	9725	-1.77
uso-tronco-bagnata	200	-0.30
uso-tronco-asciutta	6196	1.82
uso-fionda-lontano	22304	-0.12
uso-fionda-vicino	11228	2.03
asciuga	10000	-0.19
combinazione*pneumatico-pietra	727	-4
combinazione*pneumatico-rametto	543	-4
combinazione*pneumatico-elastico	893	-4
combinazione*collana	834	-4
combinazione*martello	564	-4
combinazione*fionda	6439	-1.17