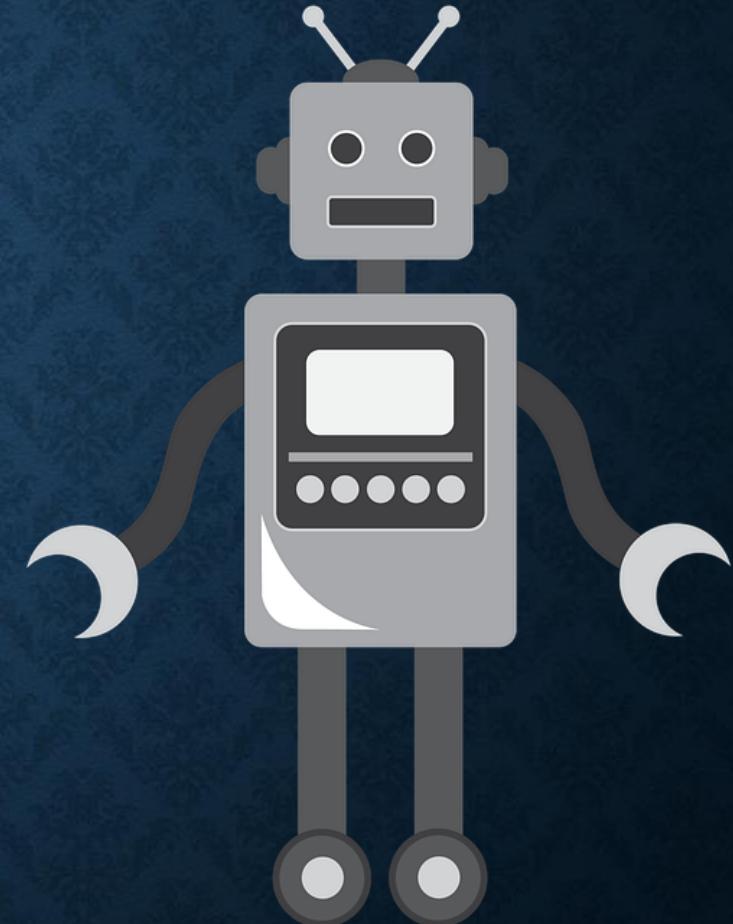


SOAR ESCAPE ROOM

Presentazione della terza parte del corso di Intelligenza Artificiale e
Laboratorio 2021/2022

Università degli Studi di Torino – Dipartimento di Informatica

Alessandro Saracco - Leonardo Magliolo - Mattia Marra



PERCORSO



DESCRIZIONE
INIZIALE



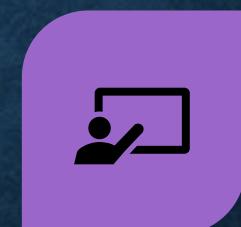
STRUTTURA E
STRATEGIE



REGOLE E AZIONI



FLOWCHART



DEMO



CONCLUSIONI

DESCRIZIONE INIZIALE



DESCRIZIONE INIZIALE (L'AGENTE)



- L'agente Soar è un robot prigioniero in una stanza dove l'unica via d'uscita è una finestra in vetro.
- Le caratteristiche e funzionalità di Soar sono le seguenti:
 - È alto 150 cm
 - È dotato di due tasche nelle quali conservare gli oggetti raccolti
 - È in grado di:
 - Vedere
 - Camminare
 - Raccogliere e combinare diversi oggetti presenti nella stanza
 - Non è in grado di:
 - scomporre gli oggetti dopo che li ha combinati
- Soar dovrà sfruttare ciò che ha a disposizione per capire come rompere e raggiungere la finestra al fine di fuggire dalla stanza in cui è prigioniero

DESCRIZIONE INIZIALE

(LA STANZA)



- La stanza è l'ambiente in cui Soar dovrà operare in cui avrà a disposizione i seguenti oggetti:
 - Molla
 - Rametto
 - Ciottoli
 - Pneumatico
 - 2 Tronchi (da un 1 mt di altezza ciascuno)
- La stanza avrà un pavimento bagnato che potrebbe far scivolare l'agente mentre cammina
- La finestra nella stanza è posta a 3,5 metri di altezza e, se colpita con una fionda, è probabile che venga rotta al primo tentativo

DESCRIZIONE INIZIALE (IL SETUP)



- Inizialmente l'agente Soar conosce solo gli oggetti che ha a disposizione e le loro possibili combinazioni ovvero:
 - Fionda (molla rametto)
 - Martello (rametto-pietra)
 - Collana (elastico-pietra)
 - Pneumatico-Elastico
 - Pneumatico-Pietra
 - Pneumatico-Rametto
- Ma non sa quale potrà essere la migliore combinazione e quale strategia adottare per poter rompere la finestra e fuggire
- Nel corso dei suoi tentativi di interazione imparerà i rinforzi che gli consentiranno di prendere le giuste (ed ottimali) decisioni per poter fuggire

DESCRIZIONE INIZIALE

(I RINFORZI)



- Ma quali sono i rinforzi che apprenderà?
- Soar, mentre cercherà di trovare la soluzione, imparerà dai seguenti rewards:
 - Camminare con il pavimento bagnato → - 0.8
 - Camminare con il pavimento asciutto → - 0.5
 - Asciugare il pavimento → + 1
 - Prendere un oggetto → 0.5
 - Posare un oggetto → - 1
 - Creare una combinazione → + 1 se crea Fionda, - 1 altrimenti
 - Posizionare un tronco con il pavimento bagnato → 0.5
 - Posizionare un tronco con il pavimento asciutto → + 1
 - Usare la fionda (combo) vicino alla finestra → + 1
 - Usare la fionda lontano dalla finestra → 0.5
 - Rompere la finestra e fuggire (SUCESSO) → + 3
 - Non riuscire a fuggire (FALLIMENTO) → - 3

STRUTTURA E STRATEGIE



COME STRUTTURARE IL PROBLEMA?

- Per l'implementazione del progetto abbiamo definito una **Soar Production** che inizializza lo spazio iniziale (prima descritto) dichiarando tutti gli elementi, presenti nella stanza, con cui Soar interagirà
- Successivamente sono state definite:
 - Le SP che definiscono le azioni dell'agente (**setup-rules.soar**)
 - Le SP che definiscono le RL-rules e i rewards dell'agente (**rl-reward_rules.soar**)

COME STRUTTURARE SUCCESSO E FALLIMENTO?

- È importante sottolineare che l'agente ha 4 casi di **successo**, descritti più avanti nella sezione: «*flowchart*»
- Mentre l'agente **fallirà** quando creerà una combo differente da «*Fionda*» in quanto non gli permetterebbe di rompere la finestra

QUALI STRATEGIE?

- Per l'esecuzione del progetto sono state adottate le seguenti strategie di Soar:
- **Strategia di Esplorazione** → **epsilon-greedy**
 - Impostata con il comando: «*decide indifferent-selection*», questa policy è particolarmente adatta per progetti che fanno uso di RL-rules ed è una delle strategie, per la selezione delle azioni, più utilizzate .
 - Infatti la strategia impostata di default, ovvero la *Softmax* non si è rivelata così efficace in questo progetto.

QUALI STRATEGIE?

- Per l'esecuzione del progetto sono state adottate le seguenti stategie di Soar:
 - **Strategia di Learning** ➔ *sarsa*
 - Impostata con il comando: «*decide rl -s learning-policy*»
 - Sarsa, essendo una variazione del Q-Learning, si è rivelata migliore rispetto a quest'ultimo, in quanto è una politica di tipo «on-policy» cioè il valore della funzione viene aggiornato utilizzando un fattore di correzione fornito da azioni selezionate con la policy corrente, cioè la stessa utilizzata per la stima delle funzioni valore.
 - Il Q-Learning invece usa due policies diverse: una viene utilizzata per stimare le funzioni di valore, un'altra viene utilizzata per controllare il processo di miglioramento. Converge prima, con risultati meno ottimali
 - L'obiettivo di Sarsa è quello di far massimizzare la ricompensa a lungo termine dell'agente.

REGOLE E AZIONI



```

# 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:
}

```

CAMMINARE CON IL PAVIMENTO ASCIUTTO

- **Precondizioni:**

- L'oggetto verso cui ci si vuole spostare:
 - Non è vicino all'agente
 - Non è preso dall'agente
 - Il pavimento della stanza è asciutto

- **Azioni:**

- L'agente non sarà più vicino all'oggetto corrente ma si sposterà verso l'oggetto desiderato

```

# 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
}

```

CAMMINARE CON IL PAVIMENTO BAGNATO

- **Precondizioni:**

- L'oggetto verso cui ci si vuole spostare:
 - Non è vicino all'agente
 - Non è preso dall'agente
- Il pavimento della stanza è bagnato

- **Azioni:**

- L'agente non sarà più vicino all'oggetto corrente ma si sposterà verso l'oggetto desiderato

- **N.B.:** quando l'agente eseguirà questa regola simulerà di essere scivolato. Questa regola, nel corso delle iterazioni, verrà scartata dall'agente in quanto ha un reward più basso della precedente

```

# 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!
}

```

ASCIUGARE IL PAVIMENTO DELLA STANZA

- **Precondizioni:**
 - Il pavimento della stanza è bagnato
- **Azioni:**
 - L'agente asciuga il pavimento e lo stato del pavimento cambia da bagnato ad asciutto

```

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
}

```

RACCOGLIERE UN OGGETTO E METTERLO IN TASCA

- **Precondizioni:**

- L'oggetto verso cui si vuole spostare:
 - È vicino all'agente
 - Non è ancora preso dall'agente
 - Una delle due tasche è vuota
 - Non è la finestra (per ovvie ragioni)

- **Azioni:**

- L'agente avrà preso l'oggetto
- Una delle due tasche conterrà l'oggetto preso

```

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
}

```

TOGLIERE UN OGGETTO DALLA TASCA E POSARLO A TERRA

- **Precondizioni:**
 - Una delle due tasche dell'agente contiene l'oggetto
 - L'unico oggetto che non può togliere è la finestra (per ovvie ragioni)
- **Azioni:**
 - L'agente non avrà più possesso dell'oggetto preso in precedenza
 - Una delle due tasche non conterrà più l'oggetto

```

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 COSTRUIRE UNA COMBINAZIONE|)
}

```

COMBINARE DUE OGGETTI

- **Precondizioni:**

- Le due tasche dell'agente non sono vuote
 - La tasca sinistra contiene il primo oggetto da combinare
 - La tasca destra contiene il secondo oggetto da combinare
 - La finestra non è rotta

- **Azioni:**

- L'agente crea la combinazione in base alla **combinazione** possibile con i due oggetti che possiede
- Si libera una delle due tasche
- L'altra tasca viene occupata dalla **combinazione** creata
- Una delle due tasche conterrà l'oggetto preso

```

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 |
}

```

USARE LA FIONDA DA LONTANO

- **Precondizioni:**

- Le due tasche dell'agente non sono vuote
- La tasca sinistra contiene la combo: "Fionda"
- La tasca destra contiene una pietra
- La finestra non è rottta
- L'agente è lontano dalla finestra (cioè, non ha potuto ancora salire sui tronchi)

- **Azioni:**

- L'agente prova ad usare la fionda per rompere il vetro della finestra
- La probabilità che si rompa è solo del 20% (perché è lontano da essa).
- Dopo il lancio, la pietra non è più nella tasca dell'agente

```

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
}

```

USARE LA FIONDA DA VICINO

- **Precondizioni:**

- Le due tasche dell'agente non sono vuote
 - La tasca sinistra contiene la combo: “Fionda”
 - La tasca destra contiene una pietra
- La finestra non è rossa
- L'agente è vicino alla finestra (cioè, ha potuto salire sui tronchi)

- **Azioni:**

- L'agente prova ad usare la fionda per rompere il vetro della finestra
- La probabilità che si rompa è del 33% (perché è vicino ad essa).
- Dopo il lancio, la pietra non è più nella tasca dell'agente

```

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
}

```

POSIZIONARE UN TRONCO CON IL PAVIMENTO ASCIUTTO

- **Precondizioni:**

- Il tronco è in una tasca dell'agente
- L'agente si trova vicino alla finestra
- Il pavimento della stanza è asciutto
- L'altezza desiderata non è ancora stata raggiunta

- **Azioni:**

- L'altezza desiderata cambia aumentando di 1 metro rispetto al valore precedente
- La tasca dell'agente non contiene più il tronco

```

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
}

```

POSIZIONARE UN TRONCO CON IL PAVIMENTO BAGNATO

- **Precondizioni:**

- Il tronco è in una tasca dell'agente
- L'agente si trova vicino alla finestra
- Il pavimento della stanza è bagnato
- L'altezza desiderata non è ancora stata raggiunta

- **Azioni:**

- L'altezza desiderata cambia aumentando di 1 metro rispetto al valore precedente
- La tasca dell'agente non contiene più il tronco
- N.B.: quando l'agente eseguirà questa regola viene simulato di che un tronco possa scivolare per via del pavimento bagnato. Questa regola, nel corso delle iterazioni, verrà scartata dall'agente in quanto ha un reward più basso della precedente

```

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)
}

```

FUGGIRE

- **Precondizioni:**
 - L'agente ha raggiunto l'altezza desiderata (cioè, ha impilato i due tronchi)
 - Il vetro della finestra è rotto

- **Azioni :**
 - L'agente fugge dalla stanza

```

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)
}

```

- **Precondizioni:**

- L'agente non ha creato la combo: "Fionda"
- Il vetro della finestra non è rotto

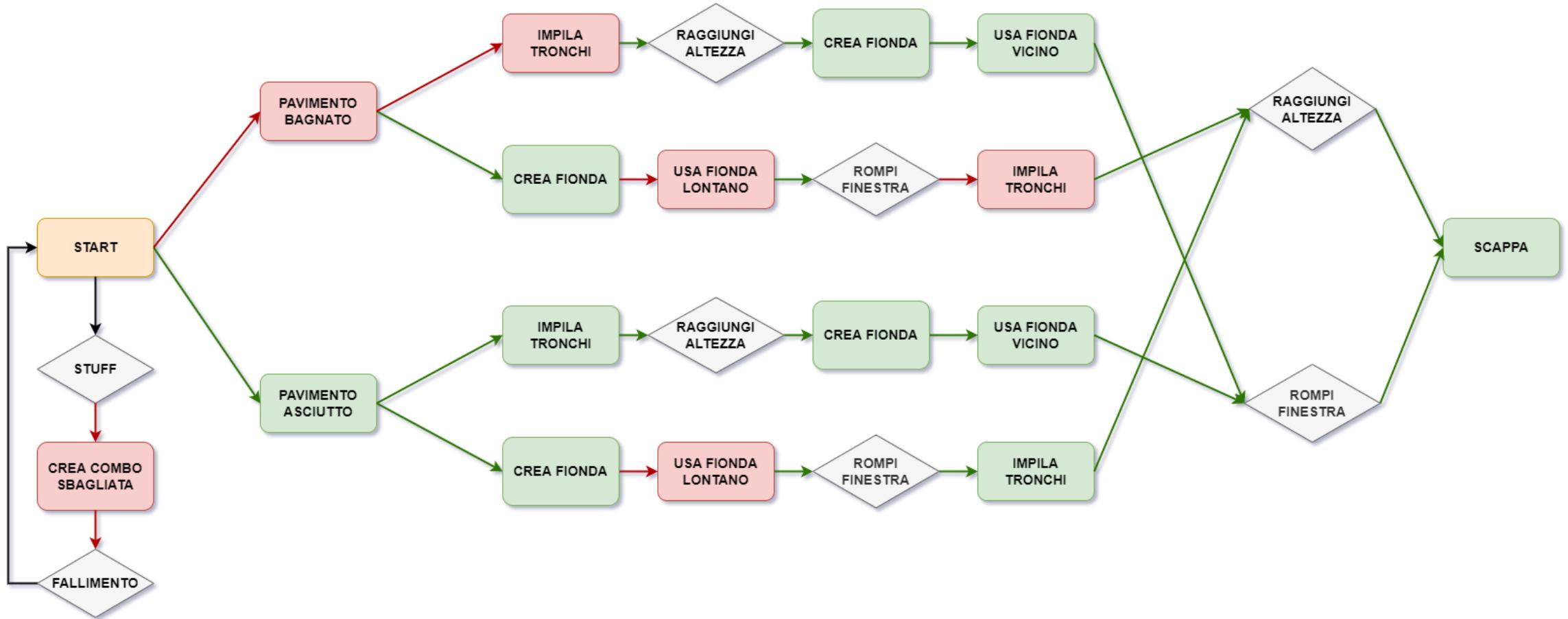
FALLIMENTO

- **Azioni :**

- L'agente fallisce la fuga dalla stanza

FLOWCHART

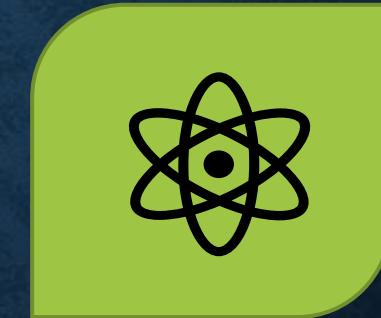




DEMO



CONCLUSIONI



CONSLUSIONI

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