UNIVERSITÀ
DI TORINO

INTELLIGENZA ARTIFICIALE & LABORATORIO

BATTAGLIA NAVALE
Progetto CLIPS - Parte n.2 - Prof. Micalizio

Data: 15 Settembre 2023

Delmastro Andrea (912954) - Ferrero Fabio (926392) - Frumento Giulia (834773)

Università degli Studi di Torino

INDICE

BATTAGLIA NAVALE

INTRODUZIONE BATTAGLIA NAVALE

In questo progetto viene utilizzato il linguaggio CLIPS al fine di modellare il comportamento di un sistema esperto che sia in grado di giocare a una versione semplificata di Battaglia Navale.

Il sistema ha a disposizione quattro tipi di azioni: *fire x y*, *guess x y*, *unguess x y* e *solve*. L'obiettivo del gioco è quello di annotare come *guessed* le 20 caselle dove si trovano le navi. Il sistema può usare 5 *fire* e 20 *guess*, inoltre il gioco termina automaticamente dopo 100 azioni.

MODELLAZIONE DELLA CONOSCENZA

MODELLAZIONE DELLA CONOSCENZA: **CREAZIONE** DI **FATTI**

Sono stati creati i seguenti **fatti non ordinati**:

- `(row-idx (multislot xs))` e `(col-idx (multislot xs))`: in fase di inizializzazione dell'agente, vengono valorizzati con tutti gli indici di riga e di colonna definiti per la tabella di gioco, rispettivamente.
- `(k-left-per-row (slot row) (slot num))` e `(k-left-per-col (slot col) (slot num))`: in fase di inizializzazione dell'agente vengono valorizzati ai valori di `k-per-row` e `k-per-col`, rispettivamente. Vengono decrementati ogni qualvolta si ha la certezza della presenza di una nave in una certa posizione.

MODELLAZIONE DELLA CONOSCENZA: **CREAZIONE** DI **FATTI**

Sono stati creati i seguenti **fatti non ordinati**:

- `(dec (slot x) (slot y))`: segnala all'agente la necessità di decrementare i valori di `k-left-per-row` e `k-left-per-col` nella posizione `(x, y)`.
- `(guess (slot x) (slot y))`: segnala all'agente la necessità di marcare come *guessed* una cella precedentemente marcata come *fired* di successo.
- `(guess-sure (slot x) (slot y))`: permette all'agente di asserire che è stata posizionata una *guess certa* sulla cella `(x, y)`.
- `(fired-gs (slot x) (slot y))`: permette all'agente di segnalare la cella `(x, y)` dove ha effettuato una *fire* a seguito di una *guess certa*.

MODELLAZIONE DELLE STRATEGIE

MODELLAZIONE DELLE STRATEGIE: **STRATEGIA 1**

La prima strategia si compone di **tre fasi** distinte:

1. Utilizzo immediato delle 5 *fire* disponibili;
2. Posizionamento delle *guess certe* data la conoscenza attuale;
3. Posizionamento delle *guess* rimanenti.

Il corretto ordine di esecuzione è assicurato da **salience decrescenti**.

MODELLAZIONE DELLE STRATEGIE: **STRATEGIA 2**

La seconda strategia si compone di **quattro fasi** distinte:

1. Posizionamento delle *guess certe* data la conoscenza iniziale;
2. Per ogni *guess certa* asserita si esegue una *fire* nel punto limitrofo più promettente;
3. Esegue una *fire* nel punto più promettente, nel caso `num. fire > 0`
4. Posizionamento delle *guess* rimanenti.

Il corretto ordine di esecuzione è assicurato da **salience decrescenti**.

MODELLAZIONE DELLE STRATEGIE: **PRINCIPIO DI OTTIMALITÀ**

Le operazioni (1) e (3) vengono svolte sulla base di un **principio di ottimalità** da noi definito: la cella più promettente è quella che massimizza la somma del numero di pezzi di barche per la sua riga e per la sua colonna.

```
1 (defrule max-fire (declare (salience 5))
2   ...
3   (not (exists (k-left-per-row (row ?x1) (num ?nx1))
4         (exists (k-left-per-col (col ?y1) (num ?ny1))
5           (not (k-cell (x ?x1) (y ?y1)))
6           (test (> (+ ?nx1 ?ny1) (+ ?nx ?ny)))
7   ))) =>  ...)
```

Analogamente viene svolta la `max-guess`.

MODELLAZIONE DELLE STRATEGIE: **GUESS CERTE**

Una *guess* viene definita **certa** quando le informazioni a disposizione sono sufficienti da avere la certezza che in una certa cella sia presente una barca.

```
1 (defrule guess-right-sure
2   ...
3   (k-cell (x ?x) (y ?y) (content left))
4   (not (exec (action guess) (x ?x) (y ?y1&:(eq ?y1 (+ ?y 1)))))
5   (not (k-cell (x ?x) (y ?y1&:(eq ?y1 (+ ?y 1)))))
6 =>  ...)
```

MODELLAZIONE DELLE STRATEGIE: **DIFFERENZE TRA LE STRATEGIE**

Strategia 1:

- Lo svolgimento delle fasi è *invariato* ad ogni partita;
- Utilizza le 5 fire all'*inizio del gioco*;
- Le fire sono *indipendenti* rispetto alle guess;

Strategia 2:

- Lo svolgimento delle fasi dipende dalla presenza di *conoscenza iniziale*;
- Utilizza le 5 fire *durante il gioco*;
- Ho due tipi di fire: quelle *dipendenti* dalle guess certe e quelle *indipendenti*;

MODELLAZIONE DELLE STRATEGIE: **SOMIGLIANZE TRA LE STRATEGIE**

I seguenti punto illustrano le somiglianze tra le due strategie:

- Le strategie usano lo stesso principio di ottimalità definito prima;
- Le regole che definiscono le guess certe sono le stesse per entrambe le strategie;
- Entrambe non fanno utilizzo dell'operazione di *unguess*: esattamente **26 operazioni** tra *fire* (5), *guess* (20), *solve* (1) vengono effettuate in entrambe le strategie.

MODELLAZIONE REGOLE DI EXPERTISE

REGOLE DI EXPERTISE: LE **FIRE** CASUALI

```
1  (defrule max-fire (declare (salience 5))
2    (status (step ?s) (currently running))
3    (moves (fires ?nf &:(> ?nf 0)))
4    (row-idx (xs $? ?x $?))
5    (col-idx (ys $? ?y $?))
6    (not (k-cell (x ?x) (y ?y)))
7    (k-left-per-row (row ?x) (num ?nx))
8    (k-left-per-col (col ?y) (num ?ny))
9    (not
10     (exists (k-left-per-row (row ?x1) (num ?nx1))
11       (exists (k-left-per-col (col ?y1) (num ?ny1))
12         (not (k-cell (x ?x1) (y ?y1)))
13         (test (> (+ ?nx1 ?ny1) (+ ?nx ?ny)))
14       )
15     )
16   )
17 =>
18   (assert(exec(step ?s)(action fire)(x ?x)(y ?y)))
19   (pop-focus)
20 )
```

Se sono ancora disponibili delle *fire*, seleziona l'indice di riga e l'indice di colonna tale per cui:

- il contenuto non sia noto
- non esista nessun altra combinazione di indici che identifichi una cella che rispetti le medesime condizioni e il cui numero di pezzi per riga e per colonna sia maggiore

REGOLE DI EXPERTISE: LE **GUESS** CASUALI

```
1  (defrule max-guess (declare (salience -5))
2    (status (step ?s) (currently running))
3    (moves (guesses ?ng &:(> ?ng 0)))
4    (row-idx (xs $? ?x $?))
5    (col-idx (ys $? ?y $?))
6    (not (exec (action guess) (x ?x) (y ?y)))
7    (not (k-cell (x ?x) (y ?y)))
8    (k-left-per-row (row ?x) (num ?nx))
9    (k-left-per-col (col ?y) (num ?ny))
10   (not
11     (exists (k-left-per-row (row ?x1) (num ?nx1))
12       (exists (k-left-per-col (col ?y1) (num ?ny1))
13         (not (exec (action guess) (x ?x1) (y ?y1)))
14         (not (k-cell (x ?x1) (y ?y1)))
15         (test (> (+ ?nx1 ?ny1) (+ ?nx ?ny)))
16       )
17     )
18   )
19 =>
20   (assert (exec (step ?s) (action guess) (x ?x) (y ?
         y)))
21   (assert (dec (x ?x) (y ?y)))
22   (pop-focus)
23 )
```

Se sono ancora disponibili delle *guess*, seleziona l'indice di riga e l'indice di colonna tale per cui:

- il contenuto non sia noto
- non vi sia stata eseguita una *guess* precedentemente
- non esista nessun altra combinazione di indici che identifichi una cella che rispetti le medesime condizioni e il cui numero di pezzi per riga e per colonna sia maggiore

REGOLE DI EXPERTISE: LE **GUESS** CERTE

```
1  (defrule guess-mid-ver-bot-sure
2    (status (step ?s) (currently running))
3    (k-cell (x ?x) (y ?y) (content middle))
4    (or
5      (k-left-col(col ?y1&:(eq ?y1(- ?y 1)))(num 0))
6      (k-left-col(col ?y2&:(eq ?y2(+ ?y 1)))(num 0))
7      (k-left-per-row (row ?x) (num 0))
8    )
9    (not (or
10       (exec (action guess)(x ?x)(y ?y3&:(eq ?y3(- ?y
               1))))
11       (exec (action guess)(x ?x)(y ?y4&:(eq ?y4(+ ?y
               1))))
12       (k-cell (x ?x) (y ?y5&:(eq ?y5 (- ?y 1))))
13       (k-cell (x ?x) (y ?y6&:(eq ?y6 (+ ?y 1))))
14   ))
15   (not(exec(action guess)(x ?x1&:(eq ?x1(+ ?x 1)))(y
          ?y)))
16   (not (k-cell (x ?x1&:(eq ?x1 (+ ?x 1))) (y ?y)))
17 =>
18   (assert(exec(step ?s)(action guess)(x(+ ?x 1))(y ?
        y)))
19   (assert (dec (x (+ ?x 1)) (y ?y)))
20   (pop-focus)
21 )
```

Se in posizione *(x, y)* è presente un pezzo di barca *middle* e nelle colonne a sinistra e a destra della cella considerata non sono certamente presenti altri pezzi di barche (`k-left-per-col` a 0), oppure nella riga considerata non sono presenti altri pezzi di barche (`k-left-per-row` a 0), allora effettua una *guess* nelle due celle sopra e sotto quella considerata.

REGOLE DI EXPERTISE: LE **FIRE** IN VICINANZA DI **GUESS** CERTE

Seleziona la cella

- in vicinanza di una seconda cella su cui sia stata effettuata una *guess* certa e nelle cui vicinanze non sia mai stata effettuata una *fire* precedentemente andata a buon fine (`guess-sure`)
- non esista nessun altra combinazione di indici che identifichi una cella che rispetti le medesime condizioni e il cui numero di pezzi per riga e per colonna sia maggiore

RISULTATI

RISULTATI: **CONFRONTO UNO**

Si è effettuato un confronto tra le strategie basandosi su **mappe differenti** e stesso numero di pezzi conosciuti all'inizio (3 pezzi noti).

Mappa	Strategia 1.a	Strategia 1.b	Strategia 2
Test 1	190	100	245
Test 2	280	280	335
Test 3	170	100	135
Test 4	225	225	225

RISULTATI: **CONFRONTO DUE**

Si è effettuato un confronto tra le strategie basandosi sulla **stessa mappa** e variando il numero di pezzi di barca noti all'inizio del gioco.

N. pezzi	1.a	1.b	2
0	25	-190	25
1	-65	-80	85
2	-10	-30	135
3	-10	-30	190
4	170	135	280

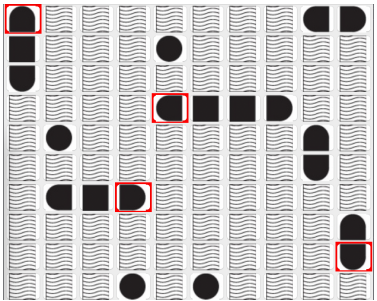

Figure: 4 pezzi noti