universidade de aveiro

Sistemas Operativos

Simulação de jogo de futebol

2020/2021

Diana Elisabete Siso Oliveira, nº 98607, P5 (60%)
Marta Sofia Azevedo Fradique, nº 98629, P5 (40%)

Índice

1. Introdução

Neste trabalho vamos construir uma simulação de um jogo de futebol constituído por um árbitro, diferentes jogadores de campo e diferentes guarda-redes. Ora, as equipas são constituídas por quatro jogadores de campo e por um guarda-redes, e vão sendo formadas à medida que os jogadores chegam. Na totalidade existem dez jogadores de campo e três guarda-redes, assim, se algum jogador de campo ou guarda-redes chegar após a formação da equipa, eles são avisados de que chegaram atrasados e não entram no jogo. Ainda, o papel do árbitro também é fulcral pois este é responsável por iniciar e terminar a partida.

Os principais objetivos do nosso trabalho são desenvolver as nossas aptidões no uso de semáforos, bem como aprimorar os nossos conhecimentos sobre o funcionamento de memória partilhada.

Neste trabalho, inicialmente falamos sobre os semáforos e sobre as variáveis que constituem o *script*. Seguidamente, analisamos as funções presentes no código dos seguintes componentes: árbitro, guarda-redes e jogador de campo. Por último, realizamos os testes para verificarmos se os nossos resultados tiveram um bom desempenho.

2. Semáforos e Variáveis

Para a realização deste projeto necessitamos de sete semáforos diferentes: *playersWaitReferee*, *refereeWaitTeams*, *playerRegistered*, *playersWaitTeam*, *goaliesWaitTeam*, *playersWaitEnd* e *mutex* (este último foi nos fornecido já configurado, pelo que não foi necessário alterá-lo).

Ora, os semáforos "*playersWaitTeam*" e "*goaliesWaitTeam*" são usados pelos jogadores de campo e pelos guarda-redes, respetivamente, para esperar pela formação da sua equipa; assim sendo, os *downs* serão realizados pelos jogadores de campo e/ou guarda-redes que, ao chegarem, não verificam a existência de condições necessárias para a formação de uma equipa (1 *down* por jogador de campo/guarda-redes), enquanto que a entidade responsável por realizar os *ups* será o capitão (4 *ups*), simbolizando as quatro entidades que esperaram pela formação da equipa.

Também, o semáforo "*playersWaitReferee*" é usado pelos jogadores de campo e pelos guarda-redes enquanto esperam que o árbitro inicie a partida, sendo, portanto, os *downs* realizados pelos jogadores/guarda-redes (1 *down* por cada) e os *ups* realizados pelo árbitro (10 *ups*, 1 por cada *down* realizado).

Ainda, o semáforo "*playersWaitEnd*" é utilizado pelos jogadores de campo e pelos guarda-redes, simbolizando a espera pelo fim da partida e, por conseguinte, as entidades competentes de realizar os *downs* serão os jogadores/guarda-redes (1 *down* por cada) e a entidade competente de realizar os *ups* será o árbitro (10 *ups*, 1 por cada *down* realizado).

Adicionalmente, o semáforo "*refereeWaitTeams*" é utilizado pelo árbitro para esperar que as equipas sejam formadas, ou seja, os *downs* serão cedidos pelo árbitro (2 *downs*, 1 por cada capitão) e, quando as equipas ficarem completamente constituídas, o capitão de cada uma é responsável por emitir os *up* (1 *up* cada).

Por último, o semáforo "*playerRegistered*" é ligado aos jogadores de campo e aos guarda-redes, de forma a que estes possam se registar numa equipa e, portanto, a entidade responsável por fornecer os *downs* serão os jogadores/guarda-redes que se estão a juntar à equipa mas aguardam a sua formação completa (1 *down* por cada) e a entidade incumbida de realizar os *ups* será o capitão da equipa, simbolizando que a equipa está completa (4 *ups*, 1 por cada jogador e/ou guarda-redes que se registou anteriormente).

Para escrever o programa usamos diferentes variáveis pertencentes a duas estruturas de dados: "STAT "e "FULL_STAT".

Assim, a primeira estrutura de dados "STAT" é referente aos estados dos componentes do jogo, "*playerStat[*NUMPLAYERS*]*" corresponde aos estados dos jogadores de campo, "*goalieStat[*NUMPLAYERS*]*" é referente aos estados dos guarda-redes, e por último "*refereeStat*" que indica o estado do árbitro.

Em seguida, na segunda estrutura de dados "FULL_STAT" temos variáveis como "*nPlayers*", "*nGoalies*" e "*nReferees*" as quais representam respetivamente o número total de jogadores de campo, o número total de guarda-redes e o número total de árbitros. Também, podemos encontrar a variável "*playersArrived*" que significa o número de jogadores de campo que já chegaram ao local da partida, e, "*goaliesArrived*" esta revela o número de guarda-redes que também já compareceram no local da partida. Ainda, podemos encontrar a variável "*playersFree*" a qual corresponde ao número de jogadores que já chegaram e ainda não foram atribuídos a nenhuma equipa, e por último, a variável "*goaliesFree*" que se refere ao número de guarda-redes que já chegaram e também ainda não foram atribuídos a uma equipa.

Na tabela abaixo, podemos analisar em cada semáforo pertencente ao nosso programa, as entidades que lhe dão "*up*" e "*down*" bem como o número de "*ups*" e "*downs*". Podemos ainda observar as funções onde estas mudanças ocorrem.

Semáforos	Entidade que dá *up*	Entidade dá *down*	Número de *ups*	Número de *downs*	Função onde ocorre o *up*	Função onde ocorre o *down*
playersWaitTeam	capitão	jogadores e guarda-redes	3/4	1/0	*playerConstituteTeam( )* e *goalieConstituteTeam( )*	*playerConstituteTeam( )* e *goalieConstituteTeam( )*
goaliesWaitTeam	capitão	jogadores e guarda-redes	1/0	1/0	*goalieConstituteTeam( )* e *playerConstituteTeam( )*	*goalieConstituteTeam( )* e *playerConstituteTeam( )*
playersWaitReferee	árbitro	jogadores e guarda-redes	10	1 cada	*startGame( )*	*waitReferee( )*
refereeWaitTeams	capitão	árbitro	1 cada	2	*playerConstituteTeam( )* e *goalieConstituteTeam( )*	*waitForTeams( )*
playersWaitEnd	árbitro	jogadores e guarda-redes	10	1 cada	*endlEnd( )*	*playUntilEnd( )*
playersRegistered	jogadores e guarda-redes que não sejam capitães	capitão	1 cada	1 cada	*playerConstituteTeam( )* e *goalieConstituteTeam( )*	*playerConstituteTeam( )* e *goalieConstituteTeam( )*

3. Código do Árbitro

A entidade árbitro, ao longo do código, pode assumir cinco estados: ARRIVING, simbolizando que está a chegar ao campo e com valor 0; WAITING_TEAMS, simbolizando que está à espera que as equipas sejam formadas e com valor 1; STARTING_GAME, simbolizando que o árbitro está a começar o jogo de futebol e com valor 2; REFEREEING, simbolizando que o árbitro está a arbitrar o jogo de futebol e com valor 3; ENDING_GAME, simbolizando que o árbitro está a encerrar o jogo de futebol e com valor 4.

Para que a entidade percorra os cinco estados mencionados, vão ser utilizadas cinco funções, as quais serão explicadas nos próximos tópicos: *arrive( )*, *waitForTeams( )*, *startGame( )*, *play( )* e *endGame( )*. Estas funções serão percorridas pela ordem em que foram referidas.

3.1 Função Arrive

Na função arrive( ) correspondente ao árbitro, a única alteração que foi feita foi a mudança de estado da entidade para ARRIVING dentro da zona crítica do mutex e o registro do estado.

```
static void arrive() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.refereeStat = ARRIVING;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      usleep((100.0*random())/(RAND_MAX+1.0)+10.0);
}
```

3.2 Função WaitForTeams

Na função waitForTeams( ) correspondente ao árbitro, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para WAITING_TEAMS e o registro do estado.

```
static void waitForTeams() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.refereeStat = WAITING_TEAMS;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, o árbitro dá dois *downs* no semáforo *refereeWaitTeams*, simbolizando que está à espera de receber a confirmação dos dois capitães que as equipas foram formadas.

```
        for (int i=0; i<2; i++) {
              if (semDown (semgid, sh->refereeWaitTeams) == -1) {
                    perror ("error on the down operation for semaphore access (RF)");
                    exit (EXIT_FAILURE);
              }
        }
  }
```

3.3 Função StartGame

Na função startGame( ) correspondente ao árbitro, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para STARTING_GAME e o registro do estado.

```
static void startGame() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.refereeStat = STARTING_GAME;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, o árbitro dá dez *ups* no semáforo *playersWaitReferee*, simbolizando a entrega da confirmação aos dez jogadores que irão jogar o jogo de futebol o mesmo começou.

```
        for (int i=0; i<(NUMTEAMPLAYERS+NUMTEAMGOALIES)*2 ; i++) {
              if (semDown (semgid, sh->refereeWaitTeams) == -1) {
                    perror ("error on the down operation for semaphore access (RF)");
                    exit (EXIT_FAILURE);
              }
        }
  }
```

3.4 Função Play

Na função play( ) correspondente ao árbitro, a única alteração que foi feita foi a mudança de estado da entidade para REFEREEING dentro da zona crítica do mutex e o registro do estado.

```
static void paly() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.refereeStat = REFEREEING;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      usleep((100.0*random())/(RAND_MAX+1.0)+900.0);
}
```

3.5 Função EndGame

Na função endGame( ) correspondente ao árbitro, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para *ENDING_GAME* e o registro do estado.

```
static void endGame() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.refereeStat = ENDING_GAME;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, o árbitro dá dez *ups* no semáforo *playersWaitEnd*, simbolizando a entrega da confirmação aos dez jogadores que jogaram o jogo de futebol que o mesmo terminou.

```
      for (int i=0; i<(NUMTEAMPLAYERS+NUMTEAMGOALIES)*2 ; i++) {
            if (semDown (semgid, sh->playersWaitEnd) == -1) {
                  perror ("error on the down operation for semaphore access (RF)");
                  exit (EXIT_FAILURE);
            }
      }
 }
```

4. Código do Guarda-Redes

A entidade guarda-redes, ao longo do código, pode assumir oito estados: ARRIVING, com valor 0, simbolizando que está a chegar ao campo; *WAITING_TEAM*, com valor 1, simbolizando que o guarda-redes chegou e está à espera que haja condições para que as equipas sejam formadas; *FORMING_TEAM*, com valor 2, simbolizando que o guarda-redes chegou e verificou que há condições de formação de equipa; *WAITING_START_1*, com valor 3, simbolizando que o guarda-redes está à espera que o jogo comece e pertence à equipa 1; *WAITING_START_2*, com valor 4, simbolizando que o guarda-redes está à espera que o jogo comece e pertence à equipa 2; *PLAYING_1*, com valor 5, simbolizando que o guarda-redes está a jogar o jogo e pertence à equipa 1; *PLAYING_2*, com valor 6, simbolizando que o guarda-redes está a jogar o jogo e pertence à equipa 2; *LATE*, com valor 7, simbolizando que o guarda-redes chegou atrasado e por isso não irá jogar. Este último estado ocorre porque apenas dois guarda-redes irão jogar - um em cada equipa -, porém teremos presentes três guarda-redes.

Para que a entidade percorra os oito estados mencionados, vão ser utilizadas quatro funções, as quais serão explicadas nos próximos tópicos: *arrive( )*, *goalieConstituteTeam( )*, *waitReferee( )* e *playUntilEnd( )*. Estas funções serão percorridas pela ordem em que foram referidas.

4.1 Função Arrive

Na função arrive( ) correspondente ao guarda-redes a única alteração que foi feita foi a mudança de estado da entidade para ARRIVING dentro da zona crítica do mutex e o registro do estado.

```
static void arrive() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.goalieStat = ARRIVING;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      usleep((200.0*random())/(RAND_MAX+1.0)+60.0);
}
```

4.2 Função GoalieConstituteTeam

Na função goalieConstituteTeam( ) realizamos mais alterações na zona crítica do mutex comparativamente às restantes funções da implementação. Inicialmente, incrementamos a variável *goaliesArrived*, uma vez que, se o guarda-redes chegou a esta etapa é porque, obviamente, ele já se encontra presente. De seguida, através de condições *if*, realizamos a filtragem do estado do guarda-redes: se já tiverem chegado dois guarda-redes, o terceiro está atrasado, pelo que não irá jogar e ficará com o estado LATE; se ainda não tiverem chegado dois guarda-redes, o que chegou assumirá o estado WAITING_TEAM, incrementamos o valor da variável *goaliesFree* e atualizamos o valor da variável *teamId* e da variável *ret* - variável que a função irá retornar e que simboliza a equipa onde o guarda-redes se encontra. Note que se apenas um guarda-redes chegou, esse guarda-redes estará na equipa 1 quando for possível formá-la; se já tiver chegado um guarda-redes, o que chegar a seguir estará na equipa 2 quando for possível formá-la; o terceiro guarda-redes não é atribuído a nenhuma equipa porque o seu estado é LATE.

```
static int goalieConstituteTeam (int id) {
      int ret = 0;
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.goaliesArrived++;
      if (sh->fSt.goaliesArrived > 2) {
            sh->fSt.st.goalieStat[id] = LATE;
      } else {
            sh->fSt.goaliesFree++;
            sh->fSt.st.goalieStat[id] = WAITING_TEAM;
            if (sh->fSt.goaliesArrived <= 1) {
                  ret = 1;
                  sh->fSt.teamId=ret;
            } else {
                  ret = 2;
                  sh->fSt.teamId=ret;
            }
      }
```

Ainda dentro da zona crítica do mutex, avaliamos a viabilidade da formação da equipa, isto é: se houver pelo menos quatro jogadores livres e um guarda-redes livre, então é possível formar uma equipa. Ao ser possível formar uma equipa, o estado do guarda-redes - que será o capitão dessa equipa - mudará para FORMING_TEAM e decrementamos as variáveis *playersFree* e *goaliesFree*, porque os quatro jogadores e o guarda-redes já não encontrar-se-ão disponíveis. Para além disso, se for viável formar equipa, então o guarda-redes dá quatro *ups* no semáforo *playersWaitTeam*, simbolizando a entrega da confirmação que quatro jogadores ingressaram numa equipa, dá quatro *downs* no semáforo *playerRegistered*, simbolizando que recebeu as

confirmações que esses jogadores ingressaram na equipa, e dá um *up* no semáforo *refereeWaitTeams*, simbolizando a entrega da confirmação ao árbitro que uma equipa foi formada. Por último, realizamos o registro do estado do guarda-redes.

```
        if (sh->fSt.playersFree >= 4 && sh->fSt.goaliesFree >= 1) {
              sh->fSt.st.goalieStat[id] = FORMING_TEAM;
              sh->fSt.playersFree-= NUMTEAMPLAYERS;
              sh->fSt.goaliesFree--;
              for (int i=0; i<NUMTEAMPLAYERS; i++) {
                    if (semUp (semgid, sh->playersWaitTeam) == -1) {
                       perror ("error on the up operation for semaphore access (PL)");
                       exit (EXIT_FAILURE);
                    }
                    if (semDown (semgid, sh->playerRegistered) == -1)  {
                      perror ("error on the down operation for semaphore access (PL)");
                      exit (EXIT_FAILURE);
                    }
              }
              if (semUp (semgid, sh->refereeWaitTeams) == -1) {
                    perror ("error on the up operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
              }
        }
        saveState(nFic, &sh->fSt);
        if (semUp (semgid, sh->mutex) == -1) {
              perror ("error on the up operation for semaphore access (GL)");
              exit (EXIT_FAILURE);
        }
```

Fora da zona crítica do mutex, realizamos a condição de verificação se o guarda-redes é capitão da equipa ou não, isto é, se o estado dele é FORMING_TEAM ou WAITING_TEAM. Caso não seja o capitão, o guarda-redes realiza um *up* no semáforo *playerRegistered*, simbolizando a confirmação de que se registrou na equipa, e dá um *down* no semáforo *goaliesWaitTeam*, simbolizando que está à espera que chegue um elemento que verifique que há condições para se formar a equipa com os membros já registrados.

```
        if (sh->fSt.st.goalieStat[id] == WAITING_TEAM) {
              if (semDown (semgid, sh->goaliesWaitTeam) == -1)  {
                    perror ("error on the down operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
              }
              if (semUp(semgid, sh->playerRegistered) == -1)  {
                    perror ("error on the up operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
              }
        }
        return ret;
}
```

4.3 Função WaitReferee

Na função waitReferee( ) correspondente ao guarda-redes, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para WAITING_START_1, se o guarda-redes pertence à equipa 1, ou WAITING_START_2, se o guarda-redes pertence à equipa 2, e o registro do estado.

```
static void waitReferee (int id, int team) {
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      if (team == 1) {
            sh->fSt.st.goalieStat[id] = WAITING_START_1;
            saveState(nFic, &sh->fSt);
      } else {
            sh->fSt.st.goalieStat[id] = WAITING_START_2;
            saveState(nFic, &sh->fSt);
      }
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, o guarda-redes dá um *down* no semáforo *playersWaitReferee*, simbolizando que está à espera de receber a confirmação do árbitro que este está preparado para arbitrar.

```
      if (semDown (semgid, sh->playersWaitReferee) == -1)  {
            perror ("error on the down operation for semaphore access (PL)");
            exit (EXIT_FAILURE);
      }
}
```

4.4 Função PlayUntilEnd

Na função playUntilEnd( ) correspondente ao guarda-redes, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para PLAYING_1, se o guarda-redes pertence à equipa 1, ou PLAYING_2, se o guarda-redes pertence à equipa 2, e o registro do estado.

```
static void playUntilEnd (int id, int team) {
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      if (team == 1) {
            sh->fSt.st.goalieStat[id] = PLAYING_1;
```

```
            saveState(nFic, &sh->fSt);
        } else {
            sh->fSt.st.goalieStat[id] = PLAYING_2;
            saveState(nFic, &sh->fSt);
        }
        if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
        }
```

Fora da zona crítica do mutex, o guarda-redes dá um *down* no semáforo *playersWaitEnd* simbolizando que está à espera de receber a confirmação do árbitro que o jogo terminou.

```
        if (semDown (semgid, sh->playersWaitEnd) == -1)  {
            perror ("error on the down operation for semaphore access (PL)");
            exit (EXIT_FAILURE);
        }
}
```

5. Código do Jogador

A entidade jogador, ao longo do código, pode assumir oito estados: ARRIVING, com valor 0, simbolizando que está a chegar ao campo; WAITING_TEAM, com valor 1, simbolizando que o jogador chegou e está à espera que haja condições para que as equipas sejam formadas; FORMING_TEAM, com valor 2, simbolizando que o jogador chegou e verificou que há condições de formação de equipa; WAITING_START_1, com valor 3, simbolizando que o jogador está à espera que o jogo comece e pertence à equipa 1; WAITING_START_2, com valor 4, simbolizando que o jogador está à espera que o jogo comece e pertence à equipa 2; PLAYING_1, com valor 5, simbolizando que o jogador está a jogar o jogo e pertence à equipa 1; PLAYING_2, com valor 6, simbolizando que o jogador está a jogar o jogo e pertence à equipa 2; LATE, com valor 7, simbolizando que o jogador chegou atrasado e por isso não irá jogar. Este último estado ocorre porque apenas oito jogadores irão jogar - quatro em cada equipa -, porém teremos presentes dez jogadores

Para que a entidade percorra os oito estados mencionados, vão ser utilizadas quatro funções, as quais serão explicadas nos próximos tópicos: *arrive( )*, *playerConstituteTeam( )*, *waitReferee( )* e *playUntilEnd( )*. Estas funções serão percorridas pela ordem em que foram referidas.

5.1 Função Arrive

Na função arrive( ) correspondente ao jogador a única alteração que foi feita foi a mudança de estado da entidade para ARRIVING dentro da zona crítica do mutex e o registro do estado.

```
static void arrive() {
      if (semDown (semgid, sh->mutex) == -1) {
            perror ("error on the down operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.st.playerStat = ARRIVING;
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
      }
      usleep((200.0*random())/(RAND_MAX+1.0)+50.0);
}
```

5.2 Função PlayerConstituteTeam

Na função playerConstituteTeam( ) realizamos mais alterações na zona crítica do mutex comparativamente às restantes funções da implementação. Inicialmente, incrementamos a variável *playersArrived*, uma vez que, se o jogador chegou a esta etapa é porque, obviamente, ele já se encontra presente. De seguida, através de condições *if*, realizamos a filtragem do estado do

jogador: se já tiverem chegado oito jogadores, os próximos estão atrasados, pelo que não irão jogar e ficarão com o estado LATE; se ainda não tiverem chegado oito jogadores, o que chegou assumirá o estado WAITING_TEAM, incrementamos o valor da variável *playersFree* e atualizamos o valor da variável *teamId* e da variável *ret* - variável que a função irá retornar e que simboliza a equipa onde o jogador se encontra. Note que se apenas quatro jogadores chegaram, esses quatro jogadores estarão na equipa 1 quando for possível formá-la; se já tiver chegado quatro jogadores, os que chegarem a seguir estarão na equipa 2 quando for possível formá-la; os últimos dois jogadores não são atribuídos a nenhuma equipa porque o seu estado é LATE.

```
static int playerConstituteTeam (int id) {
      int ret = 0;
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      sh->fSt.playersArrived++;
      if (sh->fSt.playersArrived > 2) {
            sh->fSt.st.playerStat[id] = LATE;
      } else {
            sh->fSt.playersFree++;
            sh->fSt.st.playerStat[id] = WAITING_TEAM;
            if (sh->fSt.playersArrived <= 4) {
                  ret = 1;
                  sh->fSt.teamId=ret;
            } else {
                  ret = 2;
                  sh->fSt.teamId=ret;
            }
      }
```

Ainda dentro da zona crítica do mutex, avaliamos a viabilidade da formação da equipa, isto é: se houver pelo menos quatro jogadores livres e um guarda-redes livre, então é possível formar uma equipa. Ao ser possível formar uma equipa, o estado do jogador - que será o capitão dessa equipa - mudará para FORMING_TEAM e decrementamos as variáveis *playersFree* e *goaliesFree*, porque os quatro jogadores e o guarda-redes já não encontrar-se-ão disponíveis. Para além disso, se for viável formar equipa, então o jogador dá três *ups* no semáforo *playersWaitTeam* e um *up* no semáforo *goaliesWaitTeam*, simbolizando a entrega da confirmação que três jogadores e um guarda-redes ingressaram numa equipa, dá quatro *downs* no semáforo *playerRegistered*, simbolizando que recebeu as confirmações que esses jogadores e guarda-redes ingressaram na equipa, e dá um *up* no semáforo *refereeWaitTeams*, simbolizando a entrega da confirmação ao árbitro que uma equipa foi formada. Por último, realizamos o registro do estado do jogador.

```
      if (sh->fSt.playersFree >= 4 && sh->fSt.goaliesFree >= 1) {
            sh->fSt.st.playerStat[id] = FORMING_TEAM;
            sh->fSt.playersFree-= NUMTEAMPLAYERS;
```

```
            sh->fSt.goaliesFree--;
            for (int i=1; i<NUMTEAMPLAYERS; i++) {
                  if (semUp (semgid, sh->playersWaitTeam) == -1) {
                     perror ("error on the up operation for semaphore access (PL)");
                     exit (EXIT_FAILURE);
                  }
                  if (semDown (semgid, sh->playerRegistered) == -1)  {
                    perror ("error on the down operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
                  }
            }
            if (semUp (semgid, sh->goaliesWaitTeam) == -1) {
                  perror ("error on the up operation for semaphore access (PL)");
                  exit (EXIT_FAILURE);
            }
            if (semDown (semgid, sh->playerRegistered) == -1)  {
                  perror ("error on the down operation for semaphore access (PL)");
                  exit (EXIT_FAILURE);
            }
            if (semUp (semgid, sh->refereeWaitTeams) == -1) {
                  perror ("error on the up operation for semaphore access (PL)");
                  exit (EXIT_FAILURE);
            }
      }
      saveState(nFic, &sh->fSt);
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, realizamos a condição de verificação se o jogador é capitão da equipa ou não, isto é, se o estado dele é FORMING_TEAM ou WAITING_TEAM. Caso não seja o capitão, o jogador realiza um *up* no semáforo *playerRegistered*, simbolizando a confirmação de que se registrou na equipa, e dá um *down* no semáforo *playersWaitTeam*, simbolizando que está à espera que chegue um elemento que verifique que há condições para se formar a equipa com os membros já registrados.

```
      if (sh->fSt.st.goalieStat[id] == WAITING_TEAM) {
            if (semDown (semgid, sh->goaliesWaitTeam) == -1)  {
                  perror ("error on the down operation for semaphore access (PL)");
                  exit (EXIT_FAILURE);
            }
            if (semUp(semgid, sh->playerRegistered) == -1)  {
                  perror ("error on the up operation for semaphore access (PL)");
                  exit (EXIT_FAILURE);
            }
      }
      return ret;
}
```

5.3 Função WaitReferee

Na função waitReferee( ) correspondente ao jogador, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para *WAITING_START_1*, se o jogador pertence à equipa 1, ou *WAITING_START_2*, se o jogador pertence à equipa 2, e o registro do estado.

```
static void waitReferee (int id, int team) {
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      if (team == 1) {
            sh->fSt.st.playerStat[id] = WAITING_START_1;
            saveState(nFic, &sh->fSt);
      } else {
            sh->fSt.st.playerStat[id] = WAITING_START_2;
            saveState(nFic, &sh->fSt);
      }
      if (semUp (semgid, sh->mutex) == -1) {
            perror ("error on the up operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
```

Fora da zona crítica do mutex, o jogador dá um *down* no semáforo *playersWaitReferee*, simbolizando que está à espera de receber a confirmação do árbitro que este está preparado para arbitrar.

```
      if (semDown (semgid, sh->playersWaitReferee) == -1)  {
            perror ("error on the down operation for semaphore access (PL)");
            exit (EXIT_FAILURE);
      }
}
```

5.4 Função PlayUntilEnd

Na função playUntilEnd( ) correspondente ao jogador, a única alteração realizada dentro do mutex foi a mudança de estado da entidade para *PLAYING_1*, se o jogador pertence à equipa 1, ou *PLAYING_2*, se o jogador pertence à equipa 2, e o registro do estado.

```
static void playUntilEnd (int id, int team) {
      if (semDown (semgid, sh->mutex) == -1)  {
            perror ("error on the down operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
      }
      if (team == 1) {
            sh->fSt.st.playerStat[id] = PLAYING_1;
            saveState(nFic, &sh->fSt);
```

```
        } else {
              sh->fSt.st.playerStat[id] = PLAYING_2;
              saveState(nFic, &sh->fSt);
        }
        if (semUp (semgid, sh->mutex) == -1) {
              perror ("error on the up operation for semaphore access (GL)");
              exit (EXIT_FAILURE);
        }
```

Fora da zona crítica do mutex, o jogador dá um *down* no semáforo *playersWaitEnd* simbolizando que está à espera de receber a confirmação do árbitro que o jogo terminou.

```
        if (semDown (semgid, sh->playersWaitEnd) == -1)  {
              perror ("error on the down operation for semaphore access (PL)");
              exit (EXIT_FAILURE);
        }
}
```

6. Situações de Avaliação da Abordagem

Neste ponto realizamos testes para avaliar a assertividade do nosso código e localizarmos algum erro que possamos ter cometido ao elaborar o *script*. Deste modo, foram realizados 4 testes:

1. Verificação da correta formação de equipas quando os capitães são dois guarda-redes;
2. Verificação da correta formação de equipas quando os capitães são um guarda-redes e um jogador de campo;
3. Verificação da correta formação de equipas quando os capitães são 2 jogadores de campo;
4. Verificação do comportamento do árbitro ao longo da partida;
5. Verificação do *script* na globalidade.

Ora, para verificarmos os testes observamos as colunas e as linhas das transições de estados para saber se estão corretas, isto é, se respeitam determinadas regras que mencionaremos a seguir.

6.1 Teste 1 - Verificação da correta formação de equipas quando os capitães são dois guarda-redes

O primeiro teste que realizamos tem como finalidade averiguar o comportamento correto das entidades envolvidas na formação das equipas quando estas têm como capitães (jogadores responsáveis por afirmar que já é possível formar *teams*) guarda-redes.

O que pretendemos analisar será:

- A correta transição de estados nos capitães e, para tal, é necessário que em duas das três colunas que estão atribuídas a esta entidade ocorra a mudança de estados de “0” para “2”, tal como podemos observar no exemplo a seguir.
- A correta formação das equipas, ou seja, se em oito das dez colunas reservadas para os jogadores e em duas das três colunas reservadas para os guarda-redes ocorre a transição do estado “1” para o estado “3” ou para o estado “4” de acordo com a equipa onde cada jogador encontrar-se-á.

Ao realizarmos o teste, verificamos que o *script* desenvolvido atendeu às regras propostas.

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	2	0	0	0
0	1	1	1	0	1	0	0	1	0	2	0	0	0
0	1	1	1	0	1	0	0	1	0	2	0	0	0
0	1	1	1	0	1	0	1	1	0	2	0	0	0
0	1	1	1	0	1	0	1	1	1	2	0	0	0
0	1	1	1	0	1	0	1	1	1	2	0	0	0
0	1	1	1	0	3	0	1	1	1	2	0	0	0
0	1	3	1	0	3	0	1	1	1	2	0	0	0
0	3	3	1	0	3	0	1	1	1	2	0	0	0
1	3	3	1	0	3	0	1	1	1	2	0	0	0
1	3	3	3	0	3	0	1	1	1	2	0	0	0
1	3	3	3	0	3	0	1	1	1	3	0	0	0
1	3	3	3	0	3	0	1	1	1	3	2	0	0
1	3	3	3	7	3	0	1	1	1	3	2	0	0
1	3	3	3	7	3	0	1	1	1	3	2	0	0
1	3	3	3	7	3	0	1	1	1	3	2	0	0
1	3	3	3	7	3	0	1	1	1	3	2	0	1
1	3	3	3	7	3	0	1	4	1	3	2	0	1
1	3	3	3	7	3	0	4	4	1	3	2	0	1
1	3	3	3	7	3	0	4	4	4	3	2	0	1
4	3	3	3	7	3	0	4	4	4	3	2	0	1
4	3	3	3	7	3	0	4	4	4	3	4	0	1
4	3	3	3	7	3	7	4	4	4	3	4	0	1
4	3	3	3	7	3	7	4	4	4	3	4	0	2
4	3	3	3	7	5	7	4	4	4	3	4	0	2
4	5	3	3	7	5	7	4	4	4	3	4	0	2
4	5	3	3	7	5	7	4	6	4	3	4	0	2
4	5	3	3	7	5	7	4	6	4	5	4	0	2
4	5	3	5	7	5	7	4	6	4	5	4	0	2
4	5	3	5	7	5	7	6	6	4	5	4	0	2
4	5	3	5	7	5	7	6	6	4	5	4	0	3
6	5	3	5	7	5	7	6	6	4	5	4	0	3
6	5	3	5	7	5	7	6	6	6	5	4	0	3
6	5	3	5	7	5	7	6	6	6	5	6	0	3
6	5	3	5	7	5	7	6	6	6	5	6	7	3
6	5	5	5	7	5	7	6	6	6	5	6	7	3
6	5	5	5	7	5	7	6	6	6	5	6	7	4

6.2 Teste 2 - Verificação da correta formação de equipas quando os capitães são um guarda-redes e um jogador de campo

O segundo teste que realizamos tem como finalidade averiguar o comportamento correto das entidades envolvidas na formação das equipas quando uma delas tem como capitão um guarda-redes e a outra tem como capitão um jogador de campo.

O que pretendemos analisar será:

- A correta transição de estados nos capitães e, para tal, é necessário que em uma das três colunas que estão atribuídas à entidade guarda-redes e em uma das dez colunas que estão atribuídas à entidade jogadores de campo ocorra a mudança de estados de "0" para "2", tal como podemos observar no exemplo a seguir.
- A correta formação das equipas, ou seja, se em oito das dez colunas reservadas para os jogadores e em duas das três colunas reservadas para os guarda-redes ocorre a transição do estado "1" para o estado "3" ou para o estado "4" de acordo com a equipa onde cada jogador encontrar-se-á.

Ao realizarmos o teste, verificamos que o *script* desenvolvido atendeu às regras propostas.

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02  R01
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   0   0   0   0   0   0   0     0   0   0     0
 0   0   0   1   0   0   0   0   0   0     0   0   0     0
 0   0   0   1   0   0   0   0   0   0     0   0   0     0
 0   0   0   1   0   0   0   0   0   0     0   0   0     0
 0   0   0   1   0   0   0   0   0   0     0   0   0     0
 0   0   1   1   0   0   0   0   0   0     0   0   0     0
 0   0   1   1   0   0   0   0   0   0     0   0   0     0
 0   0   1   1   0   1   0   0   0   0     0   0   0     0
 0   0   1   1   0   1   1   0   0   0     0   0   0     0
 0   1   1   1   0   1   1   0   0   0     0   0   0     0
 0   1   1   1   0   1   1   0   0   0     0   0   0     0
 0   1   1   1   1   1   1   0   0   0     0   0   0     0
 0   1   1   1   1   1   1   0   0   0     0   0   0     0
 0   1   1   1   1   1   1   0   0   0     0   0   0     0
 0   1   1   1   1   1   1   0   0   0     0   0   0     0
 0   1   1   1   1   1   1   0   0   0     0   2   0     0
 0   1   1   1   1   1   1   0   0   0     1   2   0     0
 0   1   1   1   1   1   1   0   0   1     1   2   0     0
 0   1   1   1   1   1   1   0   0   1     1   2   0     1
 0   1   1   1   1   1   1   0   2   1     1   2   0     1
 0   1   1   3   1   1   1   0   2   1     1   2   0     1
 0   1   3   3   1   1   1   0   2   1     1   2   0     1
 0   1   3   3   1   3   1   0   2   1     1   2   0     1
 0   1   3   3   1   3   3   0   2   1     1   2   0     1
 0   1   3   3   1   3   3   0   2   1     1   3   0     1
 0   4   3   3   1   3   3   0   2   1     1   3   0     1
 0   4   3   3   1   3   3   7   2   1     1   3   0     1
 0   4   3   3   4   3   3   7   2   1     1   3   0     1
 0   4   3   3   4   3   3   7   2   4     1   3   0     1
 0   4   3   3   4   3   3   7   2   4     4   3   0     1
 0   4   3   3   4   3   3   7   2   4     4   3   7     1
 0   4   3   3   4   3   3   7   4   4     4   3   7     1
 0   4   3   3   4   3   3   7   4   4     4   3   7     2
 0   4   3   5   4   3   3   7   4   4     4   3   7     2
 0   4   5   5   4   3   3   7   4   4     4   3   7     2
 0   4   5   5   4   5   3   7   4   4     4   3   7     2
 0   4   5   5   4   5   3   7   4   6     4   3   7     2
 0   4   5   5   4   5   3   7   4   6     4   3   7     3
 0   6   5   5   4   5   3   7   4   6     4   3   7     3
 0   6   5   5   4   5   3   7   6   6     4   3   7     3
 0   6   5   5   4   5   3   7   6   6     4   5   7     3
 0   6   5   5   6   5   3   7   6   6     4   5   7     3
 0   6   5   5   6   5   5   7   6   6     4   5   7     3
 0   6   5   5   6   5   5   7   6   6     6   5   7     3
 0   6   5   5   6   5   5   7   6   6     6   5   7     3
 7   6   5   5   6   5   5   7   6   6     6   5   7     3
 7   6   5   5   6   5   5   7   6   6     6   5   7     4
```

6.3 Teste 3 - Verificação da correta formação de equipas quando os capitães são dois jogadores de campo

O terceiro teste que realizamos tem como finalidade averiguar o comportamento correto das entidades envolvidas na formação das equipas quando ambas têm como capitão um jogador de campo.

O que pretendemos analisar será:

- A correta transição de estados nos capitães e, para tal, é necessário que em duas das dez colunas que estão atribuídas à entidade jogadores de campo ocorra a mudança de estados de "0" para "2", tal como podemos observar no exemplo a seguir.
- A correta formação das equipas, ou seja, se em oito das dez colunas reservadas para os jogadores e em duas das três colunas reservadas para os guarda-redes ocorre a transição do estado "1" para o estado "3" ou para o estado "4" de acordo com a equipa onde cada jogador encontrar-se-á.

Ao realizarmos o teste, verificamos que o *script* desenvolvido atendeu às regras propostas.

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02  R01
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   0   0    0
 0   0   0   0   0   0   0   0   0   0    0   1   0    0
 0   0   0   0   0   0   0   0   0   0    0   1   0    1
 0   1   0   0   0   0   0   0   0   0    0   1   0    1
 0   1   0   1   0   0   0   0   0   0    0   1   0    1
 0   1   0   1   0   0   0   0   0   0    1   1   0    1
 0   1   0   1   0   0   0   1   0   0    1   1   0    1
 2   1   0   1   0   0   0   1   0   0    1   1   0    1
 2   1   0   1   0   0   0   1   0   1    1   1   0    1
 2   1   0   1   0   1   0   1   0   1    1   1   0    1
 2   3   0   1   0   1   0   1   0   1    1   1   0    1
 2   3   0   1   0   1   0   1   0   1    1   1   0    1
 2   3   0   3   0   1   0   1   0   1    1   1   0    1
 2   3   0   3   0   1   0   3   0   1    1   1   0    1
 2   3   0   3   0   1   0   3   0   1    1   3   0    1
 3   3   0   3   0   1   0   3   0   1    1   3   0    1
 3   3   0   3   0   1   0   3   0   1    1   3   7    1
 3   3   0   3   1   1   0   3   0   1    1   3   7    1
 3   3   2   3   1   1   0   3   0   1    1   3   7    1
 3   3   2   3   1   1   0   3   0   4    1   3   7    1
 3   3   2   3   1   4   0   3   0   4    1   3   7    1
 3   3   2   3   4   4   0   3   0   4    1   3   7    1
 3   3   2   3   4   4   0   3   0   4    4   3   7    1
 3   3   4   3   4   4   0   3   0   4    4   3   7    1
 3   3   4   3   4   4   0   3   0   4    4   3   7    2
 3   3   4   3   4   4   7   3   0   4    4   3   7    2
 3   5   4   3   4   4   7   3   0   4    4   3   7    2
 3   5   4   5   4   4   7   3   0   4    4   3   7    2
 3   5   4   5   4   4   7   3   0   4    4   5   7    2
 3   5   4   5   4   4   7   3   0   6    4   5   7    2
 3   5   4   5   4   6   7   3   0   6    4   5   7    2
 3   5   4   5   4   6   7   3   0   6    6   5   7    2
 3   5   4   5   4   6   7   3   0   6    6   5   7    3
 5   5   4   5   4   6   7   3   0   6    6   5   7    3
 5   5   6   5   4   6   7   3   0   6    6   5   7    3
 5   5   6   5   6   6   7   3   0   6    6   5   7    3
 5   5   6   5   6   6   7   5   0   6    6   5   7    3
 5   5   6   5   6   6   7   5   0   6    6   5   7    3
 5   5   6   5   6   6   7   5   7   6    6   5   7    3
 5   5   6   5   6   6   7   5   7   6    6   5   7    4
```

6.4 Teste 4 – Verificação do comportamento do árbitro ao longo da partida

O quarto teste que realizamos tem como finalidade averiguar o comportamento correto do árbitro ao longo da partida. Deste modo, o que pretendemos analisar será:

- Se o árbitro permanece no estado 1 até que as equipas se formem.
- Se o árbitro inicia o jogo apenas quando a formação das equipas está completa.
- Se, quando o árbitro termina o jogo, os jogadores da equipa 1 estão no estado 5 e os jogadores da equipa 2 estão no estado 6.

Ao realizarmos o teste, verificamos que o *script* desenvolvido atendeu às regras propostas: ao chegar, o árbitro transita do estado 0 para o estado 1 e permanece nesse estado até que as equipas sejam formadas (como as equipas já estavam formadas, não permanece muito tempo); quando as mesmas são formadas, ele muda o seu estado para 2, simbolizando que o jogo vai começar e, posteriormente, muda para o estado 3, sinalizando que está a arbitrar a partida; por último, quando transita para o estado 4, simbolizando que o jogo terminou, é possível verificar que todas as entidades da equipa 1 estão no estado 5 e todas as entidades da equipa 2 estão no estado 6.

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02   R01
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   0   0   0   0   0   0   0    0   0   0     0
  0   0   0   1   0   0   0   0   0   0    0   0   0     0
  0   0   0   1   0   0   0   0   0   0    0   0   0     0
  0   0   0   1   0   0   0   0   0   0    0   0   0     0
  0   0   0   1   0   0   0   0   0   0    0   0   0     0
  0   0   1   1   0   0   0   0   0   0    0   0   0     0
  0   0   1   1   0   0   0   0   0   0    0   0   0     0
  0   0   1   1   0   1   0   0   0   0    0   0   0     0
  0   0   1   1   0   1   1   0   0   0    0   0   0     0
  0   1   1   1   0   1   1   0   0   0    0   0   0     0
  0   1   1   1   0   1   1   0   0   0    0   0   0     0
  0   1   1   1   1   1   1   0   0   0    0   0   0     0
  0   1   1   1   1   1   1   0   0   0    0   0   0     0
  0   1   1   1   1   1   1   0   0   0    0   0   0     0
  0   1   1   1   1   1   1   0   0   0    0   0   0     0
  0   1   1   1   1   1   1   0   0   0    0   2   0     0
  0   1   1   1   1   1   1   0   0   0    1   2   0     0
  0   1   1   1   1   1   1   0   0   1    1   2   0     0
  0   1   1   1   1   1   1   0   0   1    1   2   0     1
  0   1   1   1   1   1   1   0   2   1    1   2   0     1
  0   1   1   3   1   1   1   0   2   1    1   2   0     1
  0   1   3   3   1   1   1   0   2   1    1   2   0     1
  0   1   3   3   1   3   1   0   2   1    1   2   0     1
  0   1   3   3   1   3   3   0   2   1    1   2   0     1
  0   1   3   3   1   3   3   0   2   1    1   3   0     1
  0   4   3   3   1   3   3   0   2   1    1   3   0     1
  0   4   3   3   1   3   3   7   2   1    1   3   0     1
  0   4   3   3   4   3   3   7   2   1    1   3   0     1
  0   4   3   3   4   3   3   7   2   4    1   3   0     1
  0   4   3   3   4   3   3   7   2   4    4   3   0     1
  0   4   3   3   4   3   3   7   2   4    4   3   7     1
  0   4   3   3   4   3   3   7   4   4    4   3   7     1
  0   4   3   3   4   3   3   7   4   4    4   3   7     2
  0   4   3   5   4   3   3   7   4   4    4   3   7     2
  0   4   5   5   4   3   3   7   4   4    4   3   7     2
  0   4   5   5   4   5   3   7   4   4    4   3   7     2
  0   4   5   5   4   5   3   7   4   6    4   3   7     2
  0   4   5   5   4   5   3   7   4   6    4   3   7     3
  0   6   5   5   4   5   3   7   4   6    4   3   7     3
  0   6   5   5   4   5   3   7   6   6    4   3   7     3
  0   6   5   5   4   5   3   7   6   6    4   5   7     3
  0   6   5   5   6   5   3   7   6   6    4   5   7     3
  0   6   5   5   6   5   5   7   6   6    4   5   7     3
  0   6   5   5   6   5   5   7   6   6    6   5   7     3
  0   6   5   5   6   5   5   7   6   6    6   5   7     3
  7   6   5   5   6   5   5   7   6   6    6   5   7     3
  7   6   5   5   6   5   5   7   6   6    6   5   7     4
```

6.5 Teste 5 - Verificação do *script* na globalidade

O quinto teste que realizamos divide-se em quatro partes e tem como finalidade averiguar o correto desempenho de cada *script* elaborado, tanto individualmente como num todo. Para tal, numa primeira fase verificamos se os três *scripts* realizados funcionam bem entre si; numa segunda fase verificamos se o *script* realizado para o árbitro funciona bem quando o executamos com os *scripts* do jogador e do guarda-redes previamente fornecidos; numa terceira fase verificamos se o *script* realizado para o jogador funciona bem quando o executamos com os *scripts* do árbitro e do guarda-redes previamente fornecidos; e numa quarta fase verificamos se o *script* realizado para o guarda-redes funciona bem quando o executamos com os *scripts* do árbitro e do jogador fornecidos previamente.

Fase 1: Note que todas as transições estão de acordo.
Fase 2: Note que todas as transições estão de acordo.
Fase 3: Note que todas as transições estão de acordo.
Fase 4: Note que todas as transições estão de acordo.

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02  R01
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  1   0   0   0   0   0   0   0   0   0    0   0   0    0
  1   0   0   0   0   0   0   0   0   0    0   0   0    0
  1   0   0   0   0   0   0   0   0   0    0   0   0    0
  1   0   1   0   0   0   0   0   0   0    0   0   0    0
  1   0   1   0   0   0   0   0   0   0    0   0   0    0
  1   0   1   0   1   0   0   0   0   0    0   0   0    0
  1   1   1   0   1   0   0   0   0   0    0   0   0    0
  1   1   1   0   1   0   1   0   0   0    0   0   0    0
  1   1   1   0   1   0   1   0   0   0    0   0   0    0
  1   1   1   0   1   0   1   0   0   0    0   0   0    0
  1   1   1   0   1   0   1   0   0   0    0   0   0    0
  1   1   1   0   1   0   1   0   0   1    0   0   0    0
  1   1   1   0   1   0   1   0   0   1    0   0   0    0
  1   1   1   0   1   0   1   0   0   1    2   0   0    0
  1   1   1   0   1   0   1   1   0   1    2   0   0    0
  1   1   1   0   1   0   1   1   0   1    2   0   0    0
  1   1   1   0   1   0   1   1   0   1    2   0   1    0
  3   1   1   0   1   0   1   1   0   1    2   0   1    0
  3   1   3   0   1   0   1   1   0   1    2   0   1    0
  3   1   3   0   3   0   1   1   0   1    2   0   1    0
  3   3   3   0   3   0   1   1   0   1    2   0   1    0
  3   3   3   0   3   0   1   1   0   1    3   0   1    0
  3   3   3   0   3   2   1   1   0   1    3   0   1    0
  3   3   3   0   3   2   1   1   0   1    3   7   1    0
  3   3   3   7   3   2   1   1   0   1    3   7   1    0
  3   3   3   7   3   2   4   1   0   1    3   7   1    0
  3   3   3   7   3   2   4   1   0   4    3   7   1    0
  3   3   3   7   3   2   4   4   0   4    3   7   1    0
  3   3   3   7   3   2   4   4   0   4    3   7   4    0
  3   3   3   7   3   4   4   4   0   4    3   7   4    0
  3   3   3   7   3   4   4   4   0   4    3   7   4    1
  3   3   3   7   3   4   4   4   0   4    3   7   4    2
  5   3   3   7   3   4   4   4   0   4    3   7   4    2
  5   3   3   7   5   4   4   4   0   4    3   7   4    2
  5   3   5   7   5   4   4   4   0   4    3   7   4    2
  5   3   5   7   5   4   4   4   0   4    3   7   4    3
  5   5   5   7   5   4   4   4   0   4    3   7   4    3
  5   5   5   7   5   4   4   4   0   6    3   7   4    3
  5   5   5   7   5   4   6   4   0   6    3   7   4    3
  5   5   5   7   5   4   6   6   0   6    3   7   4    3
  5   5   5   7   5   4   6   6   0   6    3   7   6    3
  5   5   5   7   5   4   6   6   0   6    5   7   6    3
  5   5   5   7   5   6   6   6   0   6    5   7   6    3
  5   5   5   7   5   6   6   6   0   6    5   7   6    4
  5   5   5   7   5   6   6   6   0   6    5   7   6    4
  5   5   5   7   5   6   6   6   7   6    5   7   6    4
```

Fase 1

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02  R01
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   0   1   0   0   0   0    0   0   0    0
  0   0   0   0   1   1   0   0   0   0    0   0   0    0
  0   0   0   0   1   1   0   1   0   0    0   0   0    0
  0   0   0   0   1   1   0   1   0   0    0   0   0    0
  0   0   0   0   1   1   0   1   0   0    0   0   0    0
  0   0   0   0   1   1   0   1   0   1    0   0   0    0
  0   0   0   0   1   1   0   1   1   1    0   0   0    0
  0   0   1   0   1   1   0   1   1   1    0   0   0    0
  0   1   1   0   1   1   0   1   1   1    0   0   0    0
  0   1   1   0   1   1   0   1   1   1    0   2   0    0
  0   1   1   1   1   1   0   1   1   1    0   2   0    0
  0   1   1   1   1   1   0   1   1   1    0   2   0    1
  7   1   1   1   1   1   0   1   1   1    0   2   0    1
  7   1   1   1   1   1   0   1   1   1    2   2   0    1
  7   1   1   1   1   3   0   1   1   1    2   2   0    1
  7   1   1   1   3   3   0   1   1   1    2   2   0    1
  7   1   1   1   3   3   0   3   1   1    2   2   0    1
  7   1   1   1   3   3   0   3   1   3    2   2   0    1
  7   1   1   1   3   3   7   3   1   3    2   2   0    1
  7   1   1   1   3   3   7   3   1   3    2   3   0    1
  7   1   1   1   3   3   7   3   4   3    2   3   0    1
  7   1   4   1   3   3   7   3   4   3    2   3   0    1
  7   4   4   1   3   3   7   3   4   3    2   3   0    1
  7   4   4   4   3   3   7   3   4   3    2   3   0    1
  7   4   4   4   3   3   7   3   4   3    2   3   7    1
  7   4   4   4   3   3   7   3   4   3    4   3   7    1
  7   4   4   4   3   3   7   3   4   3    4   3   7    2
  7   4   4   4   5   3   7   3   4   3    4   3   7    2
  7   4   4   4   5   5   7   3   4   3    4   3   7    2
  7   4   4   4   5   5   7   5   4   3    4   3   7    2
  7   4   4   4   5   5   7   5   4   5    4   3   7    2
  7   4   4   4   5   5   7   5   4   5    4   5   7    2
  7   4   4   4   5   5   7   5   6   5    4   5   7    2
  7   4   6   4   5   5   7   5   6   5    4   5   7    2
  7   6   6   4   5   5   7   5   6   5    4   5   7    2
  7   6   6   4   5   5   7   5   6   5    4   5   7    3
  7   6   6   6   5   5   7   5   6   5    4   5   7    3
  7   6   6   6   5   5   7   5   6   5    6   5   7    3
  7   6   6   6   5   5   7   5   6   5    6   5   7    4
```

Fase 2

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02   R01
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   0   0   0   0   0   0   0   0   0    0   0   0      0
 0   1   0   0   0   0   0   0   0   0    0   0   0      0
 0   1   1   0   0   0   0   0   0   0    0   0   0      0
 0   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   0   0   0   0   0   0    0   0   0      0
 1   1   1   0   1   0   0   0   0   0    0   0   0      0
 1   1   1   0   1   0   0   0   0   0    0   0   0      0
 1   1   1   0   1   0   0   0   0   0    0   0   0      0
 1   1   1   1   1   0   0   0   0   0    0   0   0      0
 1   1   1   1   1   0   1   0   0   0    0   0   0      0
 1   1   1   1   1   1   1   0   0   0    0   0   0      0
 1   1   1   1   1   1   1   0   0   0    0   0   2      0
 1   1   1   1   1   1   1   0   0   0    1   0   2      0
 1   1   1   1   1   1   1   0   0   0    1   7   2      0
 1   1   1   1   1   1   1   0   0   2    1   7   2      0
 1   1   1   1   1   1   1   7   0   2    1   7   2      0
 1   1   1   1   1   1   1   7   7   2    1   7   2      0
 1   1   1   1   1   1   1   7   7   2    1   7   2      1
 1   3   1   1   1   1   1   7   7   2    1   7   2      1
 1   3   3   1   1   1   1   7   7   2    1   7   2      1
 3   3   3   1   1   1   1   7   7   2    1   7   2      1
 3   3   3   1   3   1   1   7   7   2    1   7   2      1
 3   3   3   1   3   1   1   7   7   2    1   7   4      1
 3   3   3   4   3   1   1   7   7   2    1   7   4      1
 3   3   3   4   3   1   4   7   7   2    1   7   4      1
 3   3   3   4   3   4   4   7   7   2    1   7   4      1
 3   3   3   4   3   4   4   7   7   2    4   7   4      1
 3   3   3   4   3   4   4   7   7   4    4   7   4      1
 3   3   3   4   3   4   4   7   7   4    4   7   4      2
 3   5   3   4   3   4   4   7   7   4    4   7   4      2
 3   5   5   4   3   4   4   7   7   4    4   7   4      2
 5   5   5   4   3   4   4   7   7   4    4   7   4      2
 5   5   5   4   5   4   4   7   7   4    4   7   4      2
 5   5   5   4   5   4   4   7   7   4    4   7   6      2
 5   5   5   6   5   4   4   7   7   4    4   7   6      2
 5   5   5   6   5   4   4   7   7   4    4   7   6      3
 5   5   5   6   5   4   4   7   7   4    6   7   6      3
 5   5   5   6   5   4   4   7   7   6    6   7   6      3
 5   5   5   6   5   6   4   7   7   6    6   7   6      3
 5   5   5   6   5   6   6   7   7   6    6   7   6      3
```

Fase 3

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09  G00 G01 G02  R01
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   0   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   0   0   0   0   0   0   0    0   0   0    0
  0   0   1   1   0   0   0   0   0   0    0   0   0    0
  0   0   1   1   0   0   0   0   0   0    0   0   0    0
  0   0   1   1   0   0   0   0   0   0    0   0   0    0
  0   0   1   1   0   0   1   0   0   0    0   0   0    0
  0   0   1   1   0   0   1   0   0   0    0   0   0    0
  0   1   1   1   0   0   1   0   0   0    0   0   0    0
  0   1   1   1   0   1   1   0   0   0    0   0   0    0
  0   1   1   1   0   1   1   0   0   0    0   0   0    0
  0   1   1   1   0   1   1   0   1   0    0   0   0    0
  0   1   1   1   1   1   1   0   1   0    0   0   0    0
  1   1   1   1   1   1   1   0   1   0    0   0   0    0
  1   1   1   1   1   1   1   0   1   7    0   0   0    0
  1   1   1   1   1   1   1   0   1   7    0   0   0    1
  1   1   1   1   1   1   1   7   1   7    0   0   0    1
  1   1   1   1   1   1   1   7   1   7    0   2   0    1
  1   1   1   1   1   1   1   7   1   7    2   2   0    1
  1   1   1   1   1   1   1   7   1   7    2   2   7    1
  1   1   3   1   1   1   1   7   1   7    2   2   7    1
  1   1   3   3   1   1   1   7   1   7    2   2   7    1
  1   1   3   3   1   1   3   7   1   7    2   2   7    1
  1   3   3   3   1   1   3   7   1   7    2   2   7    1
  1   3   3   3   1   1   3   7   1   7    2   3   7    1
  1   3   3   3   1   4   3   7   1   7    2   3   7    1
  1   3   3   3   1   4   3   7   4   7    2   3   7    1
  1   3   3   3   4   4   3   7   4   7    2   3   7    1
  4   3   3   3   4   4   3   7   4   7    2   3   7    1
  4   3   3   3   4   4   3   7   4   7    4   3   7    1
  4   3   3   3   4   4   3   7   4   7    4   3   7    2
  4   3   5   3   4   4   3   7   4   7    4   3   7    2
  4   3   5   5   4   4   3   7   4   7    4   3   7    2
  4   3   5   5   4   4   5   7   4   7    4   3   7    2
  4   3   5   5   4   4   5   7   4   7    4   5   7    2
  4   3   5   5   4   6   5   7   4   7    4   5   7    2
  4   3   5   5   4   6   5   7   6   7    4   5   7    2
  4   3   5   5   6   6   5   7   6   7    4   5   7    2
  4   3   5   5   6   6   5   7   6   7    4   5   7    3
  4   3   5   5   6   6   5   7   6   7    6   5   7    3
  6   3   5   5   6   6   5   7   6   7    6   5   7    3
  6   3   5   5   6   6   5   7   6   7    6   5   7    4
```

Fase 4