3. Game class

This chapter describes the class of games that can be deployed on BGE. It shows common features among these games and introduces a general framework for n-player round-based games.

3.1 Common features

All business games that are designed to run on BGE share the following characteristics of which some of them are based on the taxonomy of Greco et al. [GBN13].

Browser-based. The game runs in a web browser and includes texts and charts. It is played over a computer network (LAN, Internet).

Single/multiplayer. The game is played by one or more players. Each game defines an interval assigning the minimal and maximal number of players. Most BGs will be designed as multiplayer games where at least two opponents compete against each other. Virtual opponents can be integrated with little effort. This permits a single player to play a multiplayer game against virtual opponents or to simulate the game having virtual opponents compete against each other.

Round-based. The game is round-based in contrast to e.g. a flight simulator where interventions within the simulation are almost continuous. Each round (period), the players take decisions based on the game history and independent from the other players' decision of this round. Every player must enter his or her decision for the next period within a certain time. The time frame to take a decision is synchronized between all parties which means the players act simultaneously. If one party does not respond within the time frame it is removed from the game. Thus all parties have the same time to reflect and take a decision. In contrast to alternating rounds, e.g. chess, where one player decides after the other, decisions in round-based business games are met simultaneously.

Indirect user interaction. Interaction between the players is indirect since they submit their decisions independently from each other. There are no means of direct interactions, e.g. an integrated communication channel. It is open to the game moderator respectively the intention of the game session in question whether communication between players is prohibited or not. Teams other than several users playing in front of one computer are not supported.

Player evaluation. The goal of the game is not absolute (fixed score on a scale) but relative to the opponents or to previous sessions of the game. In most BGs the players will be evaluated individually. A collective score evaluating the performance of the players altogether can be implemented as well.

Data collection. The complete game data is stored for later analysis of the players' behaviour and strategies.

3.2 N-player round-based games

This section introduces a general framework for n-player round-based games by utilizing inputs, outputs, fixed parameters, state transition functions, a score function, and a stop condition. The framework is then applied on the round-based game *stone-paper-scissors*.

3.2.1 General framework

We define a n-player round-based game through the 7-tuple $(In, P, Out, S_0, \mathcal{F}, score, sc)$. The index $t \in T$ represents one round from round 1 to round T. We call a round also turn or period.

Inputs. $In = \begin{pmatrix} in_1 \\ .. \\ in_l \end{pmatrix}$ is a vector of input variables. An input variable is also a vector with an entry for every player n. We assume the number of input variables l constant in t, which means a game has the same inputs in every round. The domain DIn_i of the input variable i with $in_i \in DIn_i, i \in \{1, \ldots, l\}$ has to be chosen depending on the game, generally, $\mathbb{R}^n$ can be a good choice.

In each round t, n players or virtual opponents submit l input values $in_{ij}^{(t)}, i \in \{1, \ldots, l\}$, $j \in \{1, \ldots, n\}, t \in \{1, \ldots, T\}$ which results in the input matrix $In^{(t)}$ with

$$In^{(t)} = \begin{pmatrix} in_{11}^{(t)} & .. & in_{1n}^{(t)} \\ .. & .. & .. \\ in_{l1}^{(t)} & .. & in_{ln}^{(t)} \end{pmatrix} \in DIn_1 \times \cdots \times DIn_l. \tag{3.1}$$

Parameters. P is a set of fixed parameters (constants), e.g. a pre-defined number of rounds, a minimal/maximal number of players, a timeout to take a decision or an economic constant.

Outputs. $Out = \begin{pmatrix} out_1 \\ .. \\ out_k \end{pmatrix}$ is a vector of output variables. As with the inputs, we assume the number of outputs k constant in t. Output variables resemble input variables. They are a vector with length n and their domain $DOut_i, i \in \{1, \ldots, k\}$ is game-dependent.

The outcome of a round or rather the game state in t is described by the output matrix $Out^{(t)}$, which sums up the realization of the output variables in t. It is

$$Out^{(t)} = \begin{pmatrix} out_{11}^{(t)} & .. & out_{1n}^{(t)} \\ .. & .. & .. \\ out_{k1}^{(t)} & .. & out_{kn}^{(t)} \end{pmatrix} \in DOut_1 \times \cdots \times DOut_k. \tag{3.2}$$

Initial game state. The initial game state S_0 defines the state of the game at $t \leq 0$ before it has been started. It is fully described by the initial state of every output variable and thus

$$S_0 = Out^{(0)}. \tag{3.3}$$

State transition functions. $\mathcal{F}$ is a set of functions transforming the game state from turn t to $t+1$. Hence, these functions are used to calculate the game's outputs of a round t from the parameters, initial game state and game history. The latter is defined as $HISTORY := I^{(t)} \times I^{(t-1)} \times \cdots \times I^{(1)} \times S_0$. It is $|\mathcal{F}| = k$ which means there exists exactly one transition function per output variable and $f_i \in \mathcal{F}, i \in \{1, \ldots, k\}$ is used to calculate the output variable out_i. Each f_i is defined as

$$f_i \colon P \times HISTORY \times playerIDs \to DOut_{i\_pID}, \tag{3.4}$$

with $playerIDs := \{1, \ldots, n\}$ $and\_pID \in playerIDs$ being the number of the current player in question for which the output is calculated.

Score function. A player's result or score of a round is calculated with the help of the *score* function. It is

$$score \colon P \times Out^{(t)} \times playerIDs \to \mathbb{R}, \tag{3.5}$$

with $playerIDs$ as defined before.

In each round the score function is applied for every player and the results are written to a variable $roundScore^{(t)}$ which is a vector with length n just as an input or output variable. The result over all so-far played periods is defined as

$$accumulatedScore^{(t)} = \sum_{s=1}^{t} roundScore^{(s)}. \tag{3.6}$$

It is possible to define a collective score function with

$$collectiveScore \colon P \times Out^{(t)} \to \mathbb{R} \tag{3.7}$$

which assesses the performance of the players altogether for a period t.

Stop condition. *sc* is a stop condition. If this condition is met, the game ends. A stop condition can be e.g. a pre-defined number of turns.

Please note that in many games the players do not receive complete information about all game describing variables. Additionally, P, $\mathcal{F}$, *score* and *sc* can be generalized and defined dependent on t as well, e.g. in Skat $\mathcal{F}$ is defined differently when playing "Null". Furthermore, all tuple parts could also be made dependent on the number of players n.

3.2.2 Applying the framework

This subsection provides an easy example for the application of the framework by applying it on the the round-based game *stone-paper-scissors*.

Initialization. Utilizing the framework introduced above stone-paper-scissors can be described as following:

- $In = \big(choice\big)$. In stone-paper-scissors the players solely have to take one decision in each round and decide between one of the three options. Therefore, there is only **one input variable**. We call it *choice* and define it as $choice \in \{stone, paper, scissors\}^n$.
- $P = \{maxrounds, minplayers, maxplayers, timeout\}$. The rules of stone-paper-scissors are very simple and there are no game specific **parameters** as e.g. economic constants. However, every game needs to agree on the number of rounds to be played, the minimal and maximal number of players who can participate in the game as well as the maximal time to take a decision. Since stone-paper-scissors is a 2-player game consisting of three rounds we get $minplayers = 2$, $maxplayers = 2$ and $maxrounds = 3$.
- $Out = \big(roundWinner\big)$. The only output in stone-paper-scissors is the information about winner and loser or if the round ended in a draw. That being the case, there is only **one output variable** we call *roundWinner*. It is defined as $roundWinner \in \{-1, 0, 1\}^n$ where -1 stands for defeat, 0 for tie and 1 for victory. The initial state S_0 of the game is $Out^{(0)} = \big(0 \quad 0\big)$.
- $\mathcal{F} = \{calculateRoundWinner\}$. One output to calculate results in **one state transition function**. This function calculates the output variable *roundWinner* for a player i and a round t from the input choices of both players as shown in algorithm 1. With $calculateRoundWinner\colon I^{(t)} \times \{1, 2\} \to \{-1, 0, 1\}$, the definition of the function is quite simple; it only needs the current inputs in order to determine the new game state. In more sophisticated games the transition functions are more complex accessing the complete game history and P as shown in equitation 3.4.
- The **score function** of stone-paper-scissors is also very simple. The score is 1,-1 or 0 for victory, defeat or tie and thus it just returns the result of the *roundWinner* output variable of the current round. In other games the score function e.g. combines several outputs and weights them.
- The **stop condition** checks whether the current round t is greater than the pre-defined number of rounds *maxrounds*. Consequently it is $sc : t > 3$.

First round. Assuming player one chooses *stone* and player two chooses *paper*. This results in the input matrix $In^{(1)} = \big(stone \quad paper\big) = \big(0 \quad 2\big)$.

Applying the state transition function *calculateRoundWinner* as shown in algorithm 1 for both players in order to transfer the game state to $t = 1$ we get the output matrix $Out^{(1)} = \big(-1 \quad 1\big)$. The score function returns the same values which leads to $roundScore^{(1)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $accumulatedScore^{(1)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

Algorithm 1

```
1: function CALCULATEROUNDWINNER(playerID)
2:     choice1 ← get choice of player one for round t      ▷ stone=0, scissors=1, paper=2
3:     choice2 ← get choice of player two for round t
4:     if playerID = 1 then
5:         diff ← choice1 − choice2
6:     else if playerID = 2 then
7:         diff ← choice2 − choice1
8:     end if
9:     if diff = −1 or diff = 2 then
10:        result ← 1
11:    else if diff = 0 then
12:        result ← 0
13:    else
14:        result ← −1
15:    end if
16:    return result
17: end function
```

Second round. We assume $In^{(2)} = \begin{pmatrix} scissors & scissors \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix}$. For $t = 2$ we get $Out^{(2)} = \begin{pmatrix} 0 & 0 \end{pmatrix}$ which gives $roundScore^{(2)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $accumulatedScore^{(2)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

Last round. With $In^{(3)} = \begin{pmatrix} paper & scissors \end{pmatrix} = \begin{pmatrix} 2 & 1 \end{pmatrix}$ it is $Out^{(3)} = \begin{pmatrix} -1 & 1 \end{pmatrix}$, $roundScore^{(3)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $accumulatedScore^{(3)} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$. The game ends because the stop condition is met. Player two wins the game with a total score of $2 > -2$.