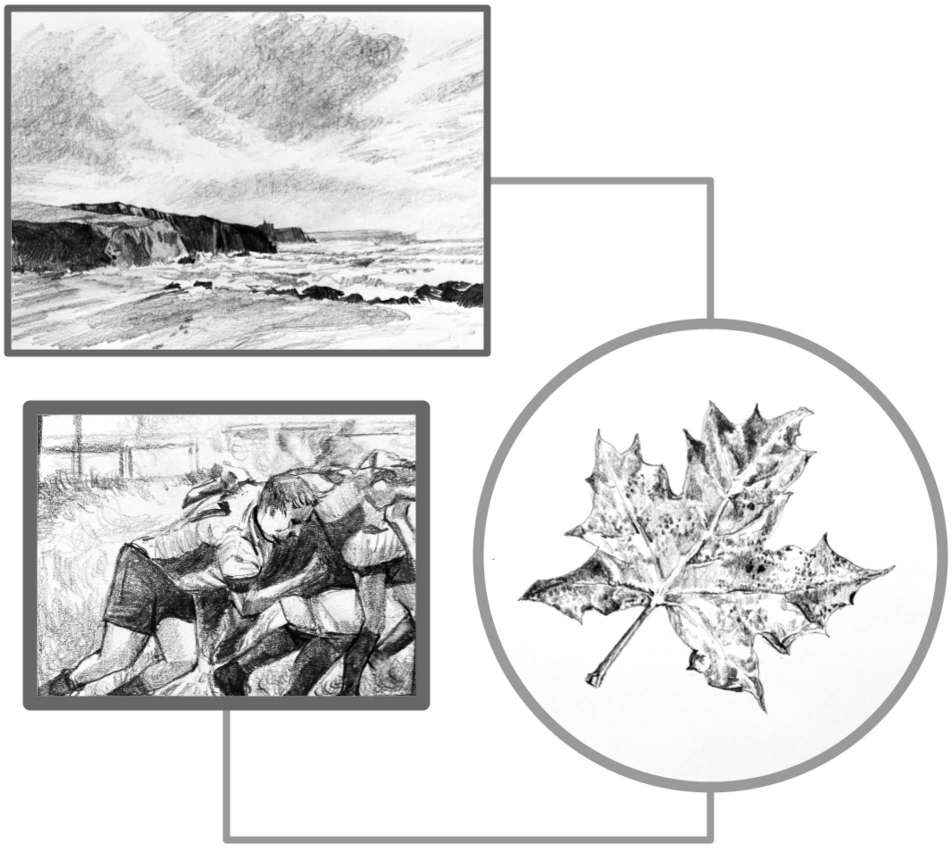

11

Managing a rugby match

Concept. Building a toy model simulation of a rugby match whose outcome can be manipulated through correctly-timed player substitutions and game management decisions. The dexetera state manipulation framework we have built around the stochadex can meet these requirements, and a dashboard can be created for user interaction. All this combines together to make a simple dashboard game, which we call: 'trywizard'. For the mathematically-inclined, this chapter will motivate the construction of a specific modelling framework for rugby match simulation. For the programmers, the public Git repository for the code described in this chapter can be found here: https://github.com/umbralcalc/trywizard.

11.1 Designing the event simulation engine

Since the basic state manipulation framework and simulation engine will run using dexetera, the mathematical novelties in this project are all in the design of the rugby match model itself. And, as ever, we're not especially keen on spending a lot of time doing detailed data analysis to come up with the most realistic values for the parameters that are dreamed up here. Even though this would also be interesting.[1]

Let's begin by specifying an appropriate state space to live in when simulating a rugby match. It is important at this level that events are defined in quite broadly applicable terms, as it will define the state space available to our stochastic sampler and hence the simulated game will never be allowed to exist outside of it. It seems reasonable to characterise a rugby union match by the following set of states: Penalty, Free Kick (the punitive states); Penalty Goal, Drop Goal, Try (the scoring states); Kick Phase, Run Phase, Knock-on, Scrum, Lineout, Maul and Ruck (the general play states). Using this set of states, in Fig. 11.1 we have summarised our approach to match state transitions into a single event graph. In order to capture the fully detailed range of events that are possible in a real-world match, we've needed to be a little imaginative in how we define the

[1] One could do this data analysis, for instance, by scraping player-level performance data from one of the excellent websites that collect live commentary data such as rugbypass.com or espn.co.uk/rugby.

kinds of state transitions which occur. In other words; it's fair to say that our simplified model here represents just a subset of states that a real rugby match could exist in.

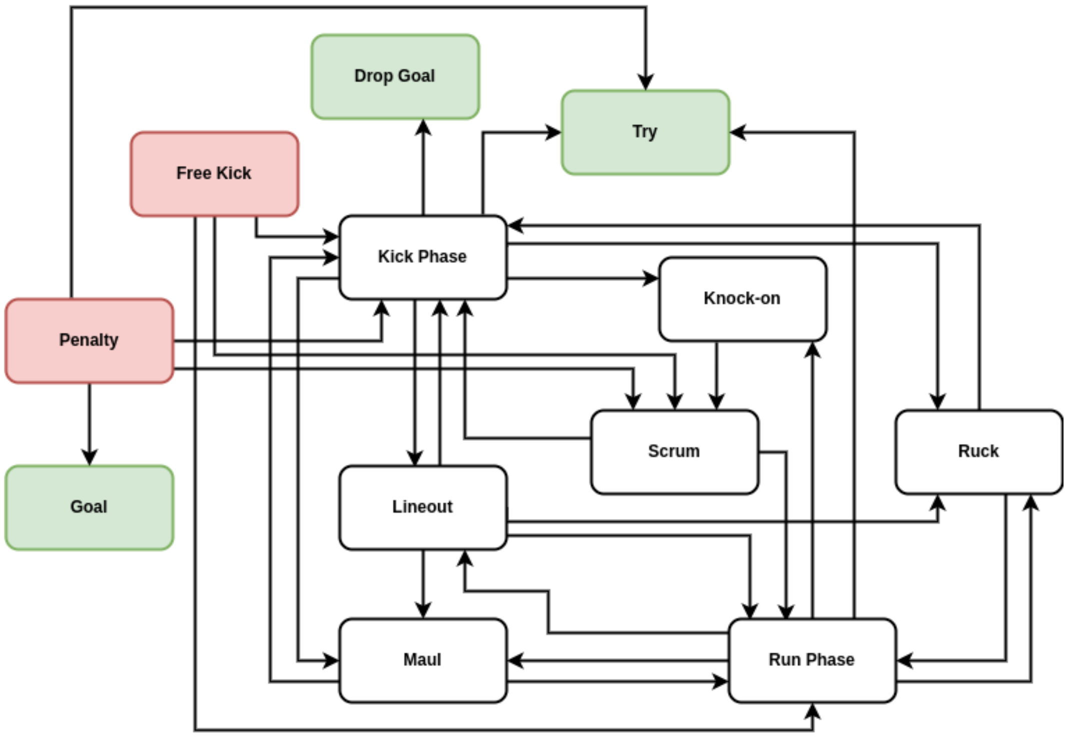

Figure 11.1: Simplified event graph of a rugby union match.

In addition to occupying some state in the event graph, the state of a rugby match must also include a binary 'possession' element which encodes which team has the ball at any moment. We should also include the 2-dimensional pitch location of the ball as an element of the match state in order to get a better sense of how likely some state transitions are, e.g., when playing on the edge of the pitch near the touchline it's clearly more likely that a Run Phase is going to result in a Lineout than if the state is currently in the centre of the pitch. To add even more detail, in the next section we will elaborate further by introducing states for each playing position on each side, which encode the playing abilities of each player, their fatigue status and their substitution status. The latter of these keeps track of whether or not the player has been injured or yellow/red-carded during the course of a game and should also enable management strategies to become more nuanced.

Since a rugby match exists in continuous time, it is natural to choose a continuous-time event-based simulation model for our game engine. As we have discussed in previous chapters already, this means we will be characterising transition probabilities of the event graph in Fig. 11.1 by ratios of event rates in time. Recalling our notation in previous chapters, if we consider the current state vector of the match X_{t}, we can start by assigning each transition $a \rightarrow b$ on the event graph an associated rate of occurance $\lambda_{a \rightarrow b}(X_{\mathsf{t}}, \mathsf{t})$ which is defined in units of continuous time, e.g., seconds. In addition to the transitions displayed on the graph, we can add a 'possession change transition';

where the possession of the ball in play moves to the opposing team. This transition may occur while the match is also in any of the white-coloured states on the graph and let's assign this a time and state-dependent expected rate of occurance $\lambda_{\rm pos}(X_{\sf t}, {\sf t})$.

Based on our dicussion above, an appropriate encoding for the overall game state at timestep index $\sf t$ could be a state vector $X_{\sf t}$ whose elements are

$$X^0_{\sf t} = \begin{cases} 0 & \text{Match State} = \textsf{Penalty} \\ 1 & \text{Match State} = \textsf{Free Kick} \\ \dots \end{cases} \tag{11.1}$$

$$X^1_{\sf t} = \begin{cases} 0 & \text{Possession} = \textsf{Home Team} \\ 1 & \text{Possession} = \textsf{Away Team} \end{cases} . \tag{11.2}$$

But how does this overall game state connect to the event rates? The probabilistic answer is quite straightforward. If the probability of the match state being $X^0_{\sf t} = a$ at timestep $\sf t$ is written as $P^0_{\sf t}(a)$, then the probability of $X^0_{\sf t+1} = b$ in the following timestep is

$$P^0_{\sf t+1}(b) = \frac{\lambda_{a\rightarrow b}(X_{\sf t}, {\sf t})P^0_{\sf t}(a)}{\left[\frac{1}{\tau} + \sum_{\forall b}\lambda_{a\rightarrow b}(X_{\sf t}, {\sf t})\right]P^0_{\sf t}(a)} , \tag{11.3}$$

where $\forall b$ in the summation indicates that all the available transitions from a should be summed over. Note that, in the expression above, we have also defined τ as a timescale short enough such that no transition is likely to occur during that interval.

Before we move on to other details, it's quite important to recognise that because our process is defined in continuous time, the event rates may well vary continuously (this will be especially true when we talk about, e.g., player fatigue). Hence, Eq. (11.3) is only an *approximation* of the true underlying dynamics that we are trying to simulate — and this approximation will only be accurate if τ is small. The reader may recall that we discussed this same issue from the point of view of simulating time-inhomogeneous Poisson processes with the rejection method when we were building the stochadex in an earlier chapter.

An equivalent to Eq. (11.3) should also apply to the possession change transition rate, i.e., the probability that the Home Team has possession $P^1_{\sf t}$ at time $\sf t$ evolves according to

$$P^1_{\sf t+1} = \frac{\lambda_{\rm pos}(X_{\sf t}, {\sf t})(1-P^1_{\sf t})}{\left[\frac{1}{\tau} + \lambda_{\rm pos}(X_{\sf t}, {\sf t})\right](1-P^1_{\sf t})} . \tag{11.4}$$

While these match state transitions and possession changes are taking place, we also need to come up with a model for how the ball location $L_{\sf t}$ changes during the course of a game, and as a function of the current game state. If we associate every state on the event graph with a single change in spatial location of the ball on the pitch, we then need to construct a process which makes 'jumps' in 2-dimensional space each time a state transition occurs. For the sake of simplicity, we will choose this model to just be a 2-dimensional Gaussian distribution such that the probability density $P_{\sf t}(\ell)$ of $L_{\sf t} = \ell$ at timestep $\sf t$ evolves according to

$$P_{\sf t+1}(\ell) = \textsf{MultivariateNormalPDF}[\ell; L_{\sf t}, C(X_{\sf t}, {\sf t})] , \tag{11.5}$$

where $C(X_{\sf t}, {\sf t})$ is some spatial covariance matrix which depends on the current state and timestep. Note that because $L_{\sf t}$ is a part of the overall game state, it will be included as information contained within some elements of X_{t} as well. To make this explicit, we can simply set $X^2_{\mathsf{t}} = L^{\rm lon}_{\mathsf{t}}$ and $X^3_{\mathsf{t}} = L^{\rm lat}_{\mathsf{t}}$ — where $(L^{\rm lon}_{\mathsf{t}}, L^{\rm lat}_{\mathsf{t}})$ denote the longitudinal (lengthwise along the pitch) and lateral (widthwise across the pitch) components of the ball location, respectively.

11.2 Associating events to player states and abilities

In the last section we introduced a continuous-time event-based simulation model for a rugby union match. In this section we are going to add more detail into this model by inventing how to associate specific player states and abilities to the event rates of the simulation. Before continuing, we want to reiterate that this model is entirely made up and, while we hope it illustrates some interesting mathematical modelling ideas in the context of rugby, there's no particular reason why a statistical inference with a reliable dataset should prefer our model to others which may exist. In other words, this toy model is just for fun!

11.3 Deciding on gameplay actions

- Need to start thinking here about the fundamental dexetera formalism before fleshing this bit out.

11.4 Writing the game itself

- Show which stochadex/dexetera methods were called and how they were used to simulate the game.
- Give a summary of how the dashboard backend works (diagram would help) and how this connects up to the streamlit frontend via protobuf messages.

Bibliography