The concept of a language game, as it is described here, was initially contemplated by Ludwig Wittgenstein. His language games are a series of thought experiments, in which language is fragmented into relatively simple components.

One of the most basic examples of a game involves two builders; call them Alex and Billie. Alex issues commands in the form of holding up a certain number of fingers. Billie interprets such a command as an instruction to fetch the relevant number of bricks. From this very simple start, Wittgenstein proceeds to add more features to the rudimentary language, such as finite and infinite numeral systems, distinctions between different forms of brick, and directions as to where Billie needs to go. In so doing he attempts to break down a highly evolved and highly complex biological system into smaller, more manageable chunks.

In my research, language games take on a slightly different role; rather than being thought experiments, they serve to test a particular language model that I study. The model focuses on the way concepts are used to describe the world around us. The essence of the conceptual approach is that although the physical properties we observe may vary continuously, we use a finite set of labels to describe them.

As a simple example, take temperature. Temperature is a physical quantity which varies in one dimension, and which we have the capability to measure to some degree of accuracy. When we talk about, say, the temperature of the sea, we use terms like “cold”, “cool”, or “warm”. Each person’s notion of what is and isn’t cool may vary, though probably not so drastically that anyone would describe 30ºC as cool. For myself and my colleagues, someone’s concept of “cool” is a range of temperatures.

Now, say we want to create a conceptual language that robots could use to communicate about temperature. We would like the language to be flexible, so that it can be adapted to the particular range of temperatures the robots encounter. Thus, if they lived in an environment where temperatures ranged from 0ºC to 30ºC, their concept of “cold” would be different than if they generally experienced temperatures between -30ºC and 0ºC.

We would also want the robots to agree on their concepts, to a great extent; communication will not
prove very useful otherwise. This goal could be achieved by allowing the robots some period during which they learn from each other. Here is where the language games come in to play.

Imagine two robots – call them Sasha and Leslie – initially have different concepts of “cool”. They are both introduced to a body of water, and they measure the temperature. Sasha says to Leslie, “This is cool”. Leslie would not have described the water as “cool”, but chooses to accept Sasha’s definition. Leslie’s concept of “cool” is thus updated.

This language game – call it the “speaker-listener” game – and several others owe much to the work of Luc Steels. Since they are discussed in relation to artificial intelligence, they go beyond the thought experiments of Wittgenstein to serve as actual training scenarios. I myself simulate the process. In a simulation, a population of agents (robots) start off with randomised concepts, and then interact through a series of speaker-listener games. The quality of the conceptual language model in question is tested by how quickly the concepts of the agents converge (meaning that they come to be approximately equal).

Other BCCS students have already worked extensively in this way, developing our language model in the process. For my own research, one of the things I am aiming to develop is how language games can be adapted such that agents could simultaneously consider separate features of some object, such as colour and texture, e.g. “The chair is green and smooth”.

Collective motion is one of the most striking examples of complex behaviour and is currently a very active area of research. Fish schools, bird flocks and countless other examples of collective behaviour offer real-world demonstrations of self-organisation in the presence of decentralised control.

When animals move collectively, we can try to explain their behaviour in terms of an interaction mechanism, a set of hypothetical rules that govern how individuals respond to one another. The application of these rules depends on the position of an individual’s neighbours relative to an interaction field. For example, “run away” is a perfectly good interaction mechanism and “from anything closer than 5m” defines a reasonable field, although this particular animal is not going to make many friends.

The success of collective motion research over the last two decades has been the development of simple interaction mechanisms that, when simulated, generate global collective behaviour similar to those found in nature. Of course, the success of a particular interaction mechanism in reproducing collective behaviour in silico does not imply that this is the mechanism that operates in nature. The goal of my research is to find and understand the interaction mechanisms that are actually being employed by animals in the real world.

In the past, researchers have tended to examine collective behaviour involving many individual animals. However, the large numbers of individuals makes it difficult to determine which animals are interacting with each other, let alone how they interact. My research focuses not on a large number of animals moving collectively, but instead on animals moving in pairs, which simplifies the range of possible interactions and removes any doubt about who is interacting with whom. As part of this research, we have recently examined how pairs of bats interact with each other.

Many species of bats live in large colonies in caves. These colonies can consist of millions of individuals, which all leave the cave around dusk to feed. During this exodus, the bats fly in dense streams, where it appears that collisions between bats are inevitable. However, the bats are consistently able to avoid each other.

While we ultimately hope to be able to reverse engineer the interaction mechanisms that bats use to achieve this, my work so far has centred around processing a large dataset of trajectories, recorded as pairs of bats fly together. This dataset includes many trajectories where no interaction is evident, as well as examples of several different behaviours. In order to categorise this dataset, we have created a


Thomas McKetterick won the award for best speaker at BritBats II at Bristol Zoo for “Cooperative Navigation in Bats and the Role of Sonar Repulsion”, Thomas John McKetterick, Luca Giuggioli, Marc Holderied.


Filtering movement behaviour in this way has allowed us to uncover some key insights into the behaviour of bats. For example, the Daubenton’s bats studied in this work were found to interact such that they either fly in synchronisation or chase each other. In both cases the bats are well aligned with each other, which greatly reduces the danger of colliding.

The chase-flight and sync-flight behaviours of the Daubenton’s bats were found mathematically, without the need to look through the dataset visually. Using this approach of filtering pair-wise movement data for relevant behaviour, we have been able to identify where and in what form interactions take place. Such work will be crucial if we are ever to realise our goal of discovering the natural mechanisms behind collective motion.

Example trajectories depicting sync-flight (top left), chase-flight (bottom left) and a null interaction (right).

Understanding the pairwise interactions of bats should help us one day crack the puzzle of collective motion.

Photograph taken by Hans Hillewaert.
Liar’s Dice

Neeraj Oak discusses the recent “Liar’s Dice” tournament, pitting algorithms against humans in a game of probability and deception.

Are you a despicable individual? Do you enjoy deceit, intrigue and trickery? Can you count? If so, dear reader, you have the makings of a fine Liar’s Dice player.

Liar’s Dice is thought to have its roots in South America; evidence is scant, but many believe it to originate in a game played widely in the Inca Empire, and exported to Europe by the notorious conquistador Francisco Pizarro.

Each player begins with 5 dice and a cup. At the start of a round, all players shake their dice, concealed in the cup, and place them down simultaneously. Each player may then peek at their dice under their cup, but may not view the dice of others. The first player begins by making a guess as to how many dice amongst all the players show a certain number: for example, a valid bid could be “I believe there are at least four dice showing the number 3”. Subsequent players may choose to make higher bids, or declare that the preceding player was bluffing. When a bluff call is made, all players reveal their dice, which are counted to determine the veracity of the previous bid. If a player is shown to have been bluffing, that player loses a dice. However, if the player was not bluffing, the opponent who called ‘bluff’ loses a dice. When a player loses all their dice, they are eliminated. Play continues clockwise around a table until all but one of the players are eliminated; the surviving player is the winner.

With rules that fit neatly into a single paragraph, Liar’s Dice is an easy game to learn but a difficult one to master. It’s been a popular pastime for BCCS students over this past year, often played at the pub or on a lazy summer afternoon. So it wasn’t surprising when, between pints, conversation turned to an important question: could a computer algorithm be written that plays Liar’s Dice better than a human can? And thus was born the 2013 Liar’s Dice tournament…

Entrants to the tournament were challenged to write code, in any language they preferred, that could play Liar’s Dice. These computer players would be pitted against humans and other computers around a table, with the winner receiving a bottle of wine.

It quickly became apparent that the strength of a computer is in its ability to crunch numbers; making probability calculations for a large number of dice is difficult for humans without significant training, but is a trivial task for an algorithm. The great weakness in a computer is its inability to spot lies. Human beings have a great many methods of spotting a lie: a wavering voice, or one made suddenly and incongruously bold; avoidance of eye contact, or a fixed stare; even a mischievous glint in the eye of an opponent can give an indication of their action. Computers can rely on none of these things, and must operate solely on the statistical data available to them. This makes them excellent at the start of a game, when there are a large number of players and dice, but increasingly vulnerable as the game goes on and the number of dice dwindles. At this point, a good human liar will have the advantage.

The victor (i.e. liar-in-chief) of our first tournament was BCCS’s own Tom Todd, valiantly beating off competition from 6 human and 2 computer opponents. I only hope the bottle of wine makes up for the distrust with which we all now view him. Computers, as predicted, began the game as formidable opponents, but their performance waned as the game wore on. Lessons were learnt though, and with more computer entrants being put forward for the next tournament, it seems only a matter of time before we have our first computerised victor. Computers that lie - a thought that sends a shiver down the spine of Terminator fans everywhere…

If you’d like to get involved in future tournaments, either as human cannon-fodder or through a computer bot, do get in touch at Neeraj.Oak@Bristol.ac.uk.