**Description of the Creature Simulation**

This simulation is a discrete event simulation which aims to simulate a creature crossing the highway. The highway traffic is modelled after the Nagel – Schreckenberg approach which divides the highway into cells that correspond to 7.5m of highway. The simulation also supports multiple lanes. At each time step in the simulation, for each lane, a new car may be generated with a probability specified in the configuration file as ENTRY\_CAR\_PROB. If there is already a car in the first lane because it hasn’t sped up enough, or traffic is congested, it is added to a queue of cars waiting to enter the highway. The entrance point is always lane zero. Cars move according to the Nagel – Schreckenberg model. That is to say, they accelerate by one until they reach their maximum speed which is specified in the configuration file as MAX\_SPEED. With the assumption that the cells are 7.5m, a maximum speed of 10 corresponds to 99km/h. If a car encounters another car in front of it, it slows down to match the speed to avoid a collision. At this time, the simulation also supports the idea of random deceleration of cars (as specified in the Nagel-Schrekenberg paper), and can be turned on or off by setting RANDOM\_DECEL to TRUE in the configuration file. So far we have not experimented with this, as we have kept the experiments simple so far, but it is possible in the future.

The creatures are implemented similar to the cars. They also use a queue so that if a creature has not yet crossed, the new creatures line up behind it. The creatures are generated with a probability of ENTRY\_CREATURE\_PROB at each time step, and at each cross point. Cross points are specified in the configuration file by CROSS\_POINT <int> and can be repeated as many times as there are places for the creature to cross. When a creature crosses, it does so one lane at a time. In a single timestep, the creature looks at the environment (where cars are, and what speed they are travelling) then decides to move. If it decides to move, it moves onto the highway, then the cars move. If a car moves into the cell the creature is occupying the creature is hit. In order for a creature to decide whether to move, it must consult the global creature “knowledgebase”. This knowledgebase is a table of states and results. The table has “fuzzy” categories for speeds such as “fast” “medium” and “slow”, and proximities such as “close” “med” “far”. The values are all set in the configuration file for “fast” “medium” “slow” “close” and “far”. As a creature attempts to cross under one of these conditions, if a creature is hit, a negative result is recorded. If the creature successfully crosses a positive result is recorded. In a multi-lane highway, the result is propagated to all of the lane – condition pairs the creature encountered on it’s trip across the lanes. There is also an option to have the creature cross the entire highway at once, however this is not implemented yet.

There are several algorithms available for the creature to cross the road and the simulator was designed in such a way that future algorithms should be easy to add. The algorithm has the knowledgebase available to it, as well as the “fear” and “desire” of the creature. When a creature is created, its fear and desire are both random between 0 and 1. The algorithm can use this information so that creatures which are fearless and have motivation such as food on the other side are more likely to cross in risky situations.

The first naiive algorithm ignores fear and desire completely and simply crosses if it is safe. If there is no data in the knowledgebase yet for a particular cross point the creature simply tries to cross. As soon as a negative result occurs for a particular cross point, it is avoided in the future.

The second naiive algorithm includes fear and desire. If the successratio + desire – fear > 0 the creature will cross for the particular crosspoint. The successratio is defined as the (result / crossed). This means that as a knowledge base is built up, this ratio will become highly biased towards previous results since result is the number of success or failures in the knowledgebase of the creature and not just in a particular experiment.

In the first fuzzy algorithm, there are three rules. The first rule is “If far or slow then cross is likely”. The membership functions for each of these are both stair-step functions where the slope is determined by close, mid and far specified by the user in the configuration file. The second rule is “If !fear and desire then cross is likely”. The last rule is simply the success ratio. The three rules are combined by finding the max of the three rules. This will give a result between 0 and 1. A random number is picked between 0 and 1 as well and if the number is <= to the fuzz result then the creature should move.