An Architecture For an Intelligent Sheep-dog In POP11

"It's sheep we're up against"
The Housemartins

Name: Tom Carter

Course: MSc Cognitive Science

Summer Project

Submission Date: 02/09/99

Approximately 12,000 words

ABSTRACT

Peter Waudby's 1996 Summer Project was a simulation of a sheep dog trial programmed in Pop11 using rule-based techniques and the sim-agent toolkit to provide an intelligent architecture for a flock of sheep. It included a user controlled sheep dog which could be used to herd the sheep into a simple pen. This project is basically an extension of that simulation in which the sheep dog is provided with an intelligent agent-like architecture such that it performs the task on its own.

1. INTRODUCTION

1.1 Motivation

The program is basically a simulation of a “sheep dog trial”, in which a “dog” agent attempts to herd a collection of “sheep” agents into a pen, constructed out of circular posts. It does this (almost) regardless of the relative starting locations of the dog and sheep, and of the starting orientation of the pen. The motivation for the program was to extend an already existing program in which a user controlled the dog using the mouse, such that the dog was able to behave “intelligently” and independently.

The reason this was an interesting task is that the behaviour involved on the dog’s part in performing the task is by no means simple. If the sheep are located behind the pen, for example, and the dog in front, then the dog must first travel around the pen, then bring the sheep to the front and finally direct them in to the pen. It must also keep track of which sheep it has herded into the pen, while noticing which have escaped since being herded. The existence of trees in the “sheep world” provides further obstacles which make the task even harder.

The solution of the problem has implications, not just for the behaviour of an intelligent agent on a basic level as it navigates around its world, but also for ways in which it can interact with other agents. The other agents have a predetermined set of behavioural responses, and the dog must use its actions to prompt specific behaviours from them.

1.2 Characterisation of the problem domain

Before starting, it is worth noticing that the interpretation of the problem domain as involving sheep, dogs and pens is not a necessary one. It is only forced upon us (or prompted), by the appearances of the objects on the graphical display. Essentially, the problem is a mathematical one, involving the movement of objects through coordinate points on a graph, so as to get them as near to the origin as possible while avoiding certain other points through which they may not pass. That the objects are constrained to perform “sheep–like” behaviour is undermined by the fact that no studies of actual sheep or dogs were undertaken in the construction of the program. We might equally well construe the graphical display as depicting police chasing prisoners into a stationary meat-wagon, or any number of other alternatives, of which the sheep interpretation is merely the most suitable.

Any solution can be judged, not only on the dog’s ability to complete the task, but also upon the efficiency with which it does so, measured by the number of time-cycles taken to herd all the sheep into the pen. That is to say that, while there may exist other solutions that achieve the desired results, they will not be acceptable if they do so in a roundabout fashion that takes a lot of time to complete. Interestingly, as will be seen, some previous draughts of the program were able to complete the task in much shorter time, given the correct starting conditions, but were also prone to bouts of undesirable inefficiency. This meant that the average time taken was larger than that of the ultimate solution, which nearly always completed the task in between 370 and 430 time-cycles.

One final interesting question that emerges from the problem domain is to what extent it is necessary for the dog to abstract from the simple mathematics of the domain world. That is to say, how important is it to give the dog a series of explicit instructions along the lines of :

“Take the sheep to such-and-such a point near the opening of the pen”.

as opposed to allowing it to determine its behaviour directly from the relative positions of the various agents. This question shall be addressed later.

1.3 The State of Play: Peter Waudby’s 1996 MSc Summer Project

The program upon which this simulation is based, and of which it is to a large extent an extension, consisted of a graphical display, on which appeared iconic representations of a sheepdog (Gromit), five sheep (Sheepy, Sneezy, Sleepy, Bashful and Doc), 10 posts (in the formation of a pen) and two trees. The simulation behaved as a simple ‘game’, in which the user

attempted to control the dog in such a way as to herd the sheep into the pen, whilst avoiding the trees.

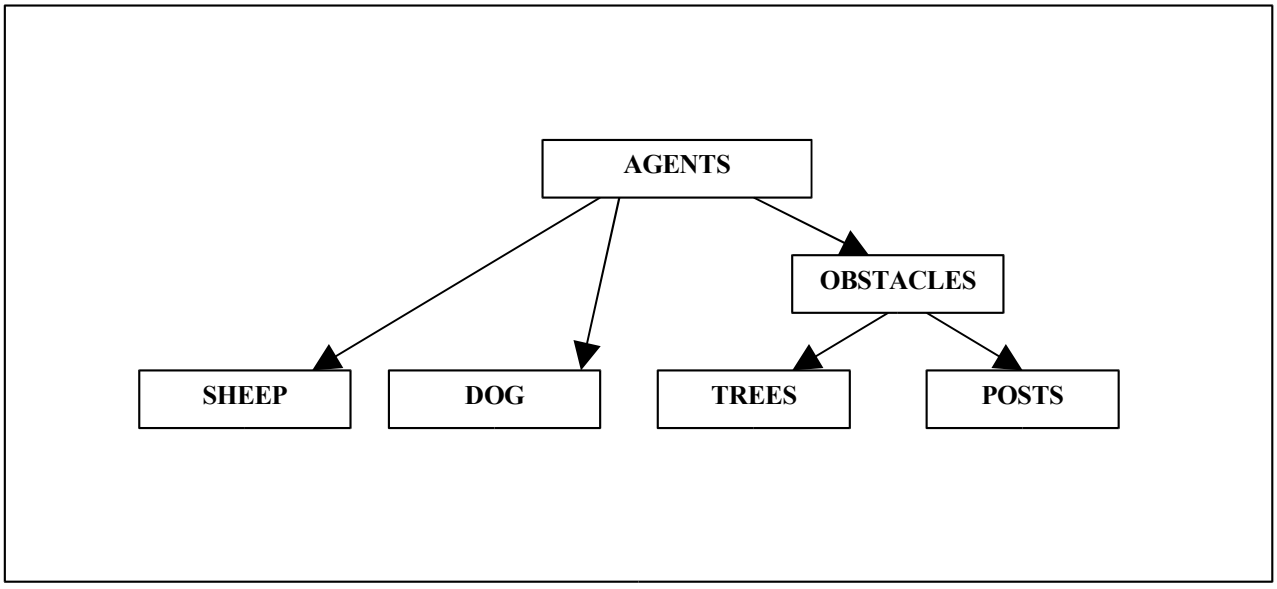
The trees and the pen were classified as obstacles (and the classes of which they were instances inherited the characteristics of a higher class, trial_obstacle). This meant that they were static objects, which could be moved by the user prior to starting the simulation. Both were circular in appearance, although the trees were larger than the posts. As neither exhibited any behaviour, neither had any rule systems associated with them. However, they affected the simulation in as much as it was impossible for sheep or dog to pass though them. In my extension of the program, these features have remained broadly unchanged.

The sheep behaved as independent agents, in a manner in which I shall describe in depth in the next section. They appeared on the display as a large circle representing the body, and a small circle representing the head. Again I have hardly changed the way in which the sheep work, except in one minor area which I shall detail later.

The dog was activated by the user, in that it's very simple rulesystem simply dictated that when the user clicked the cursor on an area of the display, the dog ran towards the point at a speed inversely proportional to the distance between it and the point.

The program made use of the objectclass, poprulebase and sim_agent libraries in the internal mechanisms of the agents, and various 'rc' libraries in their external presentations, as well as sim_geom for some of the mathematics involved in linking the two.

Figure x: Hierarchy of Classes in Original Program

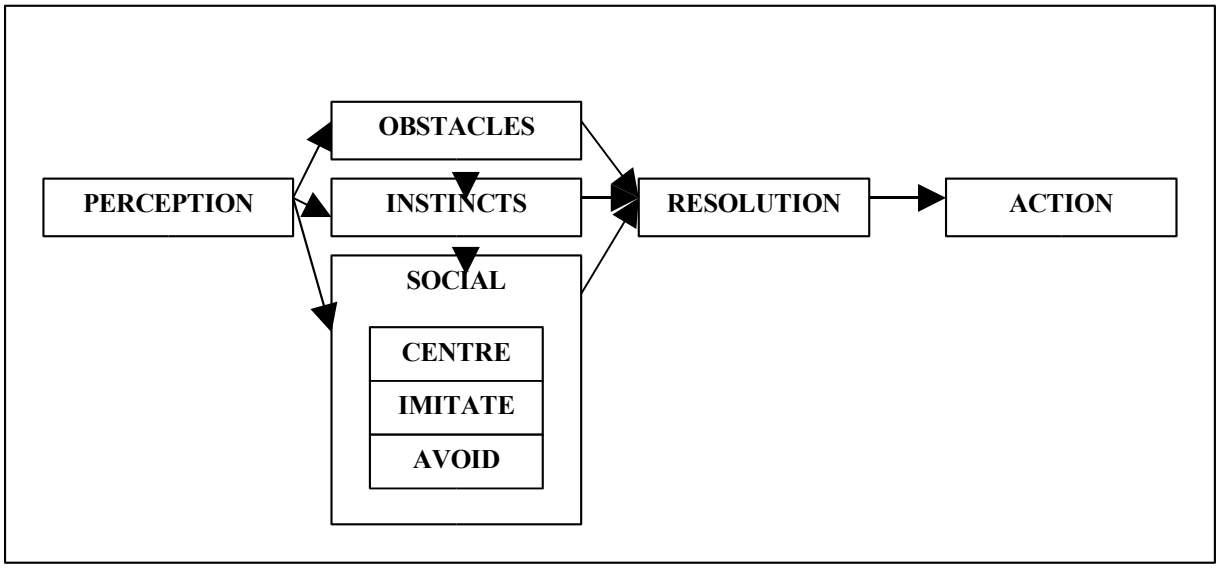


1.4 How The Sheep Behave

The behaviour of the sheep is regulated by a series of rulesets that essentially resolve the competing effects of various impulses which operate on the agent.

The broad overall structure of the sheep agents can be best understood by the following diagram.

Figure y: The structure of the rulesystems in Sheep Agents



This shows how the sheep first divides up its perceptions of the world, then processes these perceptions in paralel, decides how to resolve the various impulses and finally acts upon that resolution.

Firstly, then the sheep receives various perceptions of the other agents in the world (including trees and posts) depending upon their distance. These are entered into the agent's database at the beginning of each timeslice, along with a series of beliefs corresponding to the agents hunger, sense of urgency and so on. These perceptions are sorted into perceptions of dogs, friends and obstacles.

At the next level of the architecture, the obstacles are processed into a list of obscured ranges so that the sheep knows which directions not to head in. The instinct ruleset allows the sheep to decide whether to act upon the various different instincts: viz eating, fleeing (of which more later), wandering or idling. This corresponds to the biological level of the design: in other words it seeks to serve some of the basic survival needs of the sheep. In particular, the need to eat and the need to flee can directly effect urgency with which 'higher' level urges (e.g. social ones) are dealt with. The other two instincts are simply defaults, which are acted out when no other 'need' presents itself.

The social level of the design is reasonably complicated, and serves to cause groups of sheep to flock together and imitate each other, at a reasonable distance. Three rulesets are grouped together into a rulefamily, which cycles through them as each previous set becomes inappropriate.

The resolution level allows decisions to be made between irreconcilable impulses such as moving and eating. Vitally for the purposes at hand, unless the fleeing impulse is weak (i.e. the dog is not too close), it dominates. The action part of the architecture merely implements the decisions made by this section.

The fleeing impulse is the part of the sheeps architecture which concerns us, since it is this which directly affects the sheeps reaction to the proximate presence of the dog. Generally, this will just cause the sheep to move directly away from the dog, at a speed that is proportional to the distance of the dog. The exception to this is when an obstacle is in the way, when the sheep will ususally, although not always, move so as to avoid the obstacle. Also, the presence of other sheep nearby, will slightly affect the direction the sheep will run, if the dog isn't too close.

1.5 Summary of Achieved / Unachieved Targets

The problem has broadly been solved to a high degree of accuracy, although 100%. That is to say that given randomised starting positions for all the sheep, the dog(s), and the trees, the dog will complete the task over 90% of the time, without making any fatally unintelligent move. This is done in an efficient manner, avoiding both trees and pen posts.

The small number of times in which the dog fails to complete the task involve it getting caught between two posts in the pen, a situation which it very rarely gets into. This situation can only really occur (because of the nature of the solution because of some freak starting condition). Another similar situation which can be fatal is when the dog gets caught between two trees which happen to be placed very close to each other.

While the program is quite efficient, I'm not convinced that more efficient solutions could not be found. Indeed, much of the last week of programming was spent tweaking important numbers in efforts to improve efficiency, and it seems likely that if this process was continued, the average number of time-cycles taken to complete the task might significantly reduce. Moreover, the solution presented is, I am convinced, only one of many possible ones, and it would never be possible to say that alternative versions might not be radically quicker.

The major drawback of the solution, in my mind, and one which may indeed reduce efficiency, is the inability of the dog to deal with the sheep in flocks. This is the main unachieved target (see project proposal). It would have been nice for the dog to have attempted to herd closely situated sheep in one go, rather than individually, as this is both more realistic in terms of what real sheep dogs do, and closer to how the human user tends to solve the problem. I shall discuss ways in which this issue might have been addressed later in the report.

1.6 A bit of Philosophy / Theory

The dog agent, as it stands, is clearly little more than a rule-based expert system (Nilsson 280-86), albeit an expert in a particularly unusual field. It's sole function is to herd the sheep into the pen, and it has no other concerns to distract it from this task. It does not get tired, hungry or distracted from its task, about which more later. However it differs from the expert systems described in Nilsson, in that it's knowledge is not stored in a knowledge base, rather it consists of a set of prescribed reactions to specific situations. That is to say, the knowledge is procedural rather than propositional: it 'knows' *how* to herd the sheep without knowing anything *about* herding sheep. (Sharples et al, 60 -61).

In writing the program I deliberately avoided using any kind of search mechanism in order for the dog to complete its task. (Whereby, the dog would have started by looking for a series of finite steps that could take it to the end goal.) There were two reasons for this: firstly, the search space was simply to large (potentially infinite) to allow such a solution to be plausible. Secondly, it seemed reasonable (and in fact turned out to be the case) that the possible sheep-worlds could be usefully divided up such that the dog could react individually to different situations, without having to discern too many different situations.

To understand this, consider that the best solution to the problem may consist in providing a 4 or 5 dimensional graph, which plots for any sheep co-ordinates and dog co-ordinates the optimal direction for the dog to head, relative to the orientation of the pen. This would provide a total guarantee that if there is a way to solve the problem, then the problem would be solved. However, coding such a mapping of coordinates onto heading would be a super-human task (although it may be an interesting question whether it could be statistically uncovered by a distributed network given sufficient units to represent the task). At the very least, it would require fixing the bounds of the world so that it was finite in size.

What this solution basically involves is a division of such a mapping into identifiable and separate dog-sheep relationship types, such that the dog must simply recognise whether it is in one of between twenty and thirty situations. The interesting question is whether this is the best way in which to create the situation-divisions.

2 HOW TO RUN THE PROGRAM

The program can be found in my file-space at #msc24tmc/lastshep.p. If this is loaded into xved and then loaded, using the ENTER |1 Command, it will bring up a set of instructions. Essentially, the user can relocate the objects in the sheep world, before pressing RETURN to start the simulation running. At any point, the user can stop the simulation by right-clicking on the sheep window.

3 SCENARIO

In a typical Run through of the simulation, the dog will take the sheep in turn, fist approaching them if they are at a distance, then directing them to the front of the pen, before finally herding them into the pen. The sheep are taken in a random order, and after each sheep is herded, the dog checks to see which ones are still left out of the pen.

4. EXPLANATION

4.1 Ontology

The ontology of the sheep world remains broadly the same. That is: dog, sheep, trees, pen-posts continue to play a part. There are two significant differences that should briefly be remarked on. The first is the presence of two targets. These are used by the dog in its attempt to steer the sheep in the pen. One is located at the centre of the pen, while the other is placed in the middle of the back of the pen, such that an extended line from one to the other vertically bisects the pen. In terms of size they are negligible and make no difference to the action of the simulation. They could have been removed, and artificial points been worked out by the dog in their place, and the only reason this hasn't been done is lack of time. Moreover it didn't seem that important.

The second difference in the ontology of the world is a potential one. Since the dog agent is coded so that it never refers to itself by name (e.g. 'Gromit'), rather as 'sim_myself', it is possible to have more than one dog agent performing the task at the same time. As the sheep are listed in the dogs' 'mind' in a random order, it does make a difference when this is done, namely two or more sheep can be herded at the same time. Extra dogs have been provided for in the code, but they are commented out as they significantly reduce processing time.

One further point needs to be made, namely that the pen is now an object in its own right (albeit one that is simply a construction out of pen posts), i.e. it is an instance of the class pen. Previously no concession was made in terms of objects and classes to the existence of the pen. This means that other pens could be created in the world, although their presence would always be surplus to requirements!

4.2 Data Structure for the Dog

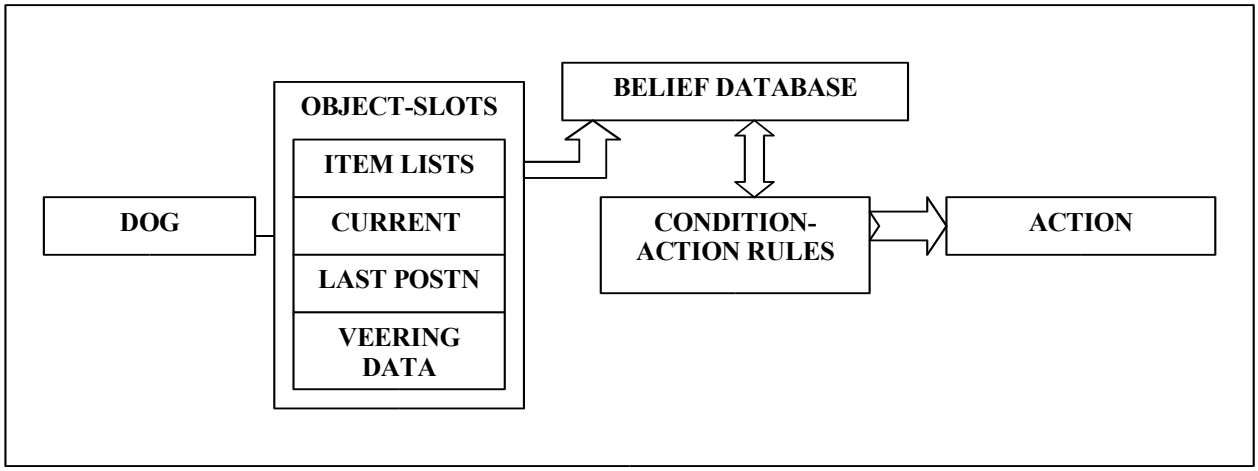
The data structure of the class dog is roughly as depicted in this diagram. The object has various sots which it uses to keep track of important information about objects in the world. For example, it maintains lists of all the sheep, trees etc, as well as keeping information about the sheep it is currently targeting. It is these slots that are updated during the dog's current time slice.

The second important feature of the dog is a belief database, (upon which the condition –action rules are worked). This is updated at the start of every turn, with the last turn's "beliefs" effectively wiped out, and replaced with new ones corresponding to the occupants of some of the dog's slots. Thus, while the dog effectively has no memory of the previous time slice (with one small exception) since none of it's beliefs persist from one time slice to the next, it's current beliefs are always dictated by the perceptions of the previous time slice. This can be seen as a

reaction time effect: The dog can not react to the things that it sees at the same time that it sees them.

Finally the mind of the dog is partly made up of the condition-action based rules that produce it's behaviour. As noted above, these can be seen as a form of procedural knowledge concerning the nature of the task of herding the sheep.

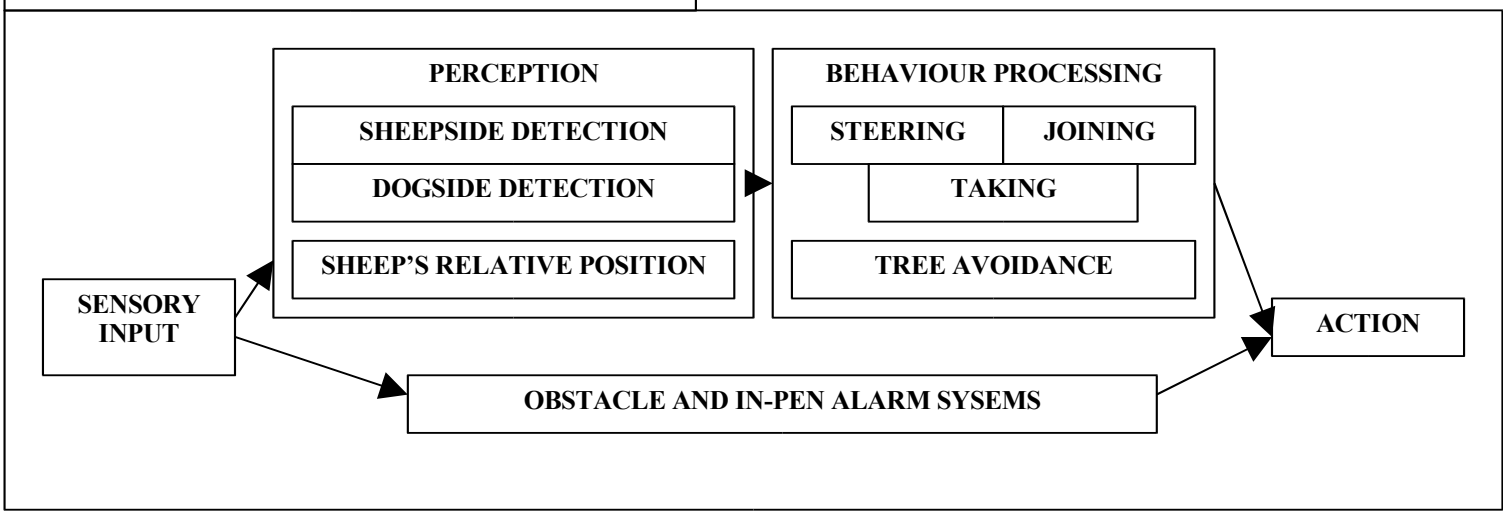
Figure z: Data Structure of the dog.



The following diagram gives some indication of how the dog proceeds from sensory perception to action. This occurs via two routes. The first route governs the type of behaviour that is used in general to solve the problem, and consists of a round of perception processing, where the dog works out the position of itself and the sheep relative to each other, and the pen and then a round of behaviour processing where it decides how to react to the information it has gathered.

The other route consists of two types of alarms, which tell the dog when it is in the pen, and when it has not moved since the last time-slice. These then cause immediate action.

Figure z2: Dog processing system



4.3 Higher Level Processes

4.3.1 Sorting the objects in The World

One of the first things the dog does, when the simulation is set running, is to create lists of all the objects in the sheep world. That is to say that, it divides the world up into sheep, posts and trees and then makes records of each of these things in slots in its longer-term memory.

These lists serve no necessary function in the running of the dog, in that the dog could be written such that on each occasion where the dog has, say, to test whether it is heading for an obstacle, it tests over all the objects in the sheep world, and adds the condition that the thing must be an obstacle. The main advantage in dividing the objects into lists is one of efficiency: now the dog can ask straight out at any point 'are any of the obstacles in my obstacle list in front of me?'.

There is also a sense in which this is a realistic agent-like piece of behaviour: it might be said that in doing this, the dog is acquiring a more conceptualised picture of the world, where it is aware that there are objects of different types (or classes).

On top of this formulation of lists, the dog keeps a record of the targets, and moreover of the front two posts of the pen. These four points are vital in the steering of the sheep into the pen. It might be said that the dog is keeping track of the most fundamental features of the pen.

Only one point ought to be made about the execution of the recording of the pen features, namely, that ideally it would have been done by extracting necessary mathematical features from the pen. In other words, the left front post might have been sorted precisely because it is in the left and front of the pen. This was not done however (largely because I couldn't figure out the maths involved in the time allowed), and rather the posts were allotted to their slots on the basis of their names (e.g. post1, post10 etc.). This means that should the names of the posts be changed for some reason, the solution would cease to work. This is not only poor coding but also unrealistic (the dog agent ought not to know what 'post10' is), and would be changed if any continuation were attempted.

4.3.2 Maintaining and Updating the record of the current sheep

Another vital part of the solution to the problem is the dog's ability to cycle through the sheep in an order, dealing with them one at a time. Initially, the dog is provided with a randomised list of the sheep in the world, such that the first sheep in the list could be any of the five. The list is randomised so that if more than one dog is used then the extra canine agents are not surplus to requirements. Initially the sheep just appeared in the list in such a way that the same sheep ('sheepy') was bound to be dealt with first. This meant that two dogs would simply carry out identical operations on the same sheep. In fact this even slowed the solution as extra dogs tended to get in the way of each other.

Giving the list of sheep a random element is a half way solution to this problem, because it still allows the dogs to deal with the same sheep at the same time, just making it unlikely that this will happen. The perfect solution would note for each dog agent, which sheep in the list are not currently either in the pen or being herded by another dog. One reason why this has not been attempted is that the most obvious solution (i.e. allowing communication between dogs) is unrealistic. What is required is an algorithm for a dog to identify "currently being herded sheep" by their behaviour. Since no obvious solution to this sprang to mind, I decided not to waste too much time on it.

What currently happens is that when the dog agent has satisfied a set of conditions related to the current sheep being far enough into the pen (namely that it is near the main target, and that the dog is also in the pen), the dog re-build its list of unherded and therefore available sheep, and picks the first in its list to be the current sheep.

One other improvement that might be made to the way in which these list are made would be to find a more useful (or maybe more realistic) algorithm for organising the lists of sheep, such that the first sheep is (for example) the nearest, or maybe the most visible (if there is a visible sheep).
This would require a series of experiments to see what sort of algorithm might be the most efficient in this context.

4.4 The Nature of The Problems

The solution to the problem involves dealing with four *sub-problems* which face the dog as it goes about its task. In the program each of the following corresponds to a type of behaviour, the

procedural knowledge involved in which is represented by a ruleset, which in turn is a member of one rulefamily, that only allows one of these behaviours to be active at once.

4.4.1 Getting to the Sheep

Much of the time, getting the dog to the vicinity of the sheep is simply a matter of heading towards the sheep at a reasonably fast speed, and then changing the behaviour when the sheep is only a certain distance away. The exception to this is when there is some sort of obstacle between the dog and its current sheep, such that heading straight for the sheep will mean that the dog gets caught on that obstacle. In the case where the obstacle is a tree, the behaviour is simply switched to 'avoiding trees'. However where the obstacle is the pen, the problem is complicated in the following ways.

Firstly, treating the pen as a simple unified object is not a straightforward task. Since the pen consists of a series of individual posts, it is necessary for the dog to create it's own unified perception of the pen.

Secondly, the fact that the pen is square means that it is simply not possible to rely on being able to travel round the pen in a circular fashion by maintaining the same distance to the centre. The problem is that where the dog starts close to the pen, there will be a tendency to become stuck on the corners. Furthermore, such a solution is inefficient and looks unrealistic.

4.4.2 Taking the Sheep to the Front of the Pen

Once the dog has arrived at the sheep, if the sheep isn't already at the front of the pen, the dog has to navigate it so that it is. The main problem here, is getting the sheep past the pen (and obviously avoiding any trees that get in the way). Again it is necessary to treat the pen as one item.

It is also important here that the dog forces the sheep to stay within a reasonable distance of the pen. It will enormously detract from the efficiency of the solution if the dog takes the sheep round the pen simply by forcing it to head directly away from it. Thus it is important that some effort is made to steer the sheep as close to the pen as possible without actually going straight for it.

There is a marked difference between taking the dog around the pen, and getting the dog to take the sheep around the pen, in that it isn't possible (or at least isn't feasible) to get the dog to be precise about what heading to make the sheep head in. This is because, by the time the dog has configured itself to make the sheep head in an appropriately calculated direction, the relative position of the sheep has changed so much as to invalidate the original calculations.

4.4.3 Steering the Sheep into the Pen

This is by far the easiest of the problems, given that the sheep is in a position in front of the pen, which it will be, when the above problems have been solved. Essentially, it involves making sure the dog passes between the two front-most posts of the pen until it is sufficiently far into the pen that it wont bounce out again.

The only difficulty, given the nature of the identity of the 'agents' that the posts are, is making sure the sheep does not get caught on the sides of the two front posts. If the sheep is heading too close to these two, then it has a tendency on the one hand to get stuck, and on the other to veer in the wrong direction and then pass down the side of the pen. This latter can add many time-cycles to the length of the solution, because to get back into a suitable position may require going all the way around the pen

The general problem of steering the sheep into the pen can be seen as analogous to taking a free kick in soccer. If the position of the kick is too wide to the left or right of the goal, then the angle between the goal posts becomes too narrow to score, and the ball must be crossed to a position more directly in front of the posts, before the goal can be attempted. Similarly, with the sheepdog, if the sheep is too far to the left or right, even though it is in front of the pen, then it will be necessary to bring it into a more central position, before finally forcing it in.

4.4.4 Avoiding Trees

There are two vital ways in which the trees must be avoided. On the one hand, if the tree lies between the dog and the sheep, then the dog must go around it to get to the sheep. This is an easier problem than travelling round the pen, because the tree is round, a single object and much smaller than the pen.

The other problem is one where the tree is between the sheep, and where the dog wants the sheep to head. That is to say that the dog has reached the sheep, and is busy steering it in a desired direction, which is obscured by a tree.
The major difficulty with trees, especially with the latter problem is recognising when there will be a problem with sufficient time to spare to do something about it. Often, particularly with the solution as it is currently designed, the dog will change course at the last minute so that where it was previously going round the tree it is now actually heading for it, either with or without the dog. It is an essential part of the solution that the dog doesn't know where it will be, and where it will be heading, in future time-cycles, rather it reacts on the spot to individual situations. Hence it is very hard to predict when these situations are going to occur.

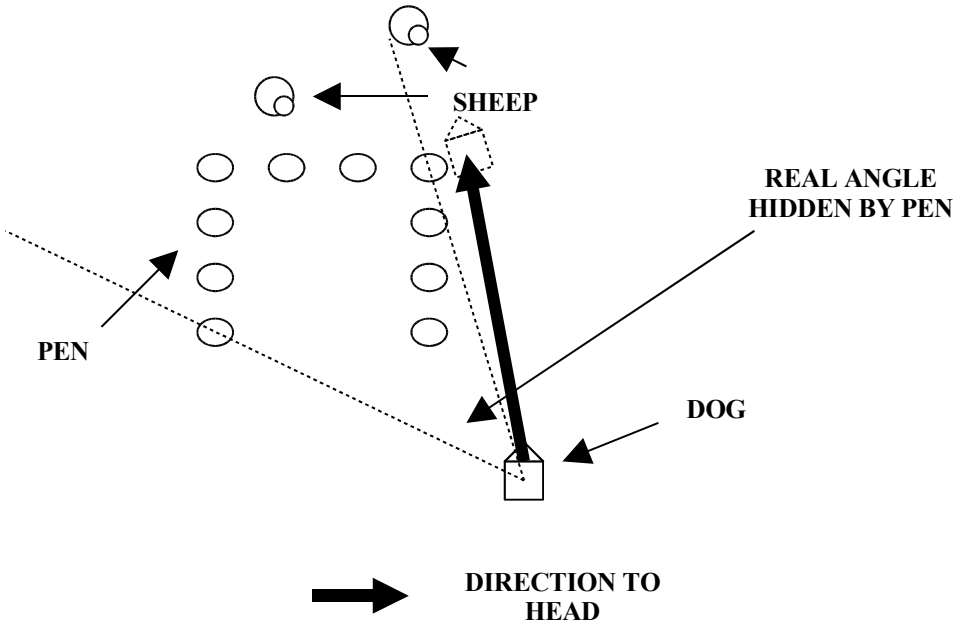
4.5 How the Problems are Solved

4.5.1 Joining

As mentioned above, if the sheep is visible to the dog from the start, this behaviour never presents a problem. It is simply a matter of heading towards the dog at a reasonable speed. The problem occurs when the pen lies between the dog and sheep. The problem can be broken into three separate ones: deciding when the pen is between the dog and the sheep, deciding which direction to head in order to avoid it, and deciding when it has been avoided, i.e. when the currently obscuring corner of the pen has been passed.

As can be seen from the figure below, the problem involved in deciding when the sheep are obscured is not as simple as it sounds. Firstly, the dog finds the posts that are to the extreme left and right of the pen as far as it is concerned visually, and records their bearings. If the sheep lies between these two then it is obscured by the pen[1], but it is not the case that 'if the sheep is obscured by the pen then it lies between these bearings', because there are frequent cases where the sheep is obscured by the edge of the outermost post and therefore does not fall between the bearings of the posts.

Figure ASD: Getting around the pen, the basic problem.



[1] And it is further away than the pen!

Consequently, as shown in the following diagram, a further calculation must be made to figure out the extra bearings which are obscured. Note this extra bearing can be quite significant if the dog is close to the pen.

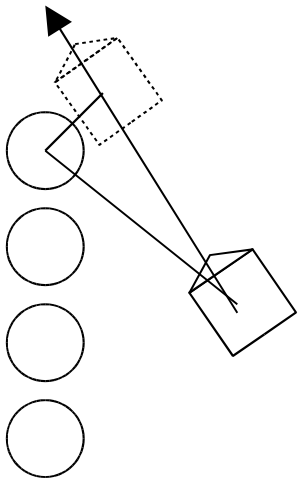


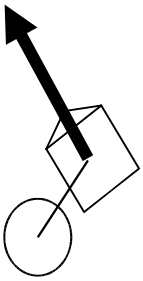
Figure n: Additional bearing for veering around the pen.

The method used to calculate the extra bearings obscured by the pen (and hence the desired heading of the dog), is rather approximate. It is assumed that the above figure describes a right angled triangle (which is far from obviously the case), and the additional bearing is calculated by finding the *sin* of the 'personal space' of the dog added to the size of the post over the distance from the dog to the extreme post. This provides the angle which must be added to the extreme post in order to calculate the total bearings obscured by the pen.

The 'personal space' of the dog is used to eliminate problems caused by the fact that the triangle which is used in these calculations is rarely a perfect right angled triangle. The personal space is some number larger than the size of the dog. What this effectively means is that while the dog can more consistently veer round the pen, it sometimes veers by an unnecessarily large amount. Ideally we would make the personal space of the dog vary systematically with the distance of the dog from the post, but since I don't know the equation for this systematicity I have used an approximation.

Once it has been established that the sheep is obscured by the pen, the extreme edge of the pen whose bearing is closest to that of the sheep is recorded, as is the post that the dog is trying to avoid. It was discovered that the dog must keep track of what it's desired heading is while it is veering around the pen, rather than re-compute it at the start of every time-slice. If the latter course of action were taken, the dog becomes confused, trying to go both ways round the pen, and often ultimately ends up stuck on the middle of it. For this purpose, a slot was provided (deshead) for the class dog, which kept track of this information.

Consequently, it becomes necessary for the dog to decide when it has successfully veered around the pen, and can therefore ignore its 'desired heading'. This is calculated in the following manner:



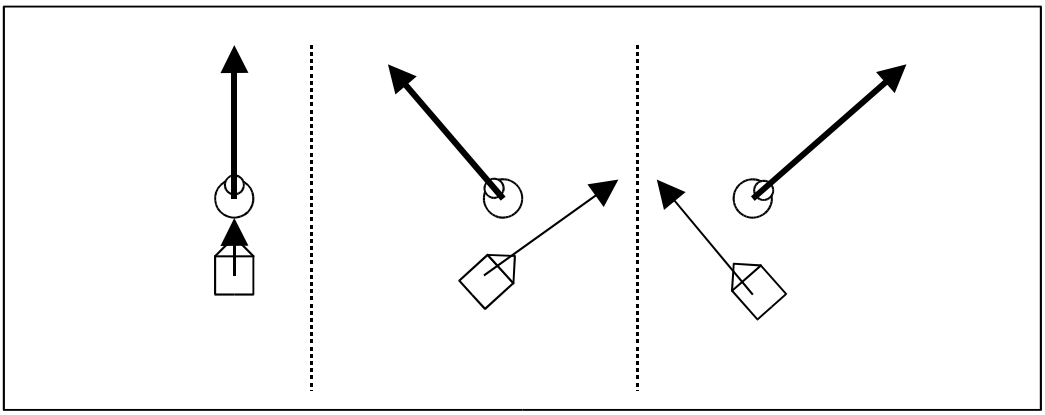
When the angle between the desired heading of the dog and the bearing of the post from the dog becomes greater than 90 degrees, it can safely be assumed that the dog has past the post, and therefore passed the pen. At this point, the dog can either head directly towards the sheep, or can compute a new desired heading.

4.5.2 Taking

The essential thing to realise in order to understand the solutions to taking and steering the dog into the pen is that in order to get the sheep to travel in a certain direction, it is necessary to align the sheep onto the bearing in which you want it to head. Since the sheep will (generally) run directly away from the dog, this will cause it to run in the desired direction.

Furthermore, it is easy to see that moving to the right of the sheep will cause it to move to the left, and vice versa:

Figure abc: how the sheep reacts to the dog



Having realised this, the task is to decide which direction the dog wants the sheep to be heading in and then to move so as to put the sheep into that heading in relation to itself.

It turns out that it isn't necessary to be precise about which direction the sheep needs to head in, rather we can specify a range of headings which are appropriate. In order for the sheep to be taken to the front of the pen, it must be driven in such a direction that it is not going towards the pen, and yet isn't getting further away from it.

The dog divides the area around itself as follows:

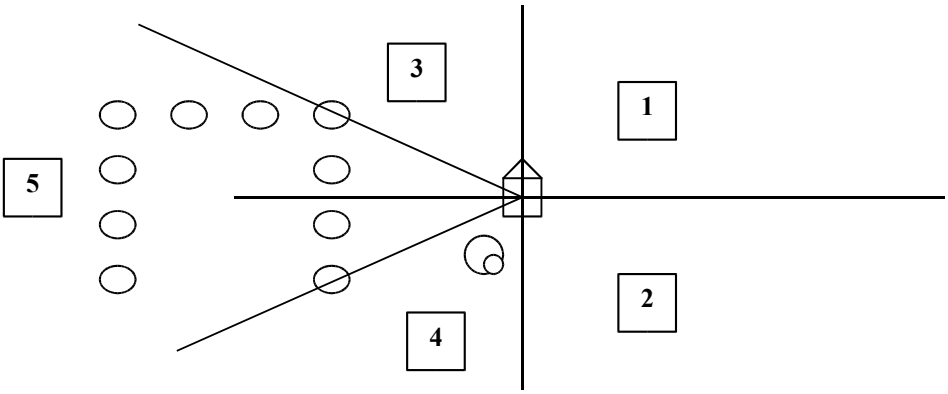


Figure d: How the dog divides the relationships between itself, the pen and the sheep

Such that it tries to keep the sheep in sectors 3 or 4. To be more specific, the first four sectors are created by taking a line from the target at the centre of the pen to the dog, and providing another line perpendicular to that at the dog. The fifth sector is defined by the bearings from the dog to the outermost edges of the pen. It should be noted that for the purpose of 'taking' the sheep to the front of the pen, as will be seen later, the sheep will never be situated in sector 5 on the other side of the pen.

If the sheep is situated in sector 2, then the dog aims to the left of it, so that the sheep moves to the right. If the sheep is in sector 1 then the dog goes to its right. These have the effect of moving the sheep into the correct sectors (3 and 4), whence the dog can aim straight for the sheep, in order to move it around the pen.

On its way around the pen, although it is in sectors 3 or 4, the sheep will often be driven into sectors 1 or 2, simply because of the changing nature of the relationship between the dog and the pen. In this case it is simply returned to its proper place by a swift movement from the dog.

If the sheep is in sector 5, then it is important to proceed with care (and thereby prevent it from running straight into the pen), so the dog moves much slower than usual. The idea is to try to get in between the sheep and the pen, so that the sheep can be made to follow a reasonable line. This is often not an easy thing to do, and some efficiency is lost during the process.

It will be noticed that is not strictly speaking necessary to allow the sheep to be driven straight in (for example) sector 3 of the above diagram. This will take the sheep on the long route round the pen. In the solution, this is generally avoided by attempting the dog to go round the back of the sheep as it approaches (by aiming slightly to the left or right of it), so that it starts its taking manoeuvre in a suitable position. The exception to this is when the sheep is close to the pen, in which case the dog can collide with the pen if it attempts to do this. Furthermore, when the sheep is close to the pen, the efficiency lost by going around the pen the wrong way is only minimal.

Admittedly, a better solution might have been to specify more tightly the bearing along which the sheep was to head based upon the direction of the extreme edge of the pen. However, the simplicity of this solution appealed to me, as it did not require any great feats of mathematics to work out what were acceptable directions.

4.5.3 Steering

The solution to the problem of steering the sheep into the pen is very similar to the solution of taking the sheep to the front of the pen. The major difference is the sector in which the sheep must me in order for the dog to head directly towards the sheep. In this case, the area around the dog is divided up as follows:

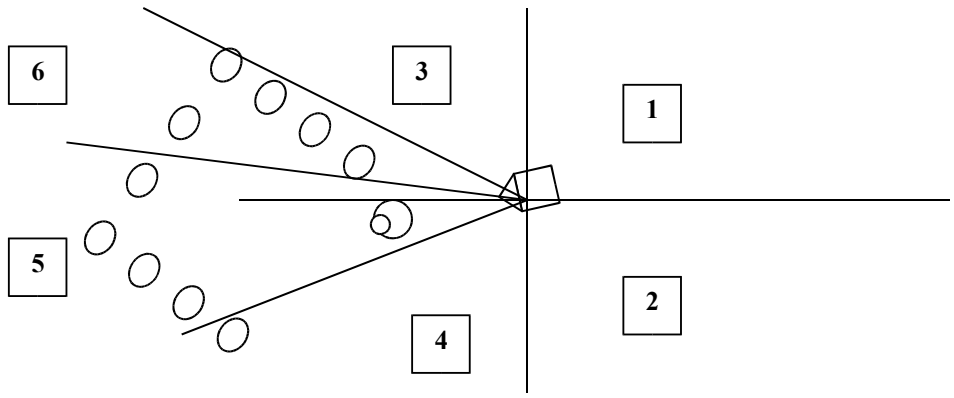


Figure d: How the dog divides the relationships between itself, the pen and the sheep for steering

Note that area 6 may be on either side of area 5 depending upon what the position of the dog is in relation to the orientation of the pen. Area 5 is defined as the range of headings between the innermost edges of the front two posts. Again, the bearings of the innermost edges are worked out in a rather approximate fashion using the *sin* of: the size of the post added to the size of the sheep over the distance between the dog and the post. These figures are then added or subtracted to the bearings of the posts. In this case, however, the approximate nature of the solution is less damaging because the sheep will naturally veer away from the posts anyway.

Thus, so long as the sheep is aiming for the correct side of the post, it will veer to avoid it altogether.

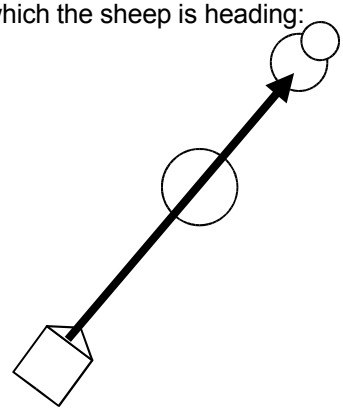
The general idea is that in sectors 1, 2, 3, 4, and 6, the dog turns the sheep an appropriate amount to the left or right, so that it is pushed towards being in zone 5. In zone 5, the dog aims straight for the sheep thus forcing it into the pen.

The amounts by which the dog turns the sheep in each zone can quite easily be varied.[2] Currently, they are on the high side. This enables the dog to guarantee pulling a fast enough turn as it reaches the front of the pen to get the sheep in the pen without the danger of sometimes over- shooting and being forced to go around again. The down side however is that it decreases the general accuracy of the solution, because quite often , one attempt at turning the sheep will force it right through zone 5 and into the other side, from where it may be turned straight back again. Such movement will actually cause the sheep to move overall in the correct direction, but not in a particularly efficient manner. A positive note about this problem is that it is very similar to the way in which human users deal with the task: i.e. not getting the dog too close to the sheep, but rather taking long sweeps behind it at a distance. This suggests that one of the reasons that the intelligent solution differs from the human one is that it is able to provide increased precision for the dogs movements.

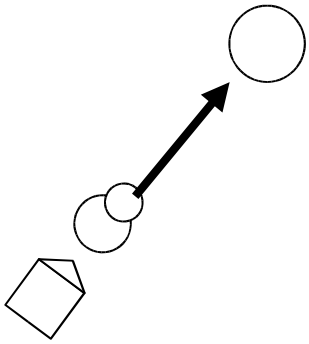
It is important to note that since zone 6 can be on either side of the zone 5, the direction in which the dog turns must depend upon the side of the pen the sheep is on. Here side is defined in terms of being on the left or right of a line from the first to the second target.

4.5.4 Avoiding

The behaviour involved in avoiding trees is not too complicated. As with the pen, a tree can cause two types of problem: either it can block the dog's path to the sheep, or it can block the direction in which the sheep is heading:



Situation 1



Situation 2

Because the tree is a relatively small and simple object, there is no great need for any intricate behaviour in order to avoid colliding with it.

As far as situation one is concerned, the solution simply involves travelling around the pen. All that has to happen to achieve this is to recognise that the tree is obscuring the sheep, and then to head a certain amount to the left or right of the tree until this is no longer the case. Effectively this will make the dog travel in an arc around the tree:

[2] The amount the sheep turns in relation to the dog is proportional to the size of x where the dog moves in the direction of the sheep plus or minus x.

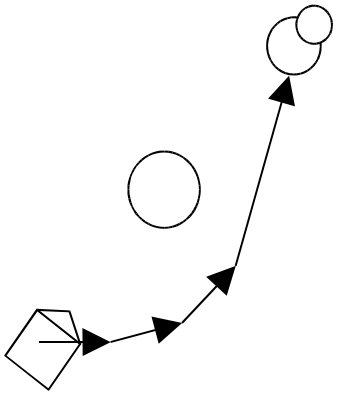
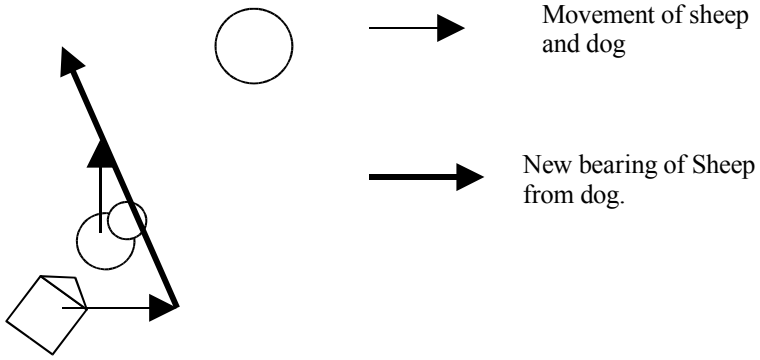


Figure n: Solution to the first tree related problem situation.

An ideal solution might adjust the direction in which the dog travels around the tree depending on the positions of the dog and the sheep, but for the moment, this seemed unnecessary: so long as the dog goes round the tree in one direction, the tree isn't large enough for it to matter which.

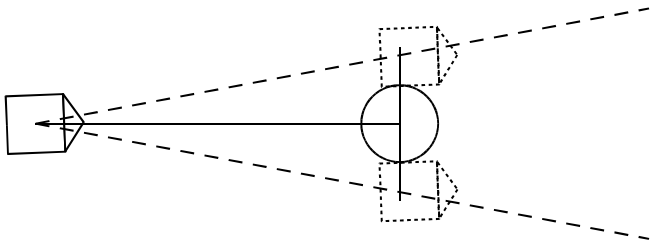
The solution to the second problem is, if anything, even simpler: it is simply necessary to move such that the sheep is no longer in line with the tree.

Figure m: Solution to the second tree-related problem situation.



This solution may occasionally bring the sheep into another troublesome situation. It does however work on the vast majority of occasions. Moreover, on those occasions where it doesn't work, it is merely at a cost to efficiency, not to the overall solution.

It only remains to explain how the dog works out when the sheep is in trouble with regards one of the trees. This is done in a very similar fashion to method used for the pen:



Essentially, the dog works out the bearing of the tree from itself, and then works out the sin of the 'size' of the tree (which = its radius) plus the size of the dog over the distance of the tree from the dog. It then adds and subtracts this latter number from the former to give two bearings which consitute the range obscured by the tree. If the bearing of the sheep falls within this range then the sheep is obscured.

Techinically, for situation two (where the sheep is in front of the tree), this is overkill, in that it isn't necessary for the dog to pass the tree, rather the sheep. Thus using the size of the dog in the calculation is allowing more room than is necessary. However, since the dog is always

larger than the sheep, and since this solution always seems to work, it has been left. The dog only accepts the calculations if it is within a certain range of the tree (it doesn't want to be affected by a tree on the other side of the world). If the tree is nearer to the dog than the sheep, it activates the first solution, otherwise it activates the second one.

4.6 Other Details of the Program

4.6.1 Alarms

As was mentioned before, there are two 'alarm systems' in the architecture. Although each is implemented in a slightly different way, they essentially serve the same sort of purpose and could have been written in the same fashion.

4.6.1.1 Dog-in-pen alarm

The first alarm system works at the level of the behavioural rulesets: this means that there is a rule at the beginning of each set which detects whether the dog is in the pen, and if so, overrides that behaviour. Whether the dog is in the pen or not is worked out geometrically by taking a line from each of the corner posts to form a square, and asking whether the dog is on the same side of each line. I.e. if it is to the left of the line from post1 to post4, to the left of the line from post4 to post 7 etc.

When the dog realises it is in the pen, it immediately heads directly out: i.e. along the line of the orientation of the pen (actually 90 degrees away from its official orientation, because of the mathematics involved in setting up the pen)

Importantly, this is where the built in reaction time effect plays a role. Because the dog doesn't react to situations until a turn after they come into existence, this allows it enough time in the pen to force the sheep in far enough to stay there after it has left.

The main reason for having this alarm system is that once in the pen, because it is such a tightly enclosed space, the dog's behaviour becomes confused and if left to its own devices it often gets caught on the sides and so on, not to mention scaring the already herded sheep out again.

4.6.1.2 Dog-not-moved alarm

This second alarm plays an important role in ensuring that eventually the task will be completed. Because the dog, despite its best efforts, still occasionally gets caught on trees or on the side of the pen (for example when two trees are very close together), it is necessary to give the dog some mechanism for dealing with these situations.

This alarm system works in paralel with the other behaviours. That is to say that it can be activated at the same time as another piece of behaviour, although by its nature, it never occurs at the same time as other movement.

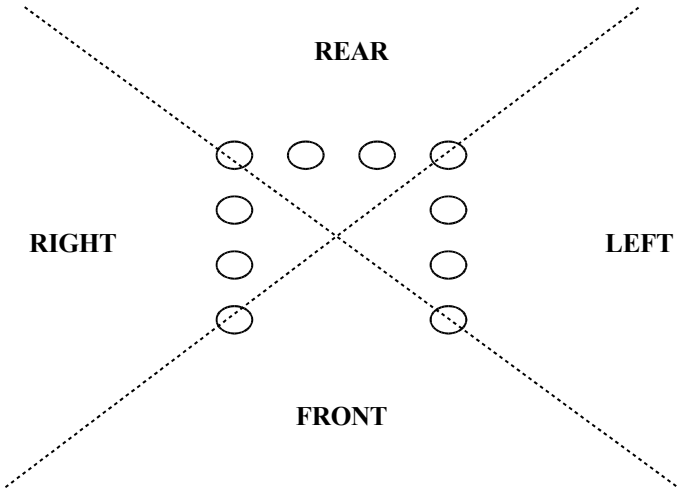
Essentially, what it detects is whether the dog has moved in the last two time-cycles. If not then it activates a kind of panic mechanism that turns the dog 90 degrees to the left or right and attempts to move quite fast. Whether the dog turns left or right is randomly decided.

This mechanism is simplistic, but effective, in that within a relatively small amount of time, the dog is usually guaranteed to have cjanged its position sufficiently to be able to re-assess the situation. Effectively it acts as a fail-safe. That it is crude is not a huge matter for worry, as it might be said to reflect the panic behaviour of the dog if stuck in a tight situation, and furthermore as an alarm it could be based upon some evolutionarily previous mechanism left over from a more crude state of the dog. (More later on this.)

4.6.3 How to decide which behaviour to use

Which behavioural ruleset is used at any point is decided by a number of factors. Tree avoidance is used on an ad hoc basis, whenever it is required. The use of other rulesets

depends fundamentally upon which area around the pen the sheep and the dog find themselves in during any time-cycle. This can be understood with the help of the following diagram.



If both sheep and dog are in the front area, then the dog uses its steering behaviour. Otherwise if both sheep and dog are in the same sector, and they are within a certain distance of each other, then the dog uses its taking behaviour. If they are in different areas, then the dog uses its joining behaviour.

Once the dog has started its steering behaviour, it is slightly harder for it to stop, because a new definition of the front area comes into play, viz the area in front of a line from one front post to another extended to infinity in either direction.

Finally, if the dog is involved in taking behaviour, it will not swap to joining just because it has taken the sheep into a different area, whilst remaining in the old area itself.

All of these rules are void if the sheep or dog are in the pen (see above).

5 Limitations and Possible Extensions

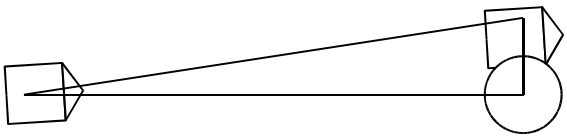
I will divide this section into two parts. In the first part I will talk about improvements that could be made in order to make the dog agent a more successful solution to the problem. In the second I shall talk about what might be necessary to make the agent more agent-like, and less of an expert system.

5.1 Limitations to the Solution

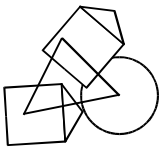
5.1.1 Veering

The fact that the veering mechanism used by the agent is not yet perfect (either for the trees or for the pen) is given away by the need for the alarm mechanism which deals with cases where the dog is caught. In a perfect solution it would be clearly unnecessary to detect whether the dog was in a troublesome situation because, such situations would never occur. It is the veering mechanism which gets the dog into troublesome situations more often than not.

The reason for the imperfection of the veering mechanism, as I have mentioned, is its reliance on there being a perfect right angle triangle between itself, the post it has to veer around and the point at which it will pass the post. While this is (almost, I think) true for posts from which it is a reasonable distance away:



This is not true when the post is up close:



This is because up close, the part of the post which needs to be passed can be a bit of the circle which juts out into the dog's path. Essentially, I think, the solution to this problem is a mathematical one, and I have already noted that the dog here uses a concept of personal space that means it tries to give the posts a wide berth. However, the best solution to the problem would in my view be to actually give the dog a new type of sensors. Under the current sensors, the dog is aware of the location of objects (in an artificial way), and of their sizes. This as patently unrealistic as it is inconvenient. A better more realistic solution, and one which lends itself to the veering problem more, is one in which the agent knows the range of headings obscured by the object, and its distance.

Something akin to this is attempted in the sheep agent, although this isn't perfect either. Although the sheep moves so slowly and randomly that it hardly ever bumps into the obstacles, it does on occasion, and is thus not totally solving the veering problem.

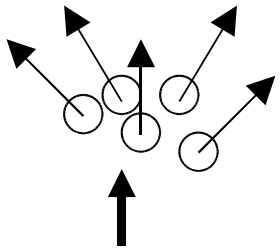
One other alternative that is a potential answer to this veering problem is offered in the teach sim agent teach file. Here the agent is given a kind of field of attention that sticks out in front of it. It will only attempt to deal with obstacles that fall within this field. The field is such that the agent will ignore obstacles at a distance if they are not almost directly in front of it. At close quarters, however, it will attend to obstacles at almost ninety degrees to itself- in other words those which are causing trouble to the dog.

Ultimately this limitation is due to the nature of the sensors on the dog. In fact, these are totally unrealistic, and I will discuss below how they ought to be updated.

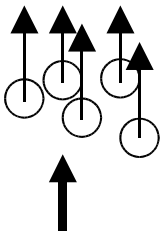
5.1.2 The Concept of "Flock"

As I have noted, on of the main targets that was not achieved in this projects was the notion of giving the dog the concept of a flock. This would both potentially improve the efficiency of the solution (more than one sheep would be herded at once), and make the whole agent more realistic: both real sheep dogs and human users of the sheep simulation tend to try to herd the sheep in in a flock.

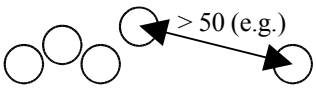
Two reasons exist as to why the solution cannot do this. On the one hand, it would require a massive overhaul of the way in which the dog goes about its business. On the other, the sheep themselves do not have a useful flocking mechanism. While they do have an instinct on a social level to coagulate in one area, at the same time, when the dog runs at them, they scatter in the following manner:



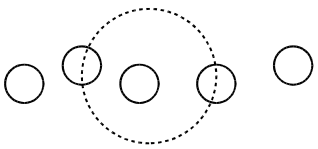
This is really unhelpful when it comes to keeping sheep in flocks, and unlike the real world, where sheep in a flock have a tendency to flee from a dog in the same direction, thus:



At the very least, one might expect real sheep to keep some coherent formation. Obviously, some of the skill involved for any sheep dog (real or electronic) is in maintaining the integrity of the flock. The obvious way to proceed in giving the dog a concept of the flock is to make it aware when one of two things happen. Firstly, when a sheep wanders more than a certain distance away from any other sheep, the dog might change its behaviour so as to direct it back.



Secondly, the dog must try to maintain the flock in some coherent formation: having the sheep spread out in a line is not useful when it comes to directing them into the pen. Thus, we can imagine that a circle might be constructed such that all sheep should be within the circle. The radius of the circle could be, say, a third of the width of the mouth of the pen:



The difficult question concerning this circle, however, is always going to be how to decide where to put the centre. Centring the circle upon one particular sheep will always be counterproductive if that sheep wanders to the edge of the circle, and the fluidity of the flock will make it hard to pinpoint any constant centre point.

Once a coagulation of the flock has been achieved, the directing of the sheep would just be an extension of the way in which directing occurs with one sheep. This is clearly possible, since human users generally manage to work out how to achieve the task in this fashion. However, the dog must maintain a greater distance from the pack, and take wider sweeps from left to right in order to retain coherence.

It seems likely that if such a method of herding is possible, then if will require far greater planning on behalf of the dog, and the relative position based solution that has currently been provided may well have to be scrapped.

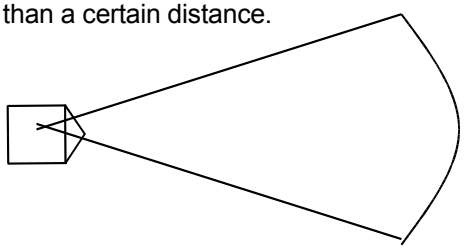
5.2 What could be done to make the dog more agent-like

As it stands, the agent is little more than an expert system, which displays few truly cognitive features. It is therefore useful and informative to ask how the agent might be extended to make it more realistic.

5.2.1 Updated Sensors.

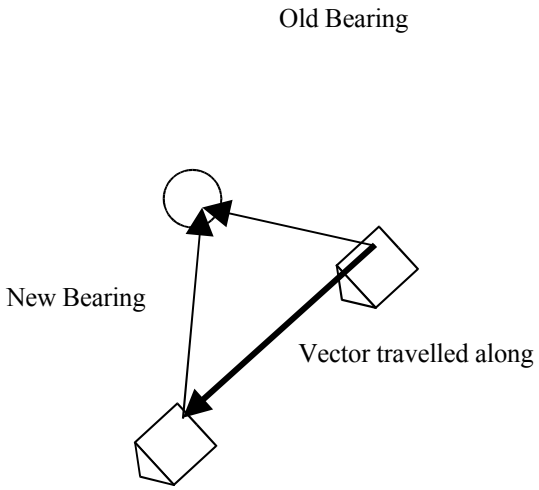
The first and most fundamental change that would have to be made would be to update the method used by the dog in sensing other objects in the world. At present, the dog is just instantly aware of all other agents, and of their positions and sizes. A number of changes could therefore be made in order to improve this situation.

Firstly, it would be good to make the dog only directly aware of agents in a sensible field of vision. That might include just those objects that are in the front 60 degrees of its surroundings and not further away than a certain distance.



This would almost immediately necessitate giving the dog some form of memory. If it can only keep track of a certain portion of its surroundings at any one time, it must be able to have some access to “where it last saw the *x*” in order to make the task even remotely possible. Continuing along this line renders the whole task much more complicated, because ultimately it is not really realistic to give the dog access to the co-ordinated map used by the simulation to display the world, rather the dog would have to build up it’s own ‘mental map’ of its surroundings, with some items thereon more or less pinpointable than others.

It is vital to have a constant notion of where the front posts are, for example, without having to be constantly updating its knowledge by “looking” at them. This can be done mathematically, by using an old bearing distance, and the vector travelled since the last reading:



Indeed this is the method which computer simulations lend themselves, because of the mathematical power of computers. However, it is extremely technical, and it seems unlikely that dogs go through such calculations in their heads.

The alternative (mental mapping) method, is much harder to simulate on a computer in this fashion, but seems more likely to bear some resemblance to how a dog really goes about its task.

Another factor which is unsuitable for modelling on computer is the ability to figure out distance and size of other objects in the world. In the real world, it seems likely that the third dimension that is not present on the screen provides extra data that can be made use of in these calculations (the way the height of a tree changes as the dog moves around the world, for example). Having said that, the dog does have access to *apparent* widths of objects (in terms of the bearing obscured by that object), which may well be used in exactly the same way.

Given these difficulties, totally realistic sensors are not really feasible in a simulation of this nature

5.2.2 Other motivations and Fatigue

The reason that the dog is purely an expert system without more agent-like qualities can be put down to the fact that rather than “evolving” in a world where it has to face competing demands on its time in order to survive (eating, sleeping, reproducing etc), it has only one explicit goal in it life: viz. to get the sheep into the pen. Providing artificial goals of such a nature that would introduce agent-like complexity into the dog is not an unfeasible task: one of the reasons that the sheep seem more realistic than the dog is that they do indeed face competing needs and desires.

Fatigue could easily be introduced by coding the dog to gradually slow as it goes about its task, and then making it have to stop altogether (and rest), if it slowed to a certain point. Hunger could be bought into the equation by providing the dog with artificial food objects that it had to fetch every often (although this would not be true to the reality of sheep dog trials!). The dog might be given variable levels of aggression that affected both the manner and efficiency with which it herded the sheep.

To the solution of the problem, however, these are superfluous, and while it may be desirable to consider adding them at some future point, without studying the real nature of sheep dogs they seem rather artificial.

One major way in which the agent could have been made more cognitive would have been to have included some notion of ‘learning’ the solution. This will be discussed in the next section.

6 How the Project Progressed, and How The Solution could have Been Learnt

6.1 Progression of the project

Essentially, the project progressed in reverse order of how the behaviours naturally occur in one run of the simulation. That is to say, first of all it was figured out how to steer the sheep into the pen, then how to take it to the front and finally how to reach the sheep from any point on the world. Almost as an afterthought, the avoidance of trees was added at the end, by which time little thought was required as to how to go about applying a solution to this problem. However, before all of these could progress, it was necessary to discover how the fundamental laws of directing a sheep could work.

6.1.1 Fundamental 'laws'

How to direct the sheep in a particular direction was discovered, almost by accident, at a very early stage of the project. An artificial target was set up, with the aim of getting the dog to take a sheep to the target.

The initial solution was simply to head towards the sheep whenever it was on the same heading as the target + or – 10, and otherwise simply to head 45 degrees to its left. This could be coded in as little as two rules. Although this solution was enormously inefficient, and often involved travelling around in circles before a correct line could be found, it actually did complete the task.

From here it was a short step to realising that that the solution could be made more efficient if a third rule was added so that when the dog was off target, it was always forced in the right direction. I.e. if the bearing of the sheep was to the right of the bearing of the target then the dog should head to the right of the sheep and the same for the opposite situation.

These rules, however were all very well for a single pinpoint target, where the nature of the range of bearings which the sheep should head for did not change as the dog and sheep moved around the world. In order to get the sheep into the pen, it had to be born in mind that the pen was very different in nature to the artificial target that had been set up.

For one thing, it could only be approached from one direction: it was essential to get the dog to realise when the sheep could be directly herded, and when not. For this purpose, clearly, different sets of laws were going to be necessary for different types of situations.

6.1.2 Application of Laws

The first part of the problem to be solved was that of steering the sheep into the pen. All it required, originally, was to realise when the sheep was in a suitable position to be herded, and when not. As mentioned, it turned out to be in a suitable position only when its heading was between the headings of the two posts (importantly, only when the posts' headings were in the correct order ie the heading of the left post was less than the heading of the right one). At this stage, therefore, the relationships between the dog, the sheep and the pen needed only to be divided up into three kinds: Where the sheep was on target, where it was to the left of target and where it was to the right of target.

Dividing the area around the dog into five or more sectors, each with its own rule associated with it occurred later, as it turned out that the task getting the dog to the front of the pen could not be accomplished with only three such rules. The number of sectors was made constant because that way the dog could constantly divide up its perceptions in the same fashion, changing only its reactions to the divisions. Moreover, it turned out to be useful that one could vary the amount of steering depending upon whether the sheep was between the dog and the pen entrance or not.

Getting the dog to veer around obstacles was really a case of trial and error: the theory behind the problem was obvious. If there is an obstacle in the way it must be veered around. What needed to be worked out was when there was an obstacle to veer around (which was a matter of tuning the mathematics), and how to veer around it, which was a very similar problem. The

only thing which needed to be figured out was that it was necessary to keep the obstacle being veered around constant, and when to decide that the obstacle had been veered around.

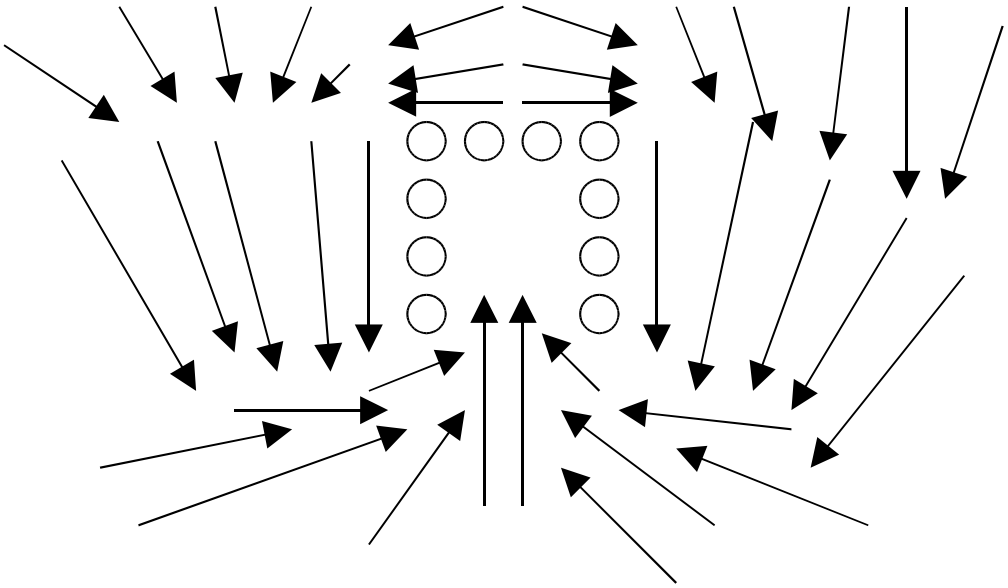
6.2 Learning the Laws

The question of how an agent might learn these laws is partly pointless: no-one would suggest that a real dog has to figure out how to veer around an obstacle, it is instinctive. Presumably there may have been evolutionarily early species which constantly walked into things they were trying to avoid, but it is part and parcel of the properties of being capable of rapid motion and of having a working visual system that one gets this right a large proportion of the time. Presumably, however, dogs do have to learn how to herd sheep into pens, and moreover there is a distinctive process involved in how the human user of the original program comes to understand how to perform the task. Typically, this will start by aiming straight at the sheep, and watching them scatter, and proceed by gradually refining the technique of sweeping along behing them so as to get them to move in the desired direction.

A human user, however has a built in concept of how the overall nature of the task will unfold: he or she knows what the nature of the pen demands. A dog agent that is fully capable of learning the problem would have to learn the properties of the pen.

On one level it is easy to see how an electronic agent might proceed when learning this task: whether what follows is an accurate description of this process is obviously debatable.

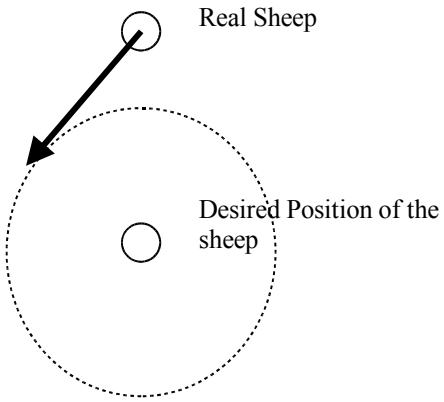
The task can ultimately be split into two mappings that have to be learnt: on the one hand, the agent would have to learn the relationships between actions it performs and resulting actions from the sheep, on the other, it must learn a series of relationships between positions which the sheep is in in relation to the pen, and directions which it must move, something like this:



On the face of it, it is not hard to see how such a mapping might easily be learnt, given enough time exploring the technical possibilities of the sheep world, since it is after all an albeit complicated series of associations between points in space and movement vectors. Certain regularities in the world however, suggest that there may be short-cuts that could be taken. For example, since the world is vertically symmetrical, it might be possible to take what one knows about one side and apply it to the other, by reflection. Similarly, once one knows that points in a certain area ought to point into the pen, one can infer a whole collection of the movement vectors. Abstractions like these suggest that simple associative learning may not be the most efficient way to figure out this task: it is certainly not the way a human goes about it.

How the dog might learn the relationship between its movement and that of the sheep could proceed as follows. The dog has only four cues as to how it should move: the distance and bearing of the sheep, and the distance and bearing of the sheep's desired position. The first thing it might realise is that moving when a certain distance away from the sheep produces no

effect. It might then try moving so as simply to reduce the distance between the actual and desired positions of the sheep. In other words, if a certain type of movement reduces this distance, then repeat it. The problem with this is that it can produce extremely inefficient paths for the sheep. Consider this:



In this example, the movement of the sheep causes it to get closer to the desired position, until the point where it touches a certain distance circle. However, the movement is clearly not the best way to get the sheep into its desired position. The reaction to this might be on the dogs part to move so as to maximise the decrease in distance between the sheep and the sheep's desired position. It is however precisely in doing this that it will start to form an association between this maximisation in decrease and a minimisation of the difference between the bearing of the actual sheep and the bearing of the sheep's desired position. Which is the point from where the total solution can be learnt.

The real difficulty for the dog in learning how to complete the task would be that it has to learn these two different mappings simultaneously, which is the main reason why it is hard to see how this learning procedure could be implemented.

7.Conclusion

Since most of the program is, I hope fairly self-explanatory, I will finish by drawing some conclusions to the questions raised of a more philosophical nature at the start of the paper.

It seems clear that the program as it is avoids using any explicit search algorithm. This was, as was pointed out, desirable, because of the likelihood of combinatorial explosion in possible search states. That is to say that since the number of possible “next states” from any one start state is extremely large, the amount of processing that the dog has to do in order for it to accuately plan a course for the sheep would be highly debilitating. The tradional solution to this problem is to implement useful generalisations within the agent's search architecture.

The current solution might easily be seen as a set of implicit generalisations as to what course the sheep should be induced to take around the worldwid. Indeed, th egeneralisations are tuned to the extent that the act of searching becomes superfluous. Fiven a state of the world, the dog instinctively knows what kind of next state is desirable (one where the sheep is at the bearing along which it ought to head), and at the same time how it ought to move in order to bring that next stat into being.

One conjecture that might be drawn is that this kind of solution is a useful one for whenever an agent has to navigate around simple worlds and that search algorithms only become advantageous when the world acquires a degree of complication. For example, when navigating the London Underground, a search algorithm will no doubt increase the efficiency of the task. However, if one wants to walk around a table that stands in one's path, it is doublessly easier to rely on implicit situation-based knowledge regarding the relationship between oneself and the table: there is no need to make a plan.

Moreover, when it comes to talking of learning the solution, I believe it is easier to see how a dog might come to learn the relationships as described above, than it is to see how the dog might learn the state space of the problem sufficiently to make any plan reasonable.

One suggestion that springs to mind is that using a search algorithm becomes more useful when the agent is forced into making higher level abstractions about the world. In this situation, the conceptualisation of the world into different parts and types of parts facilitates the use of searching by reducing the size of potential combinatorial explosions. When as in the case of the sheep dog, the world is instantly accessible to the agent, there is no need to make high level abstractions, and hence the type of solution illustrated here is more efficient.

Bibliography

Sharples et al. **Computers and Thought**

Nilsson **Artificial Intelligence**

Gardner **The Mind's New Science**

Johnson-Laird **The Computer And The Mind**

Teach Poprulebase, Objectclass, Search, Sim_sheep, sim_agent, sim_feelings

Sloman **Why can't a goldfish long for its mother? Architectural pre-requisites for various types of emotions**

Sloman **Architectural Requirements for Human-Like Agents both Natural and Artificial**