Journal of Robotics & Autonomous Systems (RA&S), special issue on Field & Service Robotics

AN ARCHITECTURE FOR DISTRIBUTED COOPERATIVE-PLANNING IN A BEHAVIOUR-BASED MULTI-ROBOT SYSTEM

David Jung and Alexander Zelinsky

Robotic Systems Laboratory,
Department of Systems Engineering
Research School of Information Sciences and Engineering,
The Australian National University,
Canberra, ACT 0200, Australia
http://wwwsyseng.anu.edu.au/rsl/

Abstract

The *Architecture for Behaviour-Based Agents* (ABBA) is an architecture designed to illustrate that situated agents can exhibit sophisticated planning while retaining reactivity, without resorting to hybrid architectures. In particular, unified planning for spatial and topological navigation, cooperation and communication can be achieved using an appropriate action-selection scheme. Joint-planning of cooperative behaviour in a multi-robot system arrises as a natural consequence of considering cooperative planning as an extension of the action-selection problem facing individual agents. This paper describes ABBA and illustrates the efficacy of our approach by presenting a solution to a cooperative cleaning task with two autonomous mobile robots.

Keywords: ABBA, Cleaning, Action selection, Distributed planning, Navigation, Cooperation, Communication

1. INTRODUCTION

Research into multi-robot systems is driven by the assumption that multiple agents have the possibility to solve problems more efficiently than a single agent does. Agents must therefore cooperate in some way. There are many tasks for which a single complex robot could be engineered; however, in many cases there are advantages to using multiple robots. A multi-robot system can be more robust because the failure of a single robot may only cause partial degradation of task performance. In addition, the robots can be less complex since each is only responsible for partial fulfilment of the task. Our philosophy is to design heterogeneous multi-robot systems where appropriate.

Most approaches to cooperation in multi-robot systems consider the control of individual robot behaviour separately from the cooperative group behaviour. The cooperative robotics community often places an implicit artificial divide between designing the behaviour of a single agent and design of group behaviour. Some systems use planners for individual behaviour, but design group behaviour to be emergent, or vice versa (eg. [Noreilis, 1992; Le Pape, 1990]). Other schemes use different planning mechanisms for individual and group action selection, such as employing plan merging or negotiation to modify the plans of a single agent to accommodate the global goals (eg. [Alami et al., 1995; Dudek et al., 1995; Heikkilä and Matsushita, 1994]). Designing both individual and group behaviour to be emergent is common in the collective robotics community (eg. [Parker, 1998; Matarić, 1995; Kube and Zhang, 1994; Arkin and Hobbs, 1992b]). We believe that, not only can situated agents have goals, but they can also exhibit a sophisticated planning ability, at both individual and cooperative levels. This can be achieved in behaviour-based systems while retaining reactivity, in a unified manner and without the need for hybrid systems.

Little research considers cooperative planning as an extension of the *action-selection problem* facing individual agents. We have developed our ***Architecture for Behaviour-Based Agents*** (ABBA) to provide an action selection substrate that can naturally accommodate cooperation. In particular, ABBA provides a distributed planning capability, that in conjunction with task specific mechanisms can achieve cooperative joint-planning and communication in heterogeneous multi-robot systems. To drive the design requirements and to illustrate the efficacy of our approach, we have used ABBA to implement a solution to a concrete task with real mobile robots.

This paper first describes the ABBA architecture and the mechanisms we developed for spatial and topological navigation, cooperative planning and communication. This is followed by a description of the component behaviours and a solution to the task that uses ABBA and afore mentioned mechanisms.

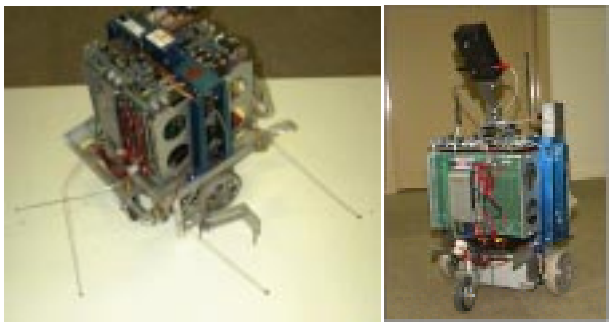

Figure 1 – The two Yamabicos 'Flo' and 'Joh'

2. COOPERATIVE CLEANING

The task chosen was for our two autonomous mobile robots to clean the floor of our laboratory. The 'Yamabico' robots [Yuta et al., 1991] shown in Figure 1 are heterogeneous in the sense that each has different tools and sensors such that neither can accomplish the task alone. In fact, the task was contrived so that this was the case.

One of the robots, *'Joh'*, has a vacuum cleaner that can be turned on and off via software. Joh's task is to vacuum piles of litter from the laboratory floor. It cannot vacuum close to walls or furniture. It has the capability to 'see' piles of litter using a CCD camera and a video transmitter that sends video to a *Fujitsu MEP tracking vision system*. The vision system is capable of landmark-based navigation and can operate safely in dynamic environments at speeds up to 600mm/sec [Cheng and Zelinsky, 1996]. The vision system uses template correlation, and can match about 100 templates at frame rate. The vision system can communicate with the robot, via a UNIX host, over radio modems.

The other robot, *'Flo'*, has a brush tool that is dragged over the floor to sweep distributed litter into larger piles for Joh to pick-up. It navigates around the perimeter of the laboratory where Joh cannot vacuum and deposits the litter in open floor space.

The task is to be performed in a real laboratory environment. Our laboratory is cluttered and the robots have to contend with furniture, other robots, people, opening doors, changing lighting conditions and other hazards.

3. THE ABBA ARCHITECTURE

We have divided the implementation of our cooperative robot system into two parts. ABBA itself is a task independent substrate that supports learning, action selection and iconic and indexical reference in an integrated manner. It serves as a framework within which simple behaviour and other more sophisticated mechanisms can be embedded – such as navigation, planning, cooperation and communication. The second part consists of the task specific behaviour and afore mentioned mechanisms. This section introduces ABBA in broad terms. An explanation of how these higher-level mechanisms were implemented using ABBA is given toward the end of the paper, after describing some low-level behaviour.

3.1 Action selection

One of the first problems we need to address in an architecture is how we will answer the "*what do I do next?*" question – the *action selection problem*. We needed to design an action selection mechanism that is distributed, grounded in the environment (situated), and employs a uniform action selection mechanism over all behaviour components. Because the design was undertaken in the context of cooperative cleaning, we also required the mechanism to be capable of cooperative behaviour, navigation and communication. Each of these requires some ability to plan. This implies that the selection of which action to perform next must be made in the context of which actions may follow – that is, within the context of an ongoing plan. In order to be reactive, flexible and opportunistic, however, a plan cannot be a rigid sequence of pre-defined actions to be carried out. Instead, a plan must include alternatives, have flexible sub-plans and each action must be contingent on a number of factors. Each action in a planned sequence must be contingent on internal and external circumstances including the anticipated effects of the successful completion of previous actions. Other important properties are that the agent should not stop behaving while planning occurs and should learn from experience.

There were no action selection mechanisms in the literature capable of fulfilling all our requirements. As our research is more concerned with cooperation than action selection per se, we adopted the most appropriate mechanism and modified it to suit.

Around 1990, Pattie Maes proposed a unique solution to the action selection problem that satisfies many of our criteria [Maes, 1990a]. Her theory "*models action selection as an emergent property of an activation/inhibition dynamics among the actions the agent can select and between the actions and the environment*". We have adapted and improved her spreading activation algorithm for the action selection mechanism in ABBA.

3.2 Components and Interconnections

The behaviour of a system is expressed as a network that consists of two types of nodes in ABBA – *Competence Modules* and *Feature Detectors*. Competence modules (CMs) are the smallest units of behaviour selectable, and feature detectors (FDs) deliver information about the external or internal environment. A CM implements a component behaviour that links sensors with actuators in some arbitrarily complex way. Only one CM can be executing at any given time – a winner-take-all scheme. A CM is not limited to information supplied by FDs – the FDs are only separate entities in the architecture to make explicit the information involved in the action selection calculation.

The graphical notation is shown below where rectangles represent competence modules and rounded rectangles represent feature detectors. Although there can be much exchange of information between CMs and FDs the interconnections shown in this notation only represent the logical organisation of the network for the purpose of action selection.

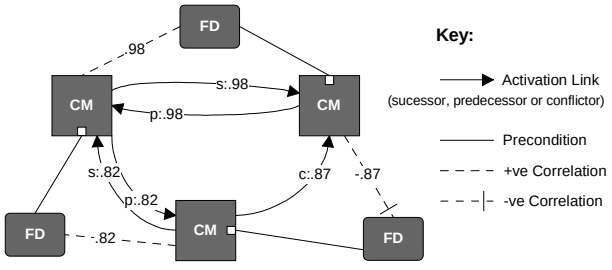

Figure 2 - ABBA Network components and interconnections

Each FD provides a single *Condition* with a confidence that is continuously updated from the environment (sensors or internal states). Each CM has an associated *Activation* and the CM selected for execution has

the highest activation from all *Ready* CMs whose activations are over the current global threshold. A CM is *Ready* if all of its *preconditions* are satisfied. The activations are continuously updated by a *spreading activation algorithm.*

It is important to note that although ABBA seems to make an arbitrary Cartesian style division between sensing and acting (FDs and CMs), that this is not necessarily so. Feature detectors can deliver conditions based in the internal state of CMs as well as conditions based on sensors. This is analogous to saying that CMs can operate by effecting FDs as well as actuators.

Care should be taken to ensure feature detectors are written to deliver information from sensors as directly as possible, rather than from any internal representation of an anthropomorphic category.

The network designer needs to be mindful that the dynamics of a multi-robot/environment system has no a priori boundaries. The boundaries can be redrawn as appropriate for thinking about the system dynamics to include arbitrary portions of robot and environment behaviour. A single network may describe part of a robot-environment interaction, or possibly a whole multi-robot-environment system.

The system behaviour is designed by creating CMs and FDs and connecting them with *precondition links*. These are shown in the diagram above as solid lines from a FD to a CM ending with a white square. It is possible to have negative preconditions, which must be false before the CM can be *Ready*. There also exist *correlation links*, dotted lines in the figure, from a CM to a FD. The correlations can take the values [-1…1] and are updated at run-time according to a learning algorithm. A positive correlation implies the execution of the CM causes, somehow, a change in the environment that makes the FD condition true. A negative correlation implies the condition becomes false. The designer usually initialises some *correlation links* to bootstrap learning.

Together these two types of links, the precondition links and the correlation links, completely determine how activation spreads thought the network. The other *activation links* that are shown in Figure 2 are determined by these two and exist to better describe and understand the network and the activation spreading patterns. The activation links will feature in the description of the *spreading activation algorithm* in the following section, and are determined as follows.

- There exists a *successor link* from CM *p* to CM *s* for every FD condition in *s*'s preconditions list that is positively correlated with the activity of *p*.
- There exists a *predecessor link* in the opposite direction of every successor link.
- There exists a *conflictor link* from CM *x* to CM *y* for every FD condition in *y*'s preconditions list that is negatively correlated with the activity of *x*.

The successor, predecessor and conflictor links resulting from the preconditions and correlations are shown in Figure 2.

In summary, a CM *s* has a predecessor CM *p*, if *p*'s execution is likely to make one of *s*'s preconditions true. A CM *x* has a conflictor CM *y*, if *y*'s execution is likely to make one of *x*'s preconditions false.

3.3 The Spreading of Activation

Before a rigorous description of the spreading activation algorithm, we'll present an overview with an example to get a feel for how it works. The figure below shows a toy example from Maes' paper, re-expressed as an ABBA network.

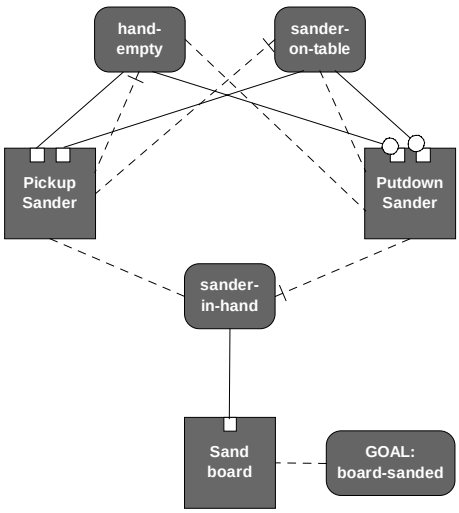

Figure 3 – Maes' Sand Board example in ABBA

The goal is for a board to be sanded using a sander. The possible actions (CMs) are picking up or putting down the sander and sanding the board. The FDs sense three conditions from the situation – whether the sander is on the table or in hand and if the hand is empty.

The figure only shows the precondition (solid) and correlation links (dashed). Only the precondition links must be assigned during the network design. Although the correlations are often initialised to sensible defaults, in theory they could be learnt from experience by the algorithm. While there are many links in the notation, they are quite straightforward. Clearly, `sand-board` requires `sander-in-hand` as a precondition, with which `pickup-sander` is correlated. `Putdown-sander` requires that both `hand-empty` and `sander-on-table` are false as preconditions and is correlated with both. `Putdown-sander` is also negatively correlated with `sander-in-hand`, and so on. The goal is also explicitly represented.

The scheme for spreading the activation follows the algorithm proposed by Maes. The spreading activation algorithm injects activation into the network via goals and via FDs whose conditions are true. Therefore, as `sand-board` can satisfy the goal – according to the correlation link – its activation is increased by the goal. CMs cannot execute if not *Ready* – if one or more preconditions hasn't been met. A CM that is not ready increases the activation of those CMs that can satisfy any of its preconditions (and decreases activation of those that can undo any of its preconditions). The quantity depends on its own activation level and total activation is not conserved. In this case, `sand-board` will increase the activation of

`pickup-sander` and decrease the activation of `putdown-sander` – as they are positively and negatively correlated with `sander-in-hand`, respectively. A CM with its preconditions satisfied feeds activation forward instead – so `pickup-sander` increases the activation of `sand-board`. However, `pickup-sander` will very quickly be the first CM to reach the activation threashold and, having the highest activation, will be selected for execution. Upon execution, a CM's activation is reset to zero. Hence, sand-board will be next in line for execution. The algorithm also increases the activation of CMs that have true preconditions – so the network favours CMs with preconditions matching the current situation.

Rather than following all these links around, the flow of activation can be seen more clearly using the activation links. Recall that the activation links are wholly determined by the preconditions and correlation links. Figure 4 shows the activation links – except for successor links. There is one successor link for every predecessor link, but in the opposite direction.

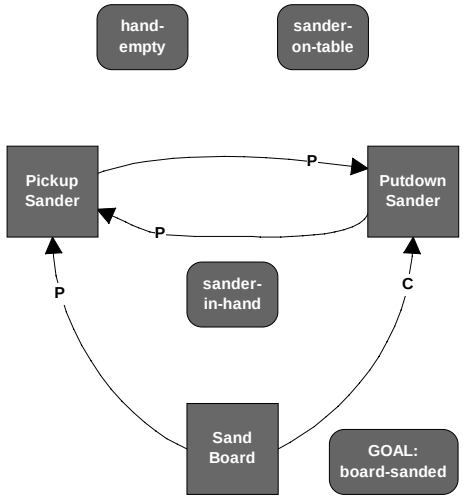

Figure 4 - Sand Board example - activation links

The activation rules can be more concisely described in terms of these activation links. The main spreading activation rules can be simply stated:

- *Unready* CMs increase the activation of predecessors and decrease the activation of conflictors, and
- *Ready* CMs increase the activation of successors.

In addition, these special rules change the activation of the network from outside in response to goals and the current situation:

- Goals increase the activation of CMs that can satisfy them and decrease the activation of those that conflict with them, and
- True FDs increase the activation of CMs that have them as a precondition.

There can be only one executing CM at any given time – the CM with the highest activation over a threshold, that is *ready*, is selected. After a CM completes execution, its activation is reset to zero.

From the rules we can imagine activation spreading backward through a network, from the goals, through CMs with unsatisfied preconditions via the precondition links until a *ready* CM is encountered. Activation will tend to accumulate at the ready CM, as it is feeding activation forward while its successor is feeding it backward. Eventually it may be selected for execution, after which its activation is reset to zero. If its execution was successful, the precondition of its successor will have been satisfied and the successor may be executed (if it has no further unsatisfied preconditions). We can imagine multiple routes through the network, activation building up faster via shorter paths. These paths of higher activation represent 'plans' within the network. The goals act like a 'homing signal' filtering out through the network and arriving at the current 'situation'.

One important different between ABBA and Maes' networks is that in ABBA the activation links depend on the correlations – which are updated continuously at run-time according to previous experience. Maes' networks are static. If, for example, repeated activation of `pick-up-sander` failed to cause the condition `sander-in-hand`, then the correlation between them would be eroded eventually eliminating the predecessor link. Hence, activation would flow via a different route to satisfy the goal (although there are no other alternatives in this example). The mechanism that 'learns' the correlations is detailed below. Another difference is that the quantity of activation spread between CMs depends not only on the network parameters, as in Maes' networks, but also on the strength of correlations and the confidence of conditions. A detailed specification of the rules and the algorithm follows.

The Rules

The system proceeds in discrete time-steps. At each step some activation is injected into the system, removed from the system, and re-distributed within the system according to the rules below. There are a number of global parameters used to tune the dynamics of the system:

- π - The mean level of activation
- θ - Threshold for becoming active (CM becomes active, if ready and $A > \theta$)
- γ - Activation injected by a goal to be achieved
- δ - Activation removed from conflictors to goals that need to remain achieved
- ϕ - Activation injected by a feature detector whose condition is true ($C > T$)
- T - The confidence threshold. A condition with confidence $c > T$ is considered true.
- R - The correlation threshold. A correlation coefficient $c > R$ is considered positively correlated.
  (and $c < -R$ is considered negatively correlated)

The first three rules determine how the network is activated and inhibited from external sources, such as the current situation as perceived by the set of FDs that output conditions, and the global goals of the agent.

1. ACTIVATION BY THE SITUATION

Feature detectors (FDs) that output a condition *c* spread activation to any CM whose precondition set contains *c*, if *c* is true. The activation sent to a CM is

(C.φ)/n where n is the number of CMs whose precondition sets contain *c*, and C is the confidence of the condition. When a CM receives activation from a FD it is divided by the number of conditions in it's precondition set.

2. **ACTIVATION BY GOALS**
   An external goal is represented by a condition *c* (as output by a FD) that must be achieved. There are two types of goals, *once only goals*, which need only be achieved once, and *permanent goals*, that once achieved need to be maintained.
   A goal increases the activation of the CMs that are correlated with it's condition *c* by *(W.γ)/n*, where n is the number of CMs activated by this goal, and W is the correlation between any particular CM and the condition *c*. When a CM receives activation from a goal it is divided by the count of predecessor links and activating goals for this CM, except that predecessor links or activating goals that share their defining condition are counted only once for each condition.

3. **INHIBITION BY PERMANENT GOALS**
   A permanent goal is an external goal that once achieved, must remain achieved. A goal inhibits CMs that are negatively correlated with it's condition *c* by *(W.δ)/n*, where *n* is the number of CMs inhibited by this goal, and W is the negative of the correlation between any particular CM and the condition *c*. When a CM receives inhibition from a permanent goal it is divided by the count of conflictor links and inhibiting goals for this CM, except that conflictor links or inhibiting goals that share their defining condition are counted only once for each condition.

The next three rules determine how activation is spread within the action selection network. They are analogous to the preceding three rules in the following manner. If a CM *p* is a predecessor of a CM *s*, then *s* treats *p* as a sub-goal by feeding activation backward to *p* until the condition in *s*'s precondition set to which *p* is correlated becomes true, as long as s is inactive.

If a CM *p* is ready or active, then it feeds activation forward to all successor CMs whose precondition sets contain a condition *c* to which *p* is correlated, as long as *c* is false. This predicts or primes the successor CMs to be ready for when the CM *p* achieves it's result and becomes inactive again. It is analogous to activation by the current situation.

A CM *x* will inhibit all conflictors for which there exists a negatively correlated condition c which is in the precondition set of *x*, as long as *c* is true. This is essentially treating conflictors as if they are permanent goal conflictors of CM *x*'s preconditions.

4. **ACTIVATION OF SUCCESSORS**
   A ready or active CM *p* sends activation forward to all successor CMs *s* for which the defining condition *c* to which *p* is correlated (which is in the precondition set of *s*), is false (the FD condition *c*'s confidence C < T). If A is the activation of *p*, then the activation sent to a successor CM *s* is *(A.(φ/γ).W)/n*, where W is the successor link weight (correlation of *p* to *c*) and *n* is the number of successor links from *p* for condition *c*. When a CM *s* receives activation from a predecessor CM, the activation is divided by a count of the number of successor links to *s*, except that successor links that share their defining condition are counted only once for each condition.

5. **ACTIVATION OF PREDECESSORS**
   An inactive CM *s* sends activation backward to all predecessor CMs *p* for which the defining condition *c* to which *p* is correlated (which is in the precondition set of *s*), is false (the FD condition *c*'s confidence C < T). If A is the activation of s, then the activation sent to a predecessor CM *p* is *(A.W)/n*, where W is the predecessor link weight (correlation of *p* to *c*) and n is the number of predecessor links from *s* for condition *c*. When a CM *p* receives activation from a successor CM, the activation is divided by the count of predecessor links and activating goals for this CM, except that predecessor links or activating goals that share their defining condition are counted only once for each condition.

6. **INHIBITION OF CONFLICTORS**
   A CM *x* inhibits all conflictor CMs *n* for which the defining condition *c* to which *n* is negatively correlated (which is in the precondition set of *x*), is true (the FD condition *c*'s confidence C > T), and there is no inverse conflictor link from *n* to *x* that would be stronger. If A is the activation of *x*, then *x* inhibits conflictor CM *n* by *(A.(δ/γ).W)/n*, where W is the conflictor link weight (negative the correlation of *n* to *c*) and *n* is the number of conflictor links from *x* for condition *c*. When a CM *n* receives inhibition from a CM with which it conflicts, it is divided by the count of conflictor links and inhibiting permanent goals for this CM *n*, except that conflictor links or inhibiting goals that share their defining condition are counted only once for each condition.

Learning

The network can learn from experience by modifying the flow of activation by change the correlation links at run-time. The mechanism for adjusting the correlation between a given CM-FD pair is simple. Each time the CM becomes active, the value of the FD's condition is recorded. When the CM is subsequently deactivated, the current value of the condition is compared with the recorded value. It is classified as one of*: Became True*, *Became False*, *Remained True* or *Remained False*. A count of these cases is maintained *(B_t, B_f, R_t, R_f)*. The correlation is then:

$$corr = \frac{(2B_t + R_t)}{2N} - \frac{(2B_f + R_f)}{2N}$$

Where the total samples $N = B_t + B_f + R_t + R_f$

At each update the counts are decayed by multiplying with $N/(N+1)$ so that recent samples have a greater effect than

historic ones. This keeps the network plastic.

The Algorithm

The action-selection mechanism proceeds by iterating the above spreading rules and selecting the CM with the highest activation above the threshold θ, from the set of *ready* CMs. A CM is *ready* if all of its preconditions are satisfied. If there are no *ready* CMs above the threshold then the threshold is decreased by 10%. The correlations between CMs and FDs are also updated according to the equation above when the active CM changes.

Maes has shown that with these rules the network exhibits a planning capability. The amount of goal-oriented verses opportunistic behaviour can be tuned by varying the ratio of γ to ϕ. This spreading activation results in CMs being selected according to a current plan, which is represented by the current activations of the CMs. Activation builds up along the path of CMs that lead to the goal because of the spreading forward from the FDs representing the current situation, and spreading backward from the goals. Conflicting goals or sub-goals inhibit each other via the conflictor rules. There may be multiple paths between a current situation and a goal, but the next appropriate behaviour (CM) on the shortest path will obtain greater activation first. This allows for contingency plans because if a CM fails to perform as expected the correlation between its execution and the expected outcome will fall until the next best plan comes into effect. The next best plan, or next shortest path from situation to goal, will already have been primed with activation. The network will never be caught in a loop performing ineffective behaviour if there are alternative solutions.

3.4 Example

The network in Figure 5 is an early implementation of Flo's litter sweeping and dumping behaviour as it appears on-screen. The solid lines represent the preconditions that are programmed to give the desired behaviour. The dashed lines represent correlations between the execution of a CM and its effect on the environment in terms of FDs. These are learnt during run-time, however it is useful to initialise them to speed learning, or to manually activate certain behaviours to force the robot into situations where the correlations will be recognised.

The activation links that result from the precondition and correlation links shown in Figure 5 are shown in Figure 6 below. For example, `Follow` is a successor of `ReverseTurn` (and hence the later is a predecessor of the former). Because when Flo is following along a wall its behaviour alternates between `Follow` and `DumpLitter`, these CMs feed activation forward to each other via successor links. Since `DumpLitter` has the `Timer` expiration as a precondition and `Follow` has this as a negative precondition (see Figure 5), the state of the Timer effectively alternates these behaviours. Remembering that a CM cannot become active until all of its preconditions are satisfied. `Follow` is also a conflictor of `DumpLitter` because `Follow` requires an `ObstacleOnLeft` – a wall to follow, and `DumpLitter` is negatively correlated with this condition because it drives the robot away from the wall.

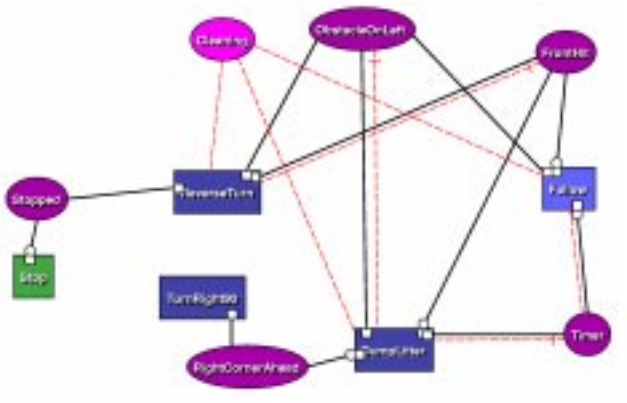

Figure 5 - Simple sweep and dump ABBA network

The network causes Flo to `Follow` walls hence sweeping up litter, until a fixed period `Timer` expires causing it to `DumpLitter` into a pile. It can also crudely navigate around corners and obstacles by repeated reversing and turning or stopping. The top-level goal is `Cleaning`. The `Stop` behaviour is activated as a hard-wired reflex upon `FrontHit` becoming true.

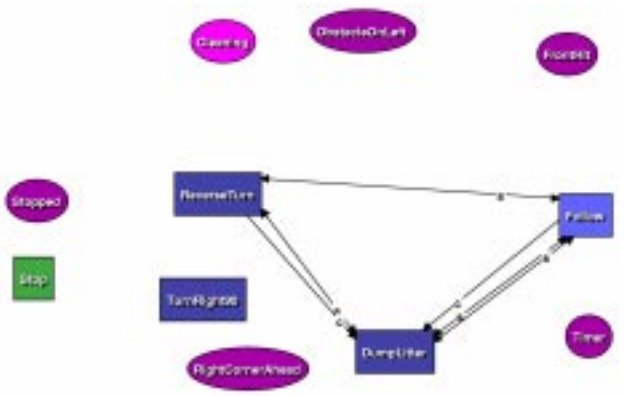

Figure 6 - Activation links of sweep and dump network

3.5 The Implementation

ABBA has been implemented as approximately 36000 lines of C++ code, including robot behaviours. Because code was developed to run on three different platforms, a *Platform Abstraction Layer* (PAL) was developed over which the rest of the system was layered. The PAL has been implemented over the VxWorks[1] operating system for use on our vision system, UNIX and the robot's custom operating system – MOSRA. The next layer provides an object-oriented framework for managing and interconnecting architecture units. The top layer enforces the particular paradigm – in this case the spreading activation rules and constraints on interconnecting FDs and CMs.

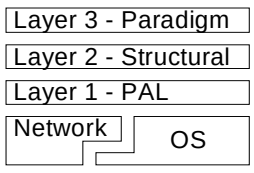

Figure 7 - ABBA Implementation architecture

[1] From Wind River Systems, Inc.

A Graphical User Interface was also developed to aid visualisation and the manual activation of behaviour sequences to help direct exploration and hence learning. In order to guarantee the real-time response of the network the feature detector conditions are updated at a fixed frequency (around 100Hz in our current implementation). The active CM is also iterated at this frequency. The spreading activation rules are applied asynchronously, so the effective frequency varies depending on the number of nodes and other processes running on the robot's main CPU.

4. NAVIGATION

Once we have a general action-selection scheme to plan behaviour, we need a method for using this to plan navigation. This section briefly describes the method we have developed.

There are two main approaches to navigational path planning. One method utilises a geometric representation of the robot environment, perhaps implemented using a tree structure. Usually a classical path planner is used to find shortest routes through the environment. The distance transform method falls into this category [Zelinsky et al., 1993]. These geometric modelling approaches do not fit with the behaviour-based philosophy of only using categorisations of the robot-environment system that are natural for its description, rather than anthropomorphic ones. Hence, numerous behaviour-based systems use a topological representation of the environment in terms only of the robot's behaviour and sensing (eg. see [Matarić, 1992]). While these approaches are more robust that the geometric modelling approach, they suffer from non-optimal performance for shortest path planning. This is because the robot has no concept of space directly, and often has to discover the adjacency of locations.

Consider the example below, where the robot in (a) has a geometric map and its planner can directly calculate the path of least Cartesian distance, directly from A to D. However, the robot in (b) has a topological map with nodes representing the points A, B, C and D and connected by a *follow-wall* behaviour. Since it has never previously traversed directly from A to D, the least path through its map is A-B-C-D.

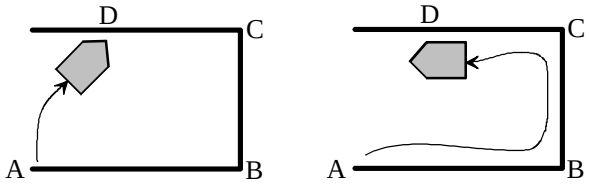

Figure 8 – (a) Geometric vs (b) Topological Path Planning

Consequently, our aim was to combine the benefits of geometric and topological map representations in a behaviour-based system using the ABBA framework.

4.1 Spatial representation

The scheme developed involves having feature detectors (FDs) to represent locations. The confidence of the FD condition relates to the certainty of the robot being at the represented location. How such FDs are implemented will be described shortly, but first consider that we have a FD for every location the robot will be. For example, distributions of FDs over the laboratory floor space. Because an accurate knowledge of the geometric location of the robot is unnecessary, a course resolution is sufficient. This will still require many FDs, hence we will also allow a non-uniform distribution of FDs over the floor, so that we can have a higher spatial resolution where required. Next, we interconnect each pair of neighbouring FDs (locations) with a behaviour that can drive the robot from one location to the other, as shown in the figure below.

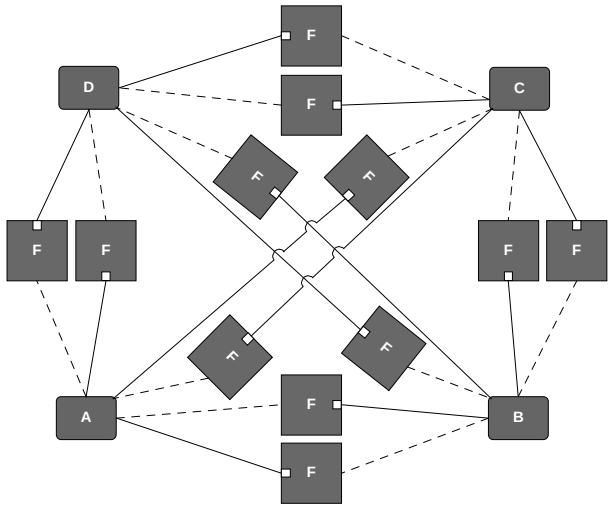

Figure 9 - Location FDs connected via 'Forward' CM

The location FD **A** is a precondition of a **F**orward CM that is correlated with **D**, through initialisation or exploration. This F CM drives the robot from the location that is its precondition to the location with which it is correlated. Hence if the robot is currently at location A and some other behaviour requires it to be at D, then activation will flow to this F CM both backward from the other behaviour and forward from A. This is due to rules 5 and 1 above. In this case F will become active and drive the robot from A to D. If an obstacle has been placed to obstruct the direct path from A to D, then the F behaviour would have failed. After a small number of failures, the correlation of F to D would be low enough that the F CM connecting A and B would receive greater activation. Hence the robot would drive from A to B and subsequently from B to D.

In practice, neighbours do not need to be fully connected initially as F CMs can be added at runtime. Exploratory behaviour can be engaged while recording the location of the robot each time a CM is initiated. If the CM successfully moves the robot to a new location, a new instance of the CM is created with the old location as precondition. The normal correlation calculations described previously will ensure this CM becomes correlated with the new location. Similarly, if a CM becomes uncorrelated with any FDs it can be removed from the network.

Now to how the FDs 'detect' the robot's location. Each location FD is associated with a node in a Kohonen self-organising-map (SOM) [Kohonen, 1990]. Each node in the SOM has an associated vector, where the elements are contain values representing the robot's sensory and behavioural state.

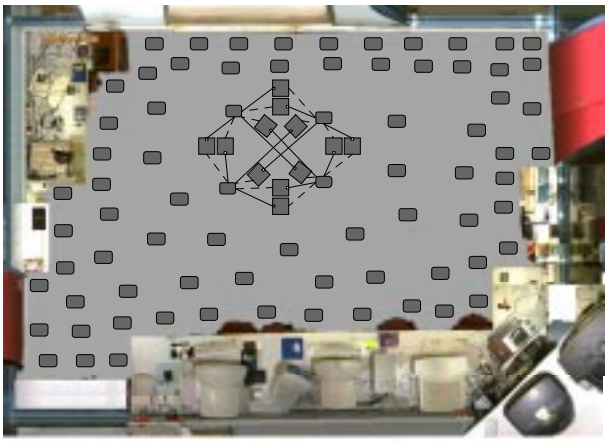

Figure 10 - SOM nodes distributed over the floor

The locomotion software on our Yamabico robots constantly delivers an estimated position and orientation of the robot in a global coordinate system. This is calculated from the wheel encoders and hence has a cumulative error. These odometry coordinates are elements of the SOM node vectors, along with other information such as ultrasonic range readings, whisker deflection and the currently active CM. So the SOM nodes are distributed over a high dimensional state space. The self-organisation of the SOM proceeds typically – by moving the nodes closer to observed states with an ever decreasing field of sensitivity. This works to distribute the SOM nodes over the space according to the probability distribution of observed robot states. As shown Figure 10, there are more nodes around the edge of the room where as Flo spends most of its time sweeping the perimeter (the figure only shows the CMs interconnecting four neighbours). SOM nodes with vectors differing significantly in the 2D-odometry coordinates are assigned to different location FDs. SOM nodes differing only in the other elements are assigned to the same location FD.

Therefore, a location FD's condition is set when its associated SOM node becomes activated by the current state. The vector elements are weighted in the distance calculation according to the importance of the corresponding sensor.

To overcome the indefinite accumulation of odometry error, we can utilise the fact that we can repeatedly detect landmarks in the environment whose position does not change. We define a *landmark* as a recognisable feature at a distinguishable location.

Note that the correlation calculations performed as part of the spreading activation algorithm will also cause FDs for significant landmark types to become correlated with the F CMs from Figure 9. For example, if Joh has a visual 'red door' feature detector, and there is a red door at location D, the F CM that drives the robot to the door will become correlated with the 'red door' FD. Since this CM is only correlated with one location FD, we have a correlation between a location and a landmark type FD. Hence, when both a landmark type FD and a location FD both have true conditions and are correlated with the same CM, we can assume we have detected a landmark. In this case, we adjust the robot's odometry and the odometry coordinates in the location FD's associated SOM node to a position between the current odometry reading and the SOM coordinates. The actual position depends on the confidence of the location FD's condition and the current error bounds on the odometry readings.

4.2 Topological representation

The above mechanism provides robust spatial path planning. Extending this to include a topological representation is simple. As mentioned previously, the **F**orward behaviour CMs can be added at runtime. By noting the location when a CM is activated and deactivated, and creating a new instance of the CM with the start location as a precondition and correlated with the final location. This is also performed for other CM types. For example, if the robot activates the *Follow* CM to wall-follow from A to C, then a new instance of the *Follow* CM is created with the location FD A as a precondition and correlated with the location FD C. Hence, a topological map connecting location FDs via behaviour CMs is built up. Because the wall follow link from A to C is shorter than the two links going via B, the spreading activation will favour this route (which still passes physically through location B in this case).

The mechanisms described above satisfy our aim of combining both spatial and topological map representations and path planning within the ABBA framework. What the networks as described so far cannot do, however, is represent an arbitrary location as a sub-goal within a larger over all plan – the destination must be 'hard-wired' during the design. To overcome this restriction ABBA networks must be capable of representing *deictic* references – which are generally useful.

4.3 Deictic Reference

From linguistics, *deictic* references (pronounced *dīk′tīk*) serve to point out or specify, as in the demonstrative pronoun *this* [American Heritage Dictionary, 1992]. As such, it is a type of flexible indexical reference. Indexical reference is a correlative association between icons. An icon is a set perceptual characteristics used to identify a class of objects. For example, the word 'red' is an association between how red objects look, how the written word looks, the sound of the spoken word and the necessary motor actions to utter it. In addition, there may be many other secondary associations involved in our representation – such as blood or fire trucks, for example. An architecture incapable of representing deictic references would lack flexibility. By adding this facility to ABBA we can represent references like '*the current location*', '*the door in front of me*', '*the last location I dumped litter*', etc. This need for deictic reference has been previously recognised in the literature. Rhodes introduced the notion of *pronomes* in behaviour networks that serve this function [Rhodes, 1995]. The *marker* concept was also conceived by Yuniyoshi *et. al.* to address similar limitations for their *Samba* architecture [Riekki and Kuniyoshi, 1996]. Markers ground task-related data on sensor data flow.

Deictic indexical references in ABBA are represented by using *proxy feature detectors*. This is essentially a FD that 'points to' another FD. The condition of the proxy FD reflects the condition of the FD to which it is currently

referring. Similarly, any CMs that are correlated with the proxy or have it as a pre-condition, behave as if they were correlated with or had the referred to FD as a pre-condition.

The 'top-level' goal in an ABBA network is a proxy FD that usually refers to the *Clean* goal. Goals are FDs whose conditions are always false. All the behaviour appropriate to achieving cleaning is correlated with the *Clean* goal. Hence, the action selection mechanism will take care of determining the specific sequences of these actions. This top-level goal is provided so the user can control the goal of the robot during development. For example, if the top-level goal were a proxy for a location FD, then the robot would navigate to the represented location and stop.

Proxy FDs are used extensively for indexical reference in some cleaning experiment network implementations. How they are used will become evident in the following section, which describes how behaviours implement cooperative cleaning and communication.

5. BASIC BEHAVIOUR

Thus far, we have described the ABBA substrate, how is supports planning and introduced the 'higher-level' navigation mechanism. Before proceeding to a description of how ABBA is used to support cooperative joint planning and communication, we shall describe some selected 'lower-level' behaviour. These component behaviours are used together in combination with the cooperation and communication mechanisms to implement a sophisticated solution to the cleaning problem.

One particular cleaning experiment involves visual observation of Flo by Joh, and communication between them. Specifically, Flo announces to Joh when it initiates the litter dumping procedure, and then communicates the relative position of the dumped pile of litter upon completion. Joh can observe Flo, and if Joh can see Flo when the announcement is made, then Joh can calculate the approximate position of the litter relative to itself. Joh then navigates to the location and visually looks for and servos on the litter in order to vacuum it. This experiment requires a number of simple behaviours and visual sensing capabilities, which will now be described.

5.1 Whisker based wall following

In order to sweep litter close to the walls, Flo needs a close wall following behaviour. We investigated a number of sensor technologies for this purpose, but finally we had to develop unique proportional whiskers [Jung and Zelinsky, 1996a]. Flo has two whiskers mounted on its left side for wall following and two whiskers in front for collision detection. The whiskers are also used for navigation (see Figure 1). The whiskers are contact sensors that give direct information about the distance between the robot and the wall being followed. The information from two whiskers is fused with odometry information using a Kalman filter to obtain an estimate of the robot's position and orientation relative to the wall. This is then fed into a standard Proportional Integral Differential (PID) controller to track along the wall. The navigation mechanism also uses the whiskers to detect landmarks, such as doors, corners, walls, poles etc.

5.2 Visual behaviour

Joh also needs to navigate reliably around the laboratory without colliding with obstacles, people or Flo, and it has the advantage of vision. A number of visual behaviours were required.

Free-space segmentation

We have implemented a visual free floor space detector using the real-time template matching capability of the Fujitsu vision system to segment the image into 'carpet' and 'non-carpet' areas.

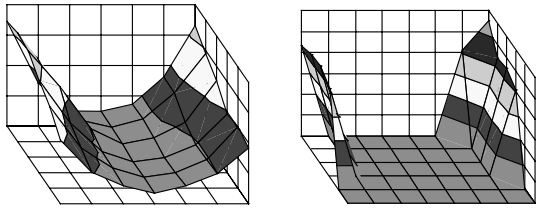

Figure 11 - Before and after normalisation and thresholding

The vision system delivers a correlation value for each template matched - the lower the value the better the match. A set of templates of the carpet in our laboratory is stored for matching. In Figure 12a, the smaller white squares indicate a better match. All values below a threshold signify free-space.

The CCD camera lens distorts the images and this effect can be seen in the correlation values, and must be compensated for using a normalisation procedure.

The first graph in Figure 11 shows the raw correlation values while looking at bare carpet. The normalisation consists of applying weights to these values that have been calculated by fitting a polynomial to the lens distortion during calibration, and then thresholding. The procedure also normalises for the average brightness of the image. The result can be seen in the graph on the right of the figure.

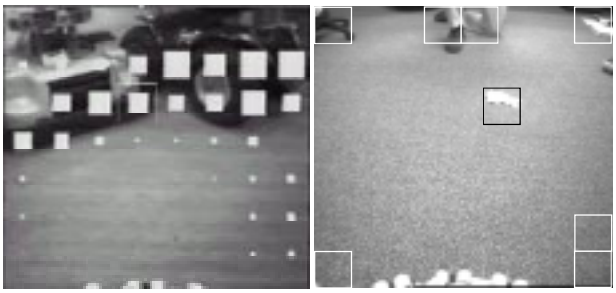

Figure 12 - (a) Obstacle avoidance (b) Interest operator

Although using template matching to match a texture such as carpet works poorly on single matches, at frame-rate and with robot motion, the stochastic behaviour is robust.

Once Joh has navigated to the approximate location of a pile of litter left by Flo, it has a vacuum behaviour that must visually locate the pile and servo on it in order to vacuum over it.

Interest Operator

Joh needs to identify piles of litter on the laboratory

floor in order to visually servo on them and vacuum over them. The vacuum is mounted under the robot so it must drive over the pile, which takes it out of view. Because a pile of litter doesn't have a definite shape, matching against a template is unlikely to locate it in the image. Hence we developed a simple 'interest operator' can locates isolated objects in the scene, with approximately the correct colouring. The interest operation primarily applies a zero-crossing convolution to the correlation values. The effect from the image in Figure 12b can be seen in the graph below.

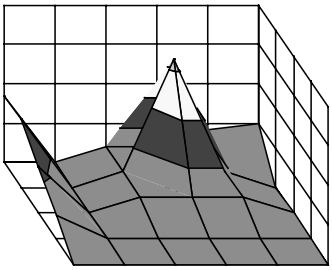

Figure 13 - Correlation values from 'interest operator'

In order to servo on the litter, a transformation from image coordinates to floor coordinates is performed and the PID controller directed to drive in the appropriate direction. Joh is also fitted with a bump sensor, which will trigger in the event that the behaviour erroneously servos on an obstacle on the floor, for example a book.

Visual Servoing

Joh has a behaviour that can visually detect and track Flo's motion. This behaviour servos on Flo to keep it visible and hence calculate the motion relative to Joh's coordinates. This information is used to deduce the approximate location of the dumped litter for the vacuum behaviour. There are two components to this behaviour: tracking Flo's image for visual servoing, and determining the 3D position and pose.

Flo has been marked with a unique rectangular pattern for tracking, as shown in Figure 14a below.

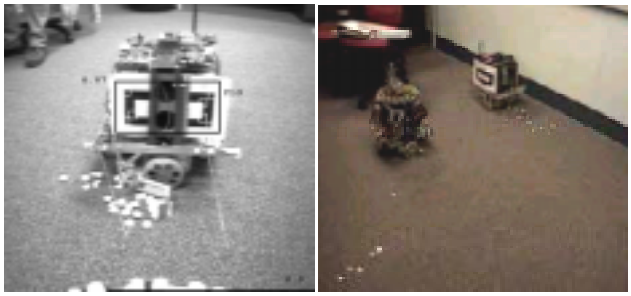

Figure 14 - (a) Flo from Joh's camera (b) Cleaning

Ten templates from the corners and sides of the rectangle are tracked. Due to changes in lighting, orientation and size, the templates would easily be lost. So a network of Kalman filters is used, one per template, to estimate the position of each from the vision system matching information and the position of the other nine templates [Jung et al., 1998a]. This results in tracking that is very robust to changes in scale and orientation.

Joh needs to know the relative location and pose of Flo in order to arrange a 'rendezvous'. The position and pose of Flo are computed using a projective transformation between the plane of the rectangular pattern marking on Joh and a model rectangular pattern marking in a known arbitrary plane. Four of the ten templates tracked on the pattern are sufficient to compute the projective transformation.

6. JOINT-PLANNING

Humans, as other primates, have the ability to co-construct plans with more that one interacting person, and flexibly adapt and repair them all in real time. We require this capability in ABBA to support sophisticated cooperative multi-robot solutions to the cleaning task. We have seen how ABBA exhibits a planning capability for selecting actions of an individual robot. This section explains how this mechanism can also exhibit a distributed planning capability for multi-robot cooperative systems.

Bond describes the construction and execution of joint plans in monkeys [Bond, 1996]. He defines a *joint plan* as a conditional sequence of actions and goals involving the subject and others. In order to achieve interlocking coordination each agent needs to adjust its action selection based on the evolution of the ongoing interaction. The cooperative interaction will consist of a series of actions - including communication acts. Each agent attempts different plans, assesses the other agents' goals and plans, and alters the selection of its own actions and goals to achieve a more coordinated interaction where joint goals are satisfied. Bond writes in reference to vervet monkeys, "*They are acutely and sensitively aware of the status and identity of other monkeys, as well as their temperaments and current dispositional states*".

Little research considers cooperative planning as an extension of the action selection problem facing individual agents. We believe that as ABBA provides a planning capability for the action selection of the individual, it can naturally accommodate distributed joint planning of cooperative actions for a multi-robot group, without modification.

Consider the following very simple hypothetical situation to illustrate how ABBA achieves distributed planning. Suppose we have a task requiring an ordered sequence of actions to be performed – such as making a cup of instant coffee. Each action has a specific effect on the environment that can be sensed – hence must be detected before the next action can be performed.

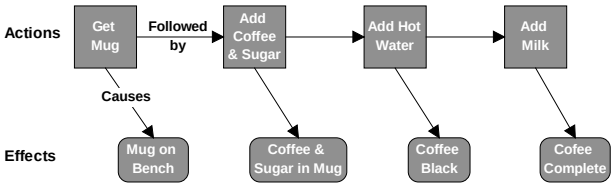

Figure 15 - Sequence of actions and their effects

We can implement an ABBA network for a robot to accomplish this task in the following way. As shown in the figure below, we implement four CMs – one for each of the actions. Additionally, we implement FDs capable of

sensing the conditions brought about by the actions. The condition `Mug-on-bench` will be a pre-condition for action `Add-coffee-&-sugar` and so on. The final condition, `Coffee-complete`, is made a goal. Either by learning or by initialisation, the effects of the actions become correlated with the actions. For example, action `Get-Mug` becomes correlated with the condition `Mug-on-bench`. This arrangement will cause activation to spread backward from the Goal and accumulate in the first action that does not have its precondition satisfied. Initially this will be action `Get-Mug`, and then actions `Add-coffee-&-sugar`, `Add-hot-water` and `Add-Milk` in turn as the conditions `Mug-on-bench`, `Coffee-&-sugar-in-mug` and `Coffee-black` are satisfied. We can consider the spreading activation as having planned the sequence for making coffee – although this is a particularly simple case, as there is only one alternative plan that will satisfy the goal.

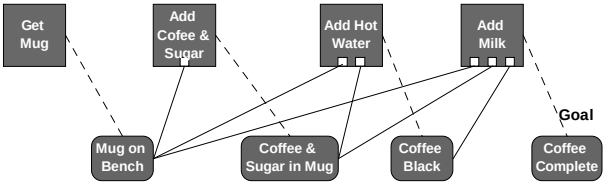

Figure 16 - ABBA Network for instant coffee

As action `Get-Mug` is being executed we can consider the state of activation in the network to represent a plan – to carry out the subsequent actions in the appropriate sequence for making coffee. It is important to notice that unlike a classical path planner, the plan is not fixed. If, for example, a person had intervened and added coffee and sugar to the mug, the robot would find condition `Coffee-&-sugar-in-mug` fulfilled and proceed directly to action `Add-hot-water`.

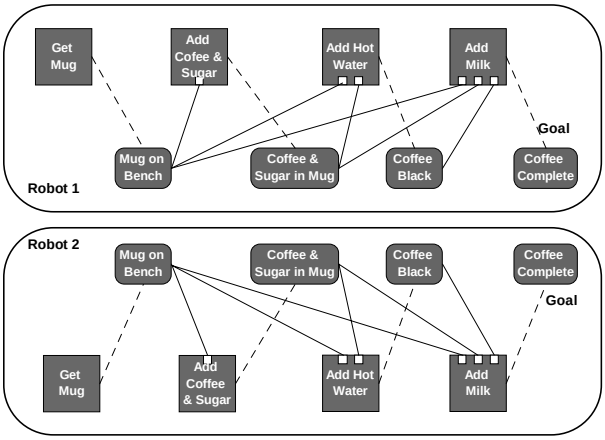

Figure 17 - Two robot instant coffee networks

Now suppose we introduce another identical robot executing the same ABBA network, as show in Figure 17. If we repeatedly ran these two robots to achieve the goal, we may find that due to the dynamics of the situation that the actions carried out by each robot may differ from run to run. For example, one run might see robot 1 get the mug and add hot water with robot 2 adding the coffee, sugar and milk. In another run robot 1 may get the mug and add hot water and milk while robot 2 only add the coffee and sugar. Although the individual behaviour of the two robots may differ from one run to the next, the global sequence of actions will always result in instant coffee. Rather than having two redundant robots, we could also achieve the same behaviour if each of the robots had specialised actuators and sensors.

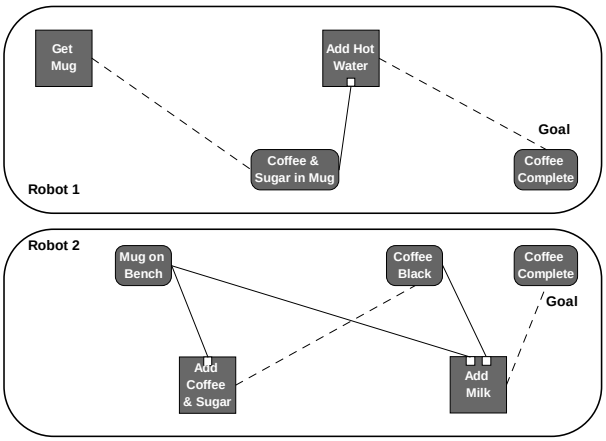

Figure 18 - Heterogeneous action sequence networks

Suppose robot 1 was only capable of getting a mug, adding hot water and sensing if the coffee & sugar are in the mug or if the coffee is made. Likewise, robot 2 can add coffee, sugar or milk and sense when the mug is on the bench, if it contains black coffee or the coffee is made. In this case, we can still achieve a system that executes the actions in the correct sequence (see Figure 18). Robot 1 will start with action `Get-mug`, hence satisfying condition `Mug-on-bench`, which will trigger robot 2 to execute action `Add-coffee-&-sugar`, and so on.

It is in this sense that ABBA networks support *joint-planning*. We have taken the simple network from Figure 16 and physically distributed it over two robots and it can still generate the only plan capable of fulfilling the goal – now a cooperative plan. This capability relies on two important *characteristics*. Firstly, the activation spreads through the networks in a way dependent on conditions that are delivered *directly* from the environment – via *Feature Detectors*. Secondly, the spreading activation also depends on the correlations that are learned from experience. It also requires feature detectors capable of perceiving the relevant conditions.

As this is such a simple example, we can easily imagine a more complex situation in which each robot has multiple alternative plans that would all accomplish the goal. In such a case, the plan chosen by one robot may depend on that chosen by another. Provided our first characteristic above is maintained – that the conditions are available *directly* from the environment, we need only equip each robot with FDs relevant to perceive the conditions on which others are basing their decisions.

Networks for complex tasks may not display this characteristic. For example, when a robot doesn't have appropriate sensors to perceive the relevant conditions directly, or when another robot has FDs whose conditions are based on internal states. In these cases, there are two possible ways around the problem. The simplest is for the other robots to communicate any hidden state information required by others – to announce their intentions. Another

solution is for a robot to represent some elements of the internal states of other robots. The spreading activation mechanism supports this well through its ability to maintain multiple hypotheses.

For example, suppose there is only one plan from two possible alternatives that will achieve a goal, but which one depends on a condition internal to a collaborator. This internal state of the other robot can be represented locally by the condition a FD (*x* in Figure 19 – circle denotes negation). Should the robot initially execute actions *A* and *B* or *A'* and *B'* when condition *x* is only uncertainly known?

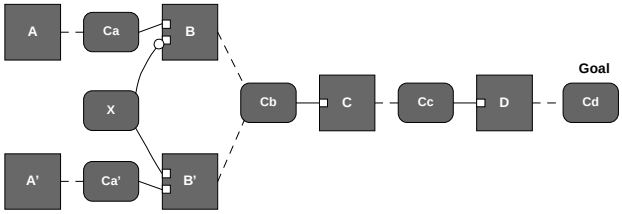

Figure 19 - Alternative plans dependant on condition x

The FD *x* will attempt to ascertain the crucial internal state of the collaborator using whatever indirect information is available from observation. ABBA FDs deliver a confidence with their conditions. If *x*'s confidence is low, activation will spread down two paths – one in accordance with a false condition (*A, B, C, D*) and the other with true (*A', B', C, D*). One path will have slightly more activation and the first action will be chosen from that alternative. The other path will continue to have high levels of activation, effectively being 'primed' in case the hypothesis is incorrect. Suppose that the hypothesis was actually incorrect – that the path chosen represented the wrong plan. If information became available to the FD *x* after execution of the first action, causing it to report the opposite condition, now with high confidence, then activation will begin to spread down the alternate path. In this case, the path already has high activation levels – having been 'primed', and will be activated rapidly. The cost of making an initially incorrect 'guess' is minimised. We can imagine classical planners that would stop and completely re-plan in cases such as this where previous information has been contradicted – causing significant delays. These examples have illustrated how an ABBA network can accomplish joint planning and flexibly adapt the plans in real-time – just as primates can.

7. COMMUNICATION

If the robots are to communicate to cooperate, the obvious question is "*what do robots talk about*?". All communication is egocentric. A robot can only communicate information from its sensory-behaviour space – sensory data, behaviour states, and indexical and iconic references. That is, any data communicated does not intrinsically carry 'meaning'. It is only meaningful in terms of its grounded context within the robot. Once it has been communicated to another robot, the context has changed, as hence so has its meaning. How can communication that doesn't necessarily preserve meaning possibly be useful? Inter-robot communication can be useful in the following three ways.

Firstly, for communication to be useful it does not have to mean the same thing to both robots. A communicated token can be re-grounded by the listener in a completely different way and still be useful information. For example, Flo communicates a specific pre-defined token when it is about to dump a pile of litter. This token is meaningless to Flo – it is involuntarily communicated since the action is pre-programmed into the *DumpLitter* behaviour, and if received by Flo it would not mean anything. To Joh, it does mean something – specifically, that if it can see Flo it will likely see a pile of litter appear in the vicinity within a short time. This meaning is grounded for Joh – only it can see piles of litter using vision.

Secondly, although Flo and Joh have no sensors in common, some robots could have identical sensors and actuators, enabling re-grounding at the iconic level to preserve meaning. For example, laughter has almost the same meaning to all humans, but not to other animals. We can make the necessary connection with the emotional state since we can hear and observe others and ourself laughing.

Finally, we can specifically design mechanisms into both the speaker and the listener such that the meaning of some communicated data, when independently grounded, is sufficiently similar to be useful for a particular task. For example, a robot may be able to create an analogous indexical association between two icons in another robot by empirical demonstration. When a human teacher shows a small child a picture of a sheep and repeats the word 'sheep' several times, this is an attempt to create an association in the child in this way. The association is built between two icons – the arbitrary utterance and the visual icon for the image (and possibly the real animal). It is this mechanism that we have utilised to enable the communication of locations between the robots.

Location Labelling

The most sophisticated layer of the cleaning task implementation enhances the efficiency of cooperation by having Flo communicate the location of litter piles it makes to Joh. As the robots each have different representations for space, their concept of location cannot be directly communicated. Although both robots use odometry, the *location FDs* also employ landmarks to deal with the accumulation of odometry error. Hence, a location for Flo may involve whisker based landmark information, while Joh may use ultrasonic information. For this reason, a location is communicated as a relative distance from one of a common set of shared known locations. These common locations are indexical associations between a label and a location that are established by a specifically designed interaction. Essentially the interaction involves Flo 'teaching' Joh the association between a label and a location. The interaction, which we'll call *labelling*, proceeds as follows.

If Joh is tracking Flo in its visual field and there are no previously labelled locations near by, then Joh communicates to Flo indicating that Flo's current location should be labelled. No label is communicated as the label for a location is simply a numerical sequence number of the labelled locations. Since both robots start counting at one, they both know which number to use as the next label. If Joh receives a confirmation signal from Flo, it associates a

proxy FD for the label with a *location FD* of Flo's location. Joh calculates Flo's location based on its own location and a calculation of Flo's range from visual tracking information. Flo also labels its location by associating the analogous label proxy FD with its current location. Figure 20 shows a partial network from Joh after two locations have been labelled.

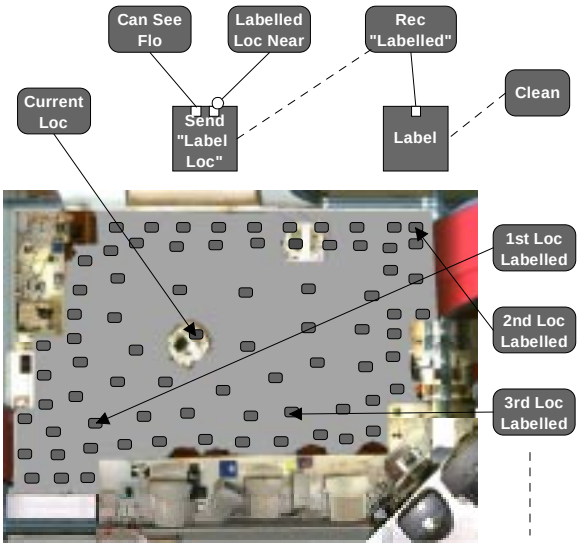

Figure 20 - Joh's Location Labelling Network

The *Rec 'pile loc'* FD becomes true when it receives the communicated location from Flo. It calculates the location FD from the map corresponding to the litter pile and assigns the proxy FD *pile location* to refer to it. Before this time the *pile location* FD doesn't refer to anything, hence activation doesn't flow past the CM *locate pile*. Once Joh has successfully navigated to the litter pile location due to the activation flowing through *locate pile* into the map, the *pile location* FD will become true and *locate pile* will be executed. The CM *locate pile* assumes that there is now a litter pile in the vicinity and rotates the robot around to bring it into view. If it comes into view, it will be detected causing the FD *can see litter* to become true and the *vacuum* CM will be executed due to activation from *clean*. The *vacuum* CM vacuums up the litter and resets the *pile location* proxy to refer to nothing.

The arrows show the referred to FD of each of the proxy FDs *Current Loc*, *1st Loc Labelled*, *2nd Loc Labelled* and so on. The CM *Label* is responsible for calculating which location FD from the SOM-based map corresponds to Flo's position and assigning the next label proxy FD to refer to it. Flo's network for labelling is similar.

To make use of these labelled locations, Flo can communicate the location of a pile of litter relative to them. For example, a pile could be specified as being approximately 3m away from the 2nd labelled location at an angle of 30^{o} relative to the direction of the 1st labelled location from the 2nd. Figure 21 shows Joh's partial network for locating a litter pile whose location was communicated by Flo.

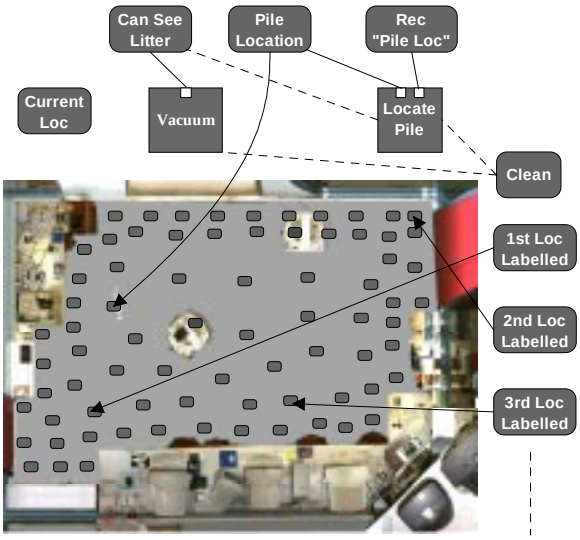

Figure 21 – Joh's Litter pile locating network

8. CONCLUSION

We have described our distributed action selection mechanism used for planning in ABBA. We have shown mechanisms utilising ABBA that implement robust and homogeneous planning of navigation, cooperation, communication and reactive behaviour. The path planning mechanism also unifies spatial and topological style map representations. A plan, as represented by a path of high activation through the network, can include any type of behaviour as elements in the action sequence.

ABBA was used to implement a real multi-robot cleaning system. Some of the component behaviours and sensing techniques were also described.

ACKNOWLEDGMENTS

This work was supported by Fujitsu who produced our vision system, and Wind River Systems, suppliers of VxWorks.

REFERENCES

[Alami et al., 1995] Alami, R., Aguilar, L., Bullata, H., Fleury, S., Herrb, M., Ingrand, F., Khatib, M. and Robert, F., "**A General Framework for Multi-Robot Cooperation and its Implementation on a Set of Three Hilare Robots**", *Proc. International Symposium on Experimental Robotics (ISER)*, 1995.

[Arkin and Hobbs, 1992b] Arkin, R. C. and Hobbs, J. D., "**Dimensions of Communication and Social Organization in Multi-Agent Robotic Systems**", *Proc. Simulation of Adaptive Behavior 92*, Honolulu, HI, Dec., 1992.

[Bond, 1996] Bond, Alan H., "**An Architectural Model of the Primate Brain**", *Dept. of Computer Science, University of California*, Los Angeles, CA 90024-1596, Jan 14, 1996.

[Cheng and Zelinsky, 1996] Cheng, Gordon and Zelinsky, Alexander, "**Real-Time Visual Behaviours for Navigating a mobile Robot**", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol 2. pp973. November 1996.

[Dudek et al., 1995] Dudek, G., Jenkin, M., Milios, E. and Wilkes D., "**Experiments in sensing and communication for robot convoy navigation**", *IEEE 0-8186-7108-4/95*, 1995.

[Heikkilä and Matsushita, 1994] Heikkilä, Tapio and Matsushita, Toshio, "**Coordination of multiple robot systems based on negotiation**", *Tech. Report 11.12.1994, Electrotechnical Laboratory* (ETL), MITI, Japan, 1994.

[American Heritage Dictionary, 1992] Houghton Mifflin Company, Ltd., "**The American Heritage® Dictionary of the English Language**", Third Edition copyright © 1992 by Houghton Mifflin Company. Electronic version licensed to Microsoft Corporation from INSO Corporation.

[Jung and Zelinsky, 1996a] Jung, David and Zelinsky, Alexander, "**Whisker-Based Mobile Robot Navigation**", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol 2. pp497. November 1996.

[Jung et al., 1998a] Jung, David, Heinzmann, Jochen and Zelinsky, Alexander, "**Range and Pose Estimation for Visual Servoing on a Mobile Robotic Target**", *Proc. IEEE International Conference on Robotics and Automation* (ICRA), 1998.

[Kohonen, 1990] Kohonen, Teuvo, "**The self-organising map**", *Proceedings of IEEE*, 78(9):1464-1479, Sep. 1990.

[Kube and Zhang, 1994] Kube, C. Ronald, Zhang, Hong, "**Collective Robotics: From Social Insects to Robots**", *Adaptive Behaviour*, Vol. 2, No. 2, 189-218. 1994.

[Le Pape, 1990] Le Pape, Claude, "**A Combination of Centralized and Distributed Methods for Multi-Agent Planning and Scheduling**", *Robotics Laboratory, Stanford University*, Stanford CA 94305, 1990.

[Maes, 1990a] Maes, P., "**Situated Agents Can Have Goals.**", *Designing Autonomous Agents*. Ed: P. Maes. MIT-Bradford Press, 1991. ISBN 0-262-63135-0. Also published as a special issue of the Journal for Robotics and Autonomous Systems,Vol. 6, No 1, North-Holland, June 1990.

[Matarić, 1992] Matarić, Maja J., "**Integration of Representation Into Goal-Driven Behavior-Based Robots**", in *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, June 1992, 304-312.

[Matarić, 1995] Matarić, Maja J., "**Issues and Approaches in the Design of Collective Autonomous Agents**", to appear in "*Moving the Frontiers Between Robotics and Biology*", special issue of Robotics and Autonomous Systems, Vol. 16, Nos. 2-4, December 1995.

[Noreilis, 1992] Noreilis, Fabrice R., "**An Architecture for Cooperative and Autonomous mobile Robots**", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France - May 1992.

[Parker, 1998] Parker, Lynne E., "**ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation**", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 2, April 1998.

[Rhodes, 1995] Rhodes, Bradley, "**Pronomes in Behaviour Nets**", *Tech. Report #95-01*, MIT Media Lab, Learning and Common Sense Section. Jan 1995.

[Riekki and Kuniyoshi, 1996] Riekki, J. and Kuniyoshi, Y., "**Behavior Cooperation Based on Markers and Weighted Signals**", *Proceedings of the World Automation Congress (WAC) - International Symposium on Robotics and Manufacturing* (ISRAM), Montipellier, France, March, 1996.

[Yuta et al., 1991] Yuta, S., Suzuki, S. and Iida, S., "**Implementation of a small size experimental self-contained autonomous robot - sensors, vehicle control, and description of sensor based behavior**", *Proc. Experimental Robotics*, Tolouse, France, LAAS/CNRS (1991).

[Zelinsky et al., 1993] Zelinsky, A., Kuniyoshi, Y. and Tsukue, H., "**A Qualitative Approach to Achieving Robust Performance by a Mobile Agent**", *Robotics Society of Japan Conference*, Japan, November 1993.