Enabling Multiple Execution Agents

Ruth Aylett, Alex Coddington, IT Institute, University of Salford
David Barnes, Rob Ghanea-Hercock, EEE Dept. University of Salford

1. Introduction

This extended abstract describes work at the University of Salford Mobile Robotics Lab with a hybrid architecture which combines a symbolic predictive planner - embodied in a Reflective Agent with no sensing capabilities - with a number of non-symbolic reactive (in fact Behavioural) agents - embodied in multiple cooperating mobile robots. This work is funded by the UK Engineering and Physical Science Research Council under the title MACTA - Multiple Automata for Complex Task Achievement. As described below, the behavioural agents (BAs) are constructed using the Behavioural Synthesis Architecture (BSA) created in Salford (Barnes&Gray 91, Barnes 96), while the Reflective Agent uses a standard non-linear planning system, currently ZENO from the University of Washington (Penberthy&Weld 94). These components are described below in section 3.

The motivation for MACTA was to try to combine the strengths of predictive planning - efficiency, coherence and high-level user interface - with the strengths of a behavioural system - resilience and responsiveness. From a planning perspective this involves integrating planning and execution, and the project team has drawn a number of lessons about how some of the problems inherent in this relationship may be tackled which are discussed in section 5.

A number of the domain characteristics outlined in the symposium call are relevant to this work. Real-world (that is, not simulated) robotic domains are inherently difficult for well-known reasons. Firstly, complete information is out of the question, either for the planning or executing systems. Accurate knowledge of the current state of the domain can only be gathered via sensors and these are practically and theoretically limited in scope. Only the part of the domain within range can be perceived and even here sensor data is noisy and difficult to interpret. The construction of higher level models from raw data is fraught with problems so that the prerequisite for correct symbolic planning - the existence of an accurate and complete symbolic model - is usually far from being met.

This might not be so bad if the domains were completely static, for then it might be feasible to build an accurate model off-line with static, calibrated sensors. However no useful mobile robot domain is that static in practice - changes outside the control of the robot itself are inevitable. For this reason reactive capabilities are nearly always required to allow the robot to cope with the world as it is not as it was predicted to be. Mobile robot domains thus exhibit both complexity and uncertainty.

In the MACTA project, domains also include a further complication: multiple cooperating robots. This means that concurrency and coordination are also important issues.

A further issue - not discussed in the call - is that robot execution actions are normally rather low-level compared to the overall task. This can be seen even in a toy domain such as the blocks world. Where many planners discuss this problem in terms of 'move block' with no indication of what execution agent will actually do this, a real manipulator is more likely to require actions such as 'move arm' 'open gripper' and 'close gripper' (Aylett 92).

Moreover many of the goals being met within the plan may be irrelevant at execution time if the execution agent lacks the sensors necessary to detect them: that a block is 'clear' for example, or even that a block is successfully gripped in some cases. One might even argue that goals and actions have opposite importance in planning and in execution. In the former, the goals are the important thing: actions are a way of packaging pre- and post-conditions together but themselves have almost no independent content. In the latter as we have just argued, some goals may be of no practical significance at all while others may just be a way of enabling the all important actions which actually change the state of the real world. We return to this issue below.

Finally, on the positive side, while robot actions tend to be low-level, the designer often has some control over exactly what they are. The decision on what to leave to planning and what to incorporate in the execution agent may be an open one, with the possibility of solving some planning problems by incorporating extra execution

facilities. The philosophy of the MACTA project has been to put as much intelligence as possible into the execution system and this has helped to reduce planning problems as discussed below in section 5.

2. Relationships between Planning and Execution

Insofar as classical planning considered execution at all, it assumed a direct determining relationship between planner actions (or planner primitive actions in the case of hierarchical planning) and executed actions. It should be clear that for robots, the low-level planning actions above may each require a number of even lower level actuations well below the level one would wish to manipulate symbolically (because of real-time constraints and the need for numerical representations).The 'open gripper' action just mentioned results in many more actions at the level of motors and gears in a manipulator. Nevertheless in the classical approach there is still a direct causal one-to-many relationship between the planner action and the eventual actuations.

However this is not the only possible relationship between planned and executed actions, as much work in the last ten years or so has shown, for example (Agre & Chapman 87, Gat 92, Lyons & Hendrik 92). One alternative is to see planner actions as *enabling* execution actuations(Gat 92): a planner action constrains actuation but the latter is directly determined by sensor input so that there is no longer a direct causal relationship between planner primitives and executed actuations. Such a relationship makes it possible to use the behavioural approach - with its requirement for a tight connection between sensor inputs and actuator outputs - as an execution mechanism while still giving symbolic planning a powerful role. It is the basis for the MACTA project and exactly how it has been applied will be discussed in section 3.

An even more execution-oriented approach is to see planning as a resource so that execution system can use a planner as-and-when this seems necessary. Again, a number of workers have taken this approach (Agre & Chapman 87, Lyons & Hendrik 92). The MACTA project sees this as a means of dealing with execution failures that cannot be handled by a behavioural system (knowing which these are is still a research issue). This flows from the decision to centralise planning but distribute execution as we will see in section 3.

The relationships between planning and execution described so far are essentially static but of course there is also a temporal dimension: plan-then-execute, or interleave; and if the latter then with what granularity. In robotics this dynamic relationship is constrained in two ways. In the first place, giving a high degree of autonomy to the robot means that if the planner is not installed on it then the ability of planning and executing systems to communicate may be limited. It was this factor which underlay the MACTA decision to minimise communication between the two. On the other hand, one may install symbolic planning onto the robot itself, but in this case there may be severe resource implications and the interaction of two systems working on such different time-scales, usually with different representations, must be very carefully managed (Aylett et al 91). The existence of multiple cooperating robots may also result in a high inter-robot communications overhead since problems of model inconsistency at the planner level then become significant.

3. The MACTA approach

In this extended abstract, only an overview of the overall MACTA system is possible. The reader is referred elsewhere (Barnes 96, Aylett 96) for a more detailed account.

The original motivation for multi-robot work at Salford was the desire to explore the idea that many small simple robots might be more successful at carrying out complex tasks than the traditional single more complicated one had turned out to be. A novel behavioural architecture (different from subsumption) was developed for this purpose.

3.1 The Behavioural Synthesis Architecture (BSA)

The basic unit of the BSA is the behaviour pattern. Each pattern is composed of two functions: a stimulus-response mapping which for any sensory stimulus determines the motion response (either in terms of rotation or translation), and a stimulus-utility mapping, which for any sensory stimulus determines the importance of the motion response. Thus a collision avoidance behaviour pattern has a stimulus-response function

which reduces translation as the distance sensor returns a stronger response (indicating an obstacle is close) together with a stimulus-utility function which gives a higher importance to this response as the obstacle gets closer. Note that both of these functions are non-symbolic in nature.

As is usual in a behavioural architecture, many behaviour patterns are concurrently active and thus conflict resolution is required. Unlike the subsumption architecture in which patterns are time-sliced, with only one pattern controlling the actuators at a given moment, the BSA synthesises the responses of all active patterns as weighted by their current utility. Thus the emergent translation and rotation are a combination of all the weighted outcomes of the active patterns.

The BSA as so far described suffers from the same problem as any other behavioural architecture (and some other reactive architectures too): if all possible patterns are always active, then all types of unwanted interactions may occur. An obvious example is the conflict between patterns producing collision avoidance and patterns producing docking for a mobile robot, or worse still, grasping for a manipulator. The patterns that should be active at a particular time are those required to carry out the current task, in other words the behavioural system needs to be able to use the subtask structure in order to avoid destructive conflicts. At Salford, this subtask structure was embodied in a construct known as a behaviour script.

3.2 Behaviour Scripts

A Behaviour Script is made up of a sequence of behaviour packets. Each behaviour packet names a number of behaviour patterns which are to be active, and includes a sensory precondition for their activation and a sensory post-condition for their deactivation. Thus a behaviour script can be thought of as a sequence of triplets: (<sensory pre-condition(s)><behaviour patterns><sensory post-condition(s)>).

Each behaviour packet within the script may be thought of as accomplishing a part of the overall task: thus a navigate-to-beacon packet contains a sensory pre-condition that the beacon is sensed, the behaviour patterns required to carry out the subtask (translate to beacon, rotate to beacon, and obstacle avoidance patterns), and the sensory post-condition that the robot is within the target distance of the beacon. If the robot is to dock at the beacon, the next packet would control this process, with its list of active behaviour patterns leaving out collision avoidance.

A behaviour script therefore uses the sub-task structure to create a series of contexts in which only relevant behaviour patterns are active.

3.3 The Reflective Agent

Initially, behaviour scripts were hand-crafted. However it was clear that the translation of an overall mission into a subtask structure was exactly what a planner carried out, and the MACTA project began with the intention of incorporating the BSA into a larger multi-agent architecture.

The work has always had an industrial focus, so that the typical organisational context envisaged was one in which a human operator was responsible for a team of behavioural agents. This was the first reason for segregating planning into a single reflective agent running on a work station which could interact in a straightforward way with such an operator.

The second reason for this approach lay in a policy decision about the relationship between planning and execution to be adopted. As already argued above, an enabling relationship allows a behavioural component to deal with all sensor output leaving the planner to create the context for its reactions. Where then does the model used in planning come from? The answer adopted was that the planning model would be confined to an abstract a priori level and to those aspects of the world which were relatively unchanging. The dynamic aspects of the world would be confronted at execution time only.

This separation would not work in every planning domain. It depends on the sub-task structure - at the level of primitive planning actions - referencing only these relatively stable aspects of the domain. However this seems very plausible in the types of semi-structured industrial domains targeted by the project: within a factory for example, the basic internal layout is well-known to those working there precisely because this simplifies the execution of their plans. This separation of the world into abstract planning aspects and real-time execution aspects simplifies planning as already shown by work with manip-

ulators at Edinburgh (Malcolm&Smithers 90).

A further advantage of this separation is that it removes the need to derive symbolic models directly from sensor data, an expensive and error-prone process as already noted. The subtask structure, in the form of a behaviour script, can be communicated to every robot involved in a cooperative task and execution can then proceed autonomously with a final success message allowing the planner to update its model with the logical consequences of the planned actions. A behaviour script can be encoded into a very small data-structure since it only references patterns which are actually held within the robots, forming their overall behavioural repertoire. Thus only a low bandwidth connection is needed.

Interleaving of planning and execution would only be required when the robots failed to execute a behaviour script and therefore only at a very coarse level of granularity. Detecting failure in a behavioural context is an interesting problem but the detail is not relevant to this paper. Suffice it to say that just as a design decision must be made initially about the division of responsibility between behavioural execution and predictive planning, so must analysis of possible failures divide those which can be handled behaviourally from those which require planner intervention.

An example of the first case would be a robot getting stuck in a dead-end while navigating to a beacon. There is every possibility of dealing with this problem locally and work is currently being carried out in this area. An example of the second case would be a robot breaking down on the way to a rendezvous with a second robot with which it is due to perform a cooperative task. The second robot has no behavioural way of recruiting a new partner even were it desirable to allow this, and communication of failure to the planner is needed with replanning of the structure of the task. The behaviour script gives the planner information about where failure has occurred so that it can update its model correspondingly.

3.4. Converting plans to behaviour scripts

It was decided to use an off-the-shelf planner to produce a plan-net from an initial user goal rather than write a new one from scratch. This was partly to test the hypothesis that the planner could be sufficiently de coupled from the execution system so that different ones could be slotted in. Work is currently being carried out with the ZENO planner from the University of Washington, but the Prodigy planner from Stanford is also being evaluated.

The plan net created during planning is passed to a separate component in the Reflective Agent, the Mission Organiser, currently under construction, which converts the plan net into the correct number of behaviour scripts. Primitive planner actions have been designed to match behaviour packets one-to-one as far as possible. It would be entirely possible for a planning action to convert into several behaviour packets, but in this case a failure in one of the packets would be much harder to deal with in the replanning process.

For example, within the planner, `navigate(robot,near,place1,near,place2,time1,time2)` indicates that the robot should end up <u>near</u> the beacon at `place2` and translates into a `navigate-to-beacon` packet. However the same action with one different parameter: `navigate(robot,near,place1,at,place2,time1,time2)` which leaves the robot <u>at</u> `place2` would imply docking at the beacon and could be translated into `navigate-to-beacon` and `dock-with-beacon` packets. It would remove the necessity for the planner to have a specific docking action itself, but if the robot got to the beacon and then failed to dock successfully, the planner would have to assume that the whole navigate action had failed making it hard to recover.

4. Experiments

Work has been carried out in two main robot scenarios using the project's two real (B12) robots, Fred and Ginger. A simulator allows investigation of problems containing more robots, though as always simulator results must be treated with extreme caution.

4.1. Object relocation

In this scenario, Fred and Ginger cooperatively relocated a flat perspex object (a pallet) which coupled them closely together - much as a table would the two people who were carrying it - while avoiding obstacles. This demonstrated that cooperation need not depend on high-level symbolic reasoning but could be produced behav-

iourally without any explicit models. The robots had no model of the environment or of each other but their active behaviour patterns allowed them to stay together as they avoided obstacles via feedback from the push or pull of the pallet.

The subtask structure embodied in the behaviour script required each robot to navigate to and dock with a loading station, have the pallet loaded and then carry it to a destination. Coordination and concurrency were handled at the behavioural level as was the avoidance of obstacles.

4.2. Object interchange and tracking.

A more complex scenario based on problems in nuclear decommissioning is currently the project focus. Here one robot collects an object, and makes a rendezvous with the second robot, passing it the object. It then tracks the progress of the first robot to a destination. While each robot was given an identical behaviour script in the first scenario, here the subtask structure is different for each.

There are also new challenges at the behavioural level. Where collaborative relocation was a close-coupled task, tracking is loose-coupled since the robots are not physically connected. Moreover the amount of coordination required to carry out object interchange is much higher than that needed for relocation. As is well-known in the community, the design of behaviour patterns is very much trial-and-error and this is worsened where hardware modification is also a possibility.

So far a subset of the overall task has been demonstrated, downloading hand-crafted behaviour scripts across a radio link to the robots. A set of planner actions has been developed and these are being implemented in ZENO. A number of interesting planning problems have come to light in the process which are discussed in the next section.

5. Lessons learned

5.1. Dividing responsibility

We would not argue that the approach taken in the MACTA project is universally applicable, even in mobile robot domains, never mind in ones where humans are the execution agents. Indeed the first lesson we would draw is that for planning problems, careful characterisation of the execution agent(s) is as important as modelling of the domain. This determines the level of primitive planning actions for one thing, and thus has an immediate effect on design of actions and on domain representation. This is likely to be true even where humans are execution agents: work in a quite different domain by one of the authors uncovered the fact that plant operating procedures in process plant are designed down to a level of detail determined by assumptions about the overall skills of the plant operators.

Elsewhere (Aylett&Jones 96) we have characterised planning domains as a triangular relationship between task, agent and domain. We believe it may be possible to use this idea as a starting point for characterising the planning facilities required for a particular combination of these three.

Our conclusions about the nature of planning for our problem domain are that a plan should lie at as high a level of abstraction as possible, where this is defined by the abilities of the execution agents. Uncertainty on the other hand should be dealt with at as low a level as possible, wherever possible at execution time.

A number of examples demonstrated to us that planning problems can sometimes be designed out by incorporating more intelligence at execution time. For example, at one point we considered the problem of a non-deterministic action, that is one where the outcome cannot be determined until it is carried out (requiring conditional planning). In our case it arose because two robots were collaboratively transporting a metal object using magnets to grasp it. In order to allow one robot to acquire the object, both would pull apart - but sometimes one would end up with the object and sometimes the other. In this case more sophisticated carrying mechanisms removed the problem altogether. A specific fix like this is of course not a general solution to the problems of non-determinism but perhaps the nature of the problems to be solved in a particular domain should be taken more into account than it sometimes is in the planning community.

5.2 Accidents

Recognising plan failure and taking advantage of favourable accidents are quite different issues where planning enables execution level reactions from ones in which it determines them.

Recognising plan failure in a behavioural system underlines the need for knowledge of the subgoal structure: without knowing what you are trying to achieve, failure is undefined. Favourable accidents are difficult to take advantage of in the MACTA system except at the very lowest level: a packet will stop when its post-condition is met, irrespective of whether this takes some time or occurs as soon as the packet becomes active. However it is very hard to see how one could skip packets because those leading up to a later one are no longer necessary. The cost of communicating the subgoal structure to a system which doesn't understand it is that it cannot change it in any sensible way.

5.3. Multiple execution agents

A major lesson emerging from the work in applying existing planners to a multi-robot domain is the lack of any special role in most planning systems for the execution agent(s). Most systems seem to have one implicit execution agent which from the level of the primitive planner actions is a human (one could say this at least of many blocks worlds examples, also of the scenarios discussed in papers on the University of Washington planners).

We perceive a parallel between the MACTA multi-robot domain and systems such as SIPE or OPLAN which have been applied to large logistics problems. In both cases, it is impossible to treat execution agents as resources which can be scheduled when planning is complete since making the execution agents available for later actions often requires planning. In both cases, actions will actually be executed by a number of different agents. However where MACTA treats linearization as a process of generating the individual plans for each agent, OPLAN, for example, seems currently to treat an attached simulator as the single execution agent of the whole multi-agent plan. Searches of the multi-agent literature have not found any treatment of planning for multiple agents so far either: systems here seem generally to incorporate planning into each individual agent, producing distributed planning in which the execution agent is still implicit and in fact identical with the planning agent.

An example of the difficulties which are being examined comes in the planning of cooperative actions. It is quite possible that in a collection of robots, some might have different weight-bearing capabilities. Thus an object might be transportable by one 10 ton-carrying robot if one is available, and by two 5-ton carrying robots if it is not. It seems rather clumsy to define different carrying actions depending on how many robots are involved, but of course the presence of the appropriate number is a logical precondition and requires further planning to get them there.

5.4. Planner goals and execution conditions

We have commented above that not all planner goals are represented at the behaviour script level in MACTA as packet pre- or post-conditions. In fact two very different roles are played by these constructs. Planner pre-conditions represent the logically necessary conditions for the correct behaviour to emerge when the corresponding behaviour packet is active; for example a cooperative transport action requires each robot to be holding the transported object. Planning is needed to make sure that these conditions will be met.

In contrast, at the execution level, transition conditions are important - the sensory inputs which deactivate one behaviour packet and activate the next are derived from the context of the task. In the MACTA scenario, the necessary condition for a robot to release an object is that the robot is holding it, but the transition condition is that the appropriate beacon is sensed.

6. Conclusion

We argue that for mobile robot applications of the type we are targeting, an enabling relationship between planning and execution may allow the combination of reactivity and foresight so as to apply the strengths of each. We would argue even more strongly that a characterisation of the skills of the proposed execution agent is vital to the design of planning for a particular domain. It may follow that execution agents should be explicitly modelled in the planner, and this is an issue that we intend to explore further.

7. References

Agre, P and Chapman, D. (1987). Pengi: An implementation of a theory of activity. Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI 87 Morgan Kaufmann

Aylett, R.S. Fish, A.N. & Bartrum, S.R. (1991) HELP - a Hierarchical Execution-Led Planner for Robotic Do-

mains. Proceedings, 1st European Workshop on Planning, EWSP '91, Springer-Verlag, 1991. pp1-12

Aylett, R.S. (1992) Linking a task planner to other agents in a generic robot architecture. Proceedings, 11th UK Planning SIG, University of Sussex, April 1992

Aylett, R.S. (1996) Communicating goals to behavioural agents - A hybrid approach. Proceedings, 3rd World Congress on Expert Systems, vol2, pp782-789, Seoul, Korea, Cognizant Corporation Ltd, 1996

Aylett, R.S. & Jones, S.D. (1996) Planner and Domain: Domain Configuration for a Task Planner. International Journal on Expert Systems Theory and Applications, Special Issue on KB Planning, forthcoming.

Barnes, D.P. & Gray, J.O. (1991)Behaviour Synthesis for Co-operant Robot Control Proceedings, IEE International Conference, Control '91, pp1153-1140

Barnes, D.P. (1996) A behaviour synthesis architecture for cooperant mobile robots. Advanced Robotics and Intelligetn Machines, eds J.O.Gray & D.G.Caldwell, IEE Control Engineering Series 51.295-314 1996

Gat, E. (1992) Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. Proceedings, 10th National Conference on Artificial Intelligence, AAAI-92, pp809-815

Lyons, D.M. & Hendriks, A.J. (1992) A Practical Approach to Integrating Reaction and deliberation. Artificial Intelligence Planning Systems: Proceedings of the 1st International Conference. Ed. James Hendler. pp 153-162. Morgan Kaufmann 1992.

Malcolm, C. & Smithers, T. (1990) Symbol grounding via a hybrid architecture in an autonomous assembly system. Robotics and Autonomous Systems, 6, pp123-44, 1990

Penberthy, J.S. & Weld, D,S. (1994) Temporal Planning with Continuous Change. 12th National Conference on AI, AAAI-94, 1994