The Autogeneration of Reactive Plans

D.M. Lyons and A.J. Hendriks
Philips Laboratories
North American Philips Corporation
Briarcliff Manor, NY 10510

Invited paper. 13th IMACS World Congress on Computation and Applied Mathematics, July 22-26, 1991, Trinity College, Dublin, Ireland.

1 Introduction

A *kitting robot* is a robot system that puts together assembly kits. Simpler and cheaper automation can construct the assemblies once they have been placed in the kits and routed appropriately. In this way, the intelligence and cost can be focused into a small number of kitting robots which feed the rest of the factory. To deal with events as they happen, the kitting robot needs to be capable of real-time, reactive response. To deal with changes in the environment (e.g., bursts of errors in parts, or failures of resources) or new goals issued by factory management, the kitting robot needs to be able to plan ahead 'on the fly'. The current literature in Robotics and AI does not yet support building a robot control system that successfully incorporates both of these attributes. Classical AI planning has difficulty operating robustly in uncertain and dynamic environments. Reactive approaches operate very robustly in some environments — those for which the reactions have been pre-programmed. This paper outlines an approach in which planning and reaction are integrated by casting planning as the continual, goal-oriented adaptation of a reactive system, the reactor.

2 The $\mathcal{RS}$ Model

We use a process-based, formal model of computation, the $\mathcal{RS}$ model [2], to describe and analyze the reactor. This model was developed to express robot plans that could 'tune' their behavior to suit their environment. A very brief introduction to this model is given here; see [3] for more details.

An $\mathcal{RS}$ description of a process, or network of processes, is called a *schema*. For example, $\texttt{Joint}_{j=v}$ denotes a process that is an instance of the schema $\texttt{Joint}$ with parameter j set to value v (when unambiguous, we write $\texttt{Joint}_v$). Processes can be coupled into networks in two ways: they can communicate messages to one another via *communication ports*, or they can be composed together using several kinds of *process composition operators*. Processes can be defined in terms of networks of other processes, grounding out with a set of atomic, pre-defined, processes. Process composition operators include: Network composition $(\texttt{A},\texttt{B})^{\texttt{c}}$, Disabling composition $(\texttt{A}\#\texttt{B})$, Enabling composition $\texttt{A}\langle x\rangle\texttt{:B}_x$, Sequential composition $\texttt{A;B}$, and two recurrent compositions, Recurrent Synchronous $\texttt{A}\langle x\rangle \texttt{:;B}_x$, and Recurrent Asynchronous $\texttt{A}\langle x\rangle \texttt{::B}_x$. To represent how process networks evolve over time as processes dynamically terminate or are created, we developed the evolves operator: process $\texttt{A}$ evolves into process $\texttt{B}$ under condition Ω if $\texttt{A}$ becomes equal to $\texttt{B}$ when condition Ω occurs; written $\texttt{A} \xrightarrow{\Omega} \texttt{B}$.

All actions are represented as processes, and can be sent messages and controlled like any other processes. This is very different from the classical planning view of actions as atomic events that simply 'happen' and are only characterized by preconditions and effects. Our approach allows us to easily handle effects that have proven important in building reactive systems: e.g., the 'inhibition' of actions, or the modification of the behavior of actions by other actions.

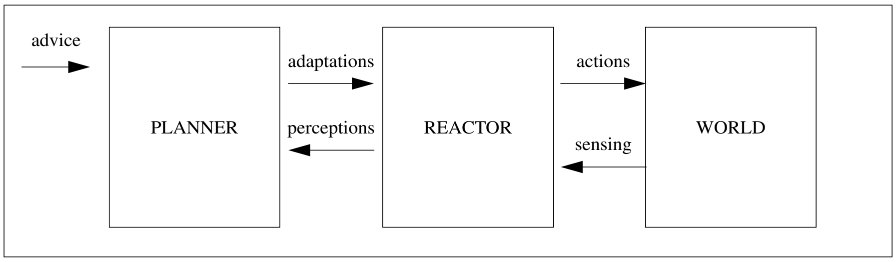

Figure 1: Planner-Reactor-World System

It is not possible to describe the behavior of a reactive system in any concise way without referring to the environment in which the system is active. Therefore, the appropriate unit to analyze is the system embedded or situated in its environment. Our approach in $\mathcal{RS}$ is to build world models from the same tools used to build the reactor: networks of concurrent processes.

3 Planner-Reactor System

The reactor is a system, like those discussed by Brooks [1] and others, that contains a network of sensory-motor 'linkages' which we will call reactions. The reactor has several key properties. First, it can produce action at *any* time. Unlike a classical plan executor, a reactor is always actively inspecting the world, and will act should one of its reactions be triggered. A reactor should produce useful behavior even without a planner. Secondly, it is a *real-time* system. The central objective of the reactor is to produce the appropriate action at the appropriate time. To make this possible, all the 'intelligence' in a reactor is encoded into reactions and cross-connections between them, and annotated with time-constraints (see our companion paper in these proceeedings for more details).

The planner is a system separate and concurrent with the reactor. It cycles, continually determining from a description of the current configuration of the reactor, if reactor behavior in the current environment will indeed conform to the set of desired objectives. If it determines that the reactor's behavior will not conform, it makes an incremental change to the reactor configuration to bring the reactor (more) into line. As indicated in Figure 1, there are two connections between planner and reactor. *Perceptions* are data gathered by the reactor, and fed into the planner. By altering the configuration of the reactor, the planner alters what perceptions it can receive. *Adaptations* are alterations of the reactor structure generated by the planner.

4 Reactor Adaptation

The general form of the reactor is the following concurrent network of processes:

$$\mathtt{REACTOR} = (\ \mathtt{E}^1\#\mathtt{P}^1, \ldots, \mathtt{E}^n\#\mathtt{P}^n,\ \mathtt{ADDPLAN}\langle k\rangle :: (\mathtt{E}^k\#\mathtt{P}^k)\ ) \tag{1}$$

where

1. $\mathtt{P}^i$ is a sensory-action pair, a reaction or part of a plan.

2. $\mathtt{E}^i$, by the definition of $\#$, is a guard process by which $\mathtt{P}^i$ can be removed by the planner, $\mathtt{E}^\mathtt{i} \longrightarrow \mathtt{STOP} \Rightarrow \mathtt{P}^\mathtt{i} \longrightarrow \mathtt{ABORT}$.

3. ADDPLAN, by the definition of ::, is the interface through which the planner can add new guarded reactions $\mathtt{E}^m\#\mathtt{P}^m$,

$$\begin{array}{l}\mathtt{ADDPLAN}\langle k\rangle \longrightarrow \mathtt{STOP} \quad \textit{and } k=m \textit{ at termination}, \Rightarrow \\ \mathtt{ADDPLAN}\langle k\rangle :: (\mathtt{E}^k\#\mathtt{P}^k) \longrightarrow (\mathtt{ADDPLAN}\langle k\rangle :: (\mathtt{E}^k\#\mathtt{P}^k), (\mathtt{E}^m\#\mathtt{P}^m)\ )\end{array} \tag{2}$$

Items 2 and 3 above indicate the only ways in which the reactor can be changed by the planner. To facilitate this, plan 'segments' are represented in the reactor in the following, *situation-labeled* form:

$$\mathtt{Pt} = \Big|_{s\in sit(t)} (\ \mathtt{CheckSit}_s : \mathtt{Action}_{t,s}\ \#\ I_{t,s}\ ) \tag{3}$$

This is a concurrent set of actions, each of which carries its own monitor process CheckSit to determine if the world and reactor present an appropriate situation with respect to task t in which the action (sequence) should be carried out. A situation is a task and world-model specific invariant on the world. $sit(t)$ is the set of situations for task t for a particular world model. Every action is 'guarded' by an invariant I that monitors the executability conditions for the action; should the invariant monitor fail, that plan segment is disabled, and the planner is notified.

We want to specify that adaptation of the reactor by the planner must *improve* the behavior of the reactor over time. This notion of improvement can be expressed as constraints on the co-evolution of the reactor and world process as they achieve the goal. Let $\mathtt{R}^*_{\mathcal{E},\mathcal{G}}$ be the reactor that perfectly achieves the objectives $\mathcal{G}$ in environment $\mathcal{E}$, thus

$$(\ \mathtt{R}^*_{\mathcal{E},\mathcal{G}}\ ,\ \mathtt{W}\ ) \xrightarrow{\omega} \mathtt{G} \tag{4}$$

That is, $\mathtt{R}^*_{\mathcal{E},\mathcal{G}}$ achieves G under constraints ω in world W, where G and ω constitute the objectives $\mathcal{G}$, and W is the process model of the environment $\mathcal{E}$. Any specific instance of a reactor $\mathtt{R}_{\mathcal{E},\mathcal{G}}$ will be an approximation to the perfect reactor. We characterize improvement as follows. If $\mathcal{E}$ (or $\mathcal{G}$) changes to $\mathcal{E}'$ ($\mathcal{G}'$ respectively) in (4) above, then, over time, $\mathtt{R}^*_{\mathcal{E},\mathcal{G}}$ should evolve to $\mathtt{R}^*_{\mathcal{E}',\mathcal{G}}$ ($\mathtt{R}^*_{\mathcal{E},\mathcal{G}'}$ respectively), and if $\mathcal{E}$ and $\mathcal{G}$ remain unchanged, then, over time, $\mathtt{R}_{\mathcal{E},\mathcal{G}}$ should evolve towards $\mathtt{R}^*_{\mathcal{E},\mathcal{G}}$. Ignoring the E and ADDPLAN components of the reactor structure, we can say that any reactor $\mathtt{R}_{\mathcal{E},\mathcal{G}}$ can be written in terms of the perfect reactor as follows

$$\begin{array}{l}\mathtt{R}^* = (\mathtt{R}\ ,\ \mathtt{R}') \\ \textit{where } sit(\mathtt{R}^*) = sit(\mathtt{R}) \cup sit(\mathtt{R}'), \quad \textit{and} \\ s \in sit(\mathtt{R}) \cap sit(\mathtt{R}') \Rightarrow \mathtt{Action}'_s \textit{ extends } \mathtt{Action}_s\end{array} \tag{5}$$

One process network extends a second if the second is a prefix of the first. So, given two reactors R1 and R2 we say R2 *improves* on R1 if when they are both written in the form (5), then

1. $sit(\mathtt{R2}') \subset sit(\mathtt{R1}')$, and
2. for each $s \in sit(\mathtt{R1}) \cap sit(\mathtt{R2})$ $\mathtt{Action2}_s$ *extends* $\mathtt{Action1}_s$.

That is, R2 is a better approximation to $\mathtt{R}^*$ than R1 .

5 Conclusion

The concept of a reactor that can be improved by a planner has been formalized in the $\mathcal{RS}$ model. Improvement is not too dissimilar from the classical concept of plan construction, so building such planners may indeed be possible. Note that the robustness of the planner may in turn be limited by its vocabulary of adaptions and perceptions.

References

[1] R. Brooks. A robust layered control system for a mobile robot. *IEEE J. Rob. & Aut.*, RA-2(1):14–22, Mar. 1986.

[2] D.M. Lyons and M.A. Arbib. A Formal Model of Computation for Sensory-Based Robotics. *IEEE Trans. Rob. & Aut.*, 5(3):280–293, June 1989.

[3] D.M. Lyons, A.J. Hendriks, and S. Mehta. Achieving robustness by casting planning as adaptation of a reactive system. In *IEEE Int. Conf. Rob. & Aut.*, Apr. 7–12th 1991.