Robotics and Autonomous Systems 11 (1993) 279–291
Elsevier

A symbolic solution to intelligent real-time control

Douglas J. Pearson *, Scott B. Huffman, Mark. B. Willis, John E. Laird, Randolph M. Jones

The University of Michigan, Artificial Intelligence Laboratory, 1101 Beal Ave., Ann Arbor, MI 48109-2122, USA

Abstract

Pearson, D.J., Huffman, S.B., Willis, M.B., Laird, J.E. and Jones, R.M., A symbolic solution to intelligent real-time control, Robotics and Autonomous Systems 11 (1993) 279–291.

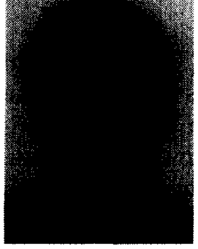
Autonomous systems must operate in dynamic, unpredictable environments in real time. The task of flying a plane is an example of an environment in which the agent must respond quickly to unexpected events while pursuing goals at different levels of complexity and granularity. We present a system, *Air-Soar*, that achieves intelligent control through fully symbolic reasoning in a hierarchy of simultaneously active problem spaces. *Achievement goals*, changing to a new state, and *homeostatic goals*, continuously maintaining a constraint, are smoothly integrated within the system. The hierarchical approach and support for multiple, simultaneous goals gives rise to *multi-level reactive* behavior, in which Air-Soar responds to unexpected events at the same granularity where they are first sensed.

Keywords: Reactive control; Real time; Symbolic; Soar; Architecture; Flight

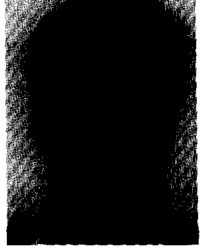
1. Introduction

Autonomous systems must function well in dynamic, unpredictable environments in real time. This paper describes a system for intelligent control of an airplane, within a realistic flight simulator. The simulator used is the Silicon Graphics Flight Simulator, modeling a light aircraft similar to a Cessna. To fly the plane, our system must perform a range of tasks at different levels of complexity and granularity while responding to unpredictable events in the environment.

To provide intelligent control in this domain, we have constructed *Air-Soar*, built within the general problem solving and learning architecture of Soar [9]. Air-Soar reasons simultaneously in a hierarchy of problem spaces at different levels of control granularity. At the highest level, it rea-



Douglas J. Pearson is a doctoral student in the Artificial Intelligence Laboratory at the University of Michigan. He received a B.S. in Computer Science and Mathematics at the University of St. Andrews, Scotland, in 1988 and an M.S. in Computer Science from the University of Michigan in 1992. His primary research interests include the integration of learning strategies and applications of learning to problem solving.



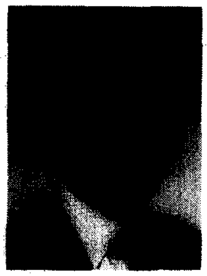
Scott B. Huffman is a doctoral candidate in the Artificial Intelligence Laboratory at the University of Michigan. He received a B.S. in Computer Engineering from Carnegie Mellon University in 1988, and an M.S. in Computer Science from the University of Michigan in 1990. His main research interests include machine learning, intelligent agenthood, and cognitive architectures. His dissertation research focuses on instructable autonomous agents: agents that learn to perform new tasks from tutorial instruction, rather than having to be programmed.

* Corresponding author.
E-mail: dpearson@engin.umich.edu

sons within a world-centered coordinate frame about absolute quantities such as altitude and heading. The level below this concerns rates of change of these quantities, such as the climb-rate (the rate of change of altitude). The next level down provides an even finer grain, involving accelerations (rates of rates of change). Below this level, the system reasons in plane-centered coordinates, mapping desired world-coordinate behaviors into the appropriate changes to the plane's orientation, and eventually bottoming out at the level of operational stick commands.

This hierarchy of concurrently active problem spaces allows Air-Soar to pursue multiple goals both *within* a reasoning level (such as achieving a new heading and altitude, at the highest level) and *between* levels (such as keeping the wings level at the plane orientation level, while performing a climb to a given altitude at the highest level, and climbing at a given rate at the rate-of-change level). The hierarchical approach combined with a uniform representation of goals supports simultaneous reactive behavior at multiple levels of granularity, allowing Air-Soar to respond to constraint failures at lower levels without waiting for them to cause constraint violations at a higher level.

Mark B. Willis received his B.S. from Taylor University in 1991. He is currently employed by BallPoint Systems, Ann Arbor, MI (313) 449-5638. His interests include interactive pen based computing, machine learning and cognitive architectures.



John E. Laird received his B.S. from the University of Michigan in 1975 and his Ph.D. in Computer Science from Carnegie Mellon University in 1983. He is currently an Associate Professor in the Electrical Engineering and Computer Science Department of the University of Michigan. His primary research interests are in the nature of the architecture underlying artificial and natural intelligence. His work is centered on the development and use of Soar, a general cognitive architecture.

Randolph M. Jones is an Assistant Research Scientist in the Artificial Intelligence Laboratory at the University of Michigan. He received his Ph.D. in Information and Computer Science from the University of California, Irvine, in 1989. He came to Michigan after three years as a Research Associate at Carnegie Mellon University and the University of Pittsburgh. His research interests include machine learning, problem solving, psychological modeling, and intelligent autonomous agents.

2. The flight domain

2.1. The nature of flying

Successful flight requires execution of a range of tasks at different levels of complexity and control granularity. High level tasks such as take-offs and landings, maneuvers, such as banked turns and steady climbs, and lower level operations such as keeping the wings level. Often, the pilot must achieve a number of these tasks simultaneously; for example, performing a banked turn while diving. To maintain stable flight, while performing these maneuvers, requires continuous manipulation of the control surfaces.

An important property of flight is that when the plane's spatial orientation changes, as the result of a change in control surfaces or because of an external force, typically many instrument readings will change at the same time. In general, readings are changing continuously which may lead to perceptual overload for the pilot, who must respond selectively to the values that are currently important.

Unexpected events may occur in the course of flying, due to wind, air turbulence or plane malfunctions. Even without such occurances, the plane's behavior is very difficult to predict, because identical control movements produce different effects depending on the plane's precise orientation and motion. Appropriate corrections to the control surfaces must be made rapidly as the plane is very unstable, with slight deviations rapidly leading to significant changes in the plane's motion. [1] Furthermore, changes to control surfaces do not produce immediate changes in the motion of the plane, so the pilot cannot depend on immediate feedback from his/her actions.

2.2. The simulated environment

We have extended the Silicon Graphics Flight Simulator to allow asynchronous control of the plane's throttle, ailerons, elevator and other control surfaces by an external system and to provide limited sensing of the plane's motion. Sensing is

[1] The simulated Cessna is much more unstable than a real Cessna, which we are told is not particularly unstable.

Fig. 1. Instrument panel and display on the flight simulator.

limited to the standard meter readings offered to a pilot (air-speed, heading, climb-rate etc.) as shown in Fig. 1. No attempt is made to model the visual information displayed by the flight simulator and therefore the agent is always forced to fly by instruments alone.

The simulator's model of the plane is updated 20 times a second, which places a tight real-time constraint on the agent's processing and speed of response. Air-Soar is connected to the flight simulator through a local network, as shown in Fig. 2. To help reduce the amount of network traffic, the simulator reports readings to the agent only when they change by more than a pre-specified amount. For example, small changes in the x, y, z coordinates of the plane are not reported. This reduction in communication means that Air-Soar spends about 1/3 of its time sending and receiving messages on the network and 2/3 of its time reasoning. On average, during the course of a turn, Air-Soar takes 0.17 seconds to respond to an event on the simulator, of which 0.06 seconds is spent interfacing with the network. To help overcome perceptual overload, Air-Soar further screens the amount of input by reasoning about the level of accuracy currently required for a task, and rounding the input accordingly. When flying at around 10,000 feet the altitude to the nearest 100 feet is usually sufficient, while when trying to land the altitude might be rounded to the nearest 5 feet.

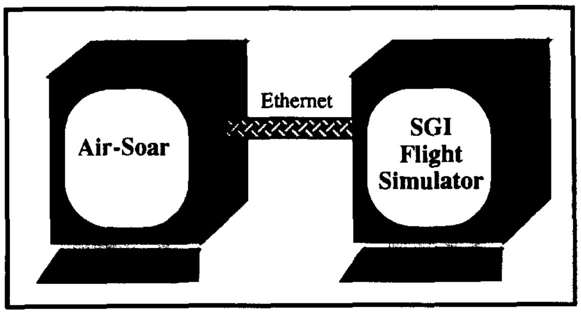


Fig. 2. SGI Flight Simulator and Air-Soar communication link.

3. The reactive control problem

In order to fly an airplane, a system must reason about the concepts and actions involved in directly controlling the flight, such as stick movements and button presses. One possible approach to this problem is to develop a purely reactive system, which would reason only at this level, mapping perceived inputs directly to motor actions (e.g., Pengi [1]). Given enough time and the appropriate feedback, such a system could learn enough such mappings to maintain controlled flight in a large number of situations.

However, there are at least two problems with a purely reactive approach, involving the transfer of knowledge to new situations, and the ability to coordinate reactive behavior with higher-level reasoning and planning. To illustrate these points, consider a typical flight plan for Air-Soar, consisting of a take-off, climbing to a series of altitudes and turning to specific headings before returning to the runway and landing.

Such a plan requires Air-Soar to do more than simply map its current perceptions into actions that keep the plane aloft. It must also reason about high-level goals and constraints. If the system's high-level goals change, it must be able to generate new behaviors that address these goals but that also maintain stable flight. Such flexibility entails a large number of possible interactions between high-level and lower-level goals and constraints. In a single-level reactive system, all of these possible interactions would have to be precompiled into a large number of specific rules.

In addition, Air-Soar's flight plans can be decomposed into intermediate goals and concepts, such as desired turn-rates and accelerations. Such a hierarchical decomposition is a methodological choice that enables the system smoothly to integrate perceptions and various types of goals in order to generate appropriate actions. The hierarchical representation of goals also allows the flexible transfer of knowledge between flight plans when they share particular subgoals. Thus, Air-Soar can execute a wide variety of flight-plans with a relatively small but general knowledge base. The ability to handle a wide variety of situations comes from the dynamic, combinatoric combination of smaller pieces of more general knowledge. The ability to combine intermediate goals and actions obviates the need for a large number of specific, reactive rules. It also allows the system to respond flexibly to particular, unexpected situations, where a single-level system might lack the appropriate specific rules.

Given our choice of representation, the system requires a number of capabilities to carry out general flight plans:

- *Multiple levels*
  The agent must achieve goals at multiple levels simultaneously. For example, during landing the plane must be brought down to zero altitude at a particular place (on the runway). These high level tasks must be achieved at the same time as lower level subtasks such as keeping the wings level and the descent-rate low.
- *Reasoning in different coordinate systems*
  Reasoning about the plane must be done both in terms of world-centered coordinates, where the plane is considered to be a point in space, and in plane-centered coordinates, which are based on the plane's orientation (Fig. 3). Both are required as adjustments to the plane's control surfaces result in changes to the orientation of the plane which in turn lead to changes in the plane's motion.
- *Maintaining constraints*
  The agent must achieve new goals while maintaining other constraints, for example maintaining the current heading during a climb to a new altitude, or maintaining a steady rate of descent while reducing air speed.
- *Responding in real time*
  The agent must respond promptly to goal and constraint violations at each level of its hierar-

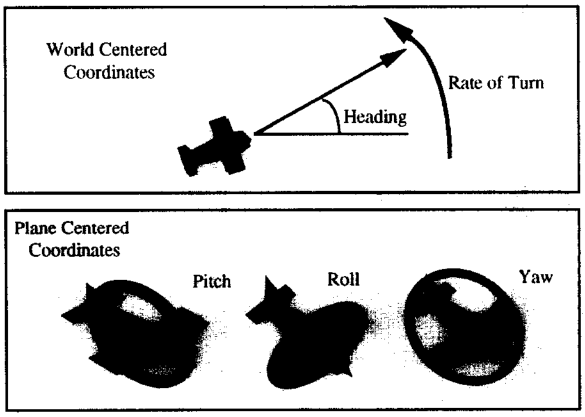


Fig. 3. World-centered and Plane-centered coordinate systems.

chy. High-level goals can be addressed relatively slowly, because they take some time to achieve in any event. In contrast, the lowest-level goals generally require immediate attention and reaction, because quantities at this level change rapidly and failures can cause disastrous results (such as crashing the plane or large deviations from the flight plan).

These capabilities together lead to a difficult control problem where the agent must reason about different classes of goals, across many levels while responding in real-time. The structure of the task makes it most natural to reason in a top-down fashion, such that high-level goals are always active and lower levels follow from them. Because higher level goals are active more of the time, violations of high-level constraints can be noticed immediately, whereas there might be a slight delay in noticing constraint violations at lower levels. Thus, response time *increases* as the granularity of the task decreases.

Unfortunately, this is in direct contrast to the real-time demands of the task. As we have mentioned, the time available to respond to a goal violation *decreases* as the granularity of the task decreases (see Fig. 4). For instance, climb-rate changes more rapidly than altitude, allowing more time for corrections to changes in altitude than for corrections of climb-rate. Thus, the system must be able to detect violations in climb-rate goals at least as quickly as it can detect violations in altitude goals.

Our solution to this problem is to keep all levels of Air-Soar's goal hierarchy active *simultaneously*. When new goals arrive from the flight plan, the system decomposes the problem into intermediate steps until it generates appropriate actions to achieve (or maintain) all of its new goals. When goals are achieved, they stay in memory so they can be monitored continuously. Portions of the goals stack only get regenerated when new high-level goals appear, either in response to changes in the flight plan or changes in the status of higher goals. Because goals are active even when they are already achieved, violations can be detected immediately, regardless of the level at which the violation occurs. In this manner, the system derives the benefits of a hierarchical planning and reasoning system, while also retaining the ability to react to all levels of goal violations in real time.

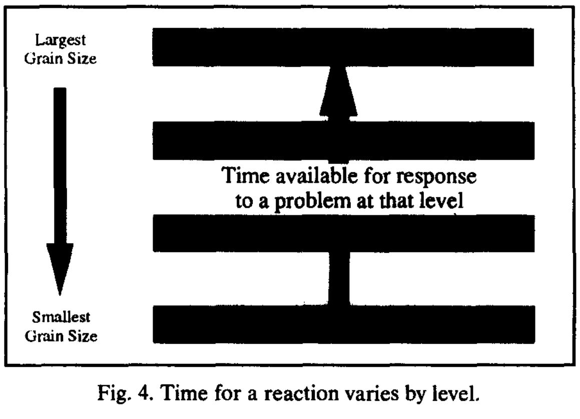


Fig. 4. Time for a reaction varies by level.

4. Air-soar problem spaces

Air-Soar controls the plane through the successive application of operators within a series of subgoals and problem spaces. For instance, when the goal is to reach a new altitude, an operator to do is selected in the highest level space. There is no output command Air-Soar can issue to the simulator to take the plane directly to the desired altitude, therefore an impasse occurs and a subgoal is created of applying this operator. New knowledge can then be brought to bear on this subgoal, by using a different problem space. In this example, Air-Soar reasons that in order to gain altitude the plane's climb-rate must be increased. The operator that represents this still cannot be directly implemented as changes to the plane's control surfaces, so another impasse occurs and a further subgoal, is created to implement the 'achieve new climb-rate' operator.

This naturally gives rise to a hierarchical approach to solving problems using a series of problem spaces, each corresponding to a different level of granularity, as shown in Fig. 5. Air-Soar's problem spaces are:

- *Absolutes space*

  In this space the agent reasons about absolute quantities such as heading, altitude and speed. Reasoning at this level (and in the rates and accelerations problem spaces) is in the world-centered coordinate system. Changes in absolute quantities cannot be achieved by simply setting the control surfaces to a specific position. It is therefore necessary to reason at lower levels until the goal can be directly

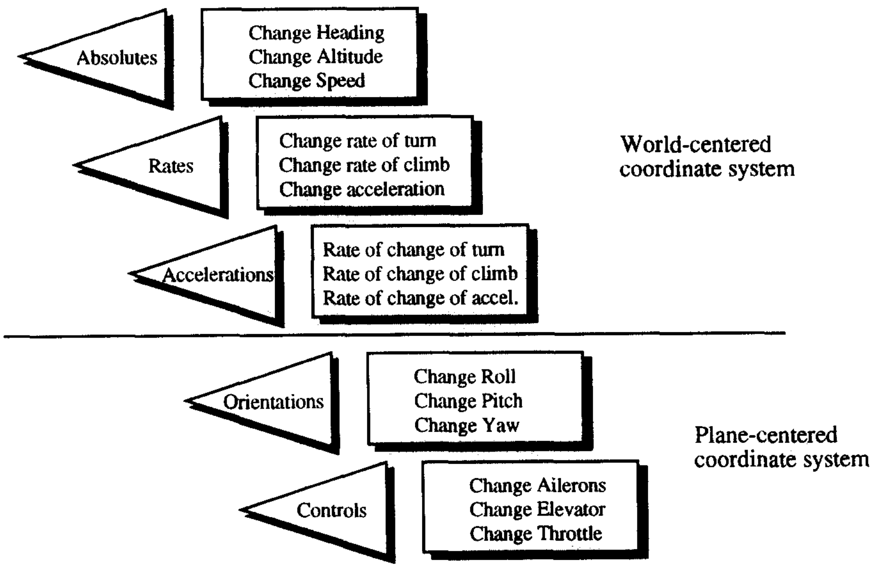


Fig. 5. Air-Soar's problem space hierarchy.

achieved through movements of the control surfaces.

- *Rates space*
  This space deals with rates of change of the absolute quantities. For instance, if the absolute goal is to achieve a particular altitude then reasoning about an appropriate climb-rate would happen in this space.
- *Accelerations space*
  This space deals with changes in rates such as changes in climb-rate and changes in turn-rate. This level of precision is required for certain tasks such as leveling off after a climb. In such a situation it is necessary to decide whether the current climb rate is constant at 0 (and therefore truly level) or whether it is changing. Without being able to reason at this level of detail, the plane tends to oscillate around a desired climb rate.
- *Orientations space*
  To achieve the desired changes in the plane's motion, which are described in world-centered coordinates, the agent must switch to reasoning within the plane-centered coordinate system (see Fig. 3). The mapping from plane orientations to changes in motion is complex; many possible changes in orientation may be used to achieve a particular change in the plane's motion. For example, to lose altitude the plane's pitch could be decreased (putting the nose down), the plane's roll could be increased (causing a loss of lift) or the throttle could be decreased (causing a loss of speed and hence lift).
- *Controls space*
  Finally, the desired orientation of the plane is achieved by changes to the control surfaces (elevator, ailerons, flaps, etc.). Operators at this level send stick commands to the flight simulator to alter the corresponding surfaces on the plane.

As Air-Soar uses purely symbolic reasoning it employs a range of responses proportional to the size of the detected deviation. If the plane begins to dive rapidly the stick will be pulled back farther and faster than for a shallow dive. In the absolutes space only two classes of response are used (one when the goal is almost achieved and one when it is far from being achieved) while at lower levels more divisions are used, allowing the response to be scaled to match the deviation. These proportional responses reduce pressure on the control system by allowing it to make one large correction rather than a number of small corrections.

5. Goals

The domain requires that Air-Soar actively monitor and pursue multiple goals. These may occur at the same hierarchical level, for example climbing and turning to a new heading, or between levels, such as keeping the wings level (a goal within the orientations space) during a climb (a goal within the rates space). Each goal may start and finish independently of others (the correct altitude may be reached before the desired heading) so the system must be able to deal with a dynamically changing set of goals at each level in the hierarchy. Air-Soar meets this requirement by allowing an operator simultaneously to pursue more than one goal within each level of the hierarchy. These goals may be removed and added independently, since they may be achieved at different times.

Air-Soar supports two types of goals [4,8]:

- *Achievement Goals* where the goal is to achieve a particular state. Examples include achieving a particular altitude, a certain level of pitch etc.
- *Homeostatic Goals* (or 'maintenance goals') continuously maintain a constraint. Examples include maintaining altitude during a turn, or keeping a steady rate of descent during a dive.

Air-Soar represents both types of goals in exactly the same way, allowing the system to reason about them with the same knowledge. For example, climbing to a new altitude (an achievement goal) and then maintaining that altitude (a homeostatic goal). Each goal is represented as a target value for a given flight parameter and an acceptable range. Air-Soar only reacts when values fall out of this range. Thus, whenever the current value is within the acceptable range a goal can be considered homeostatic. As soon as the value moves outside the range it becomes an achievement goal. Within the system, goals are not marked as being homeostatic or achievement goals, and any transitions between the two types are implicit.

5.1. Example of goals

To illustrate the way that different types of goals interact, consider an example of a plane initially flying due north (heading 0 degrees) at 15,000 feet and attempting to turn west (heading 90 degrees). The goals of the system are shown in Fig. 6, which for simplicity only shows information from the top two spaces. Achievement goals are in italics while homeostatic goals are in the regular typeface.

The first frame shows the initial situation with

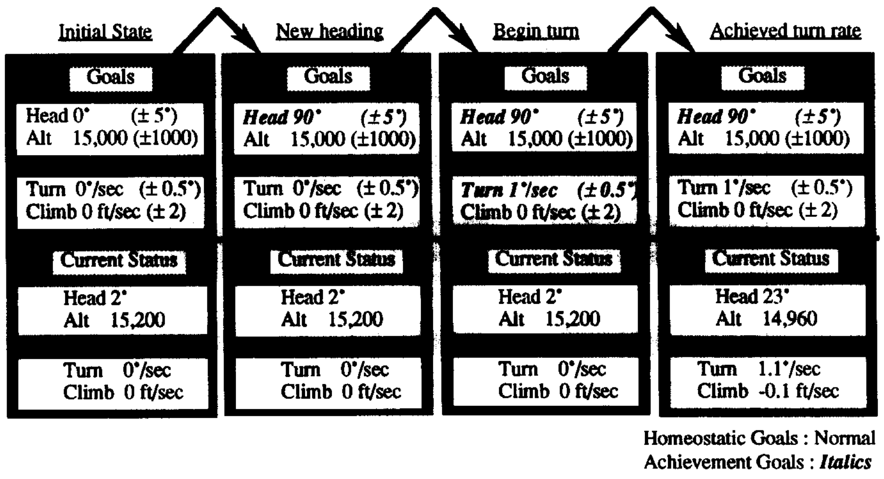


Fig. 6. Example of interacting goals.

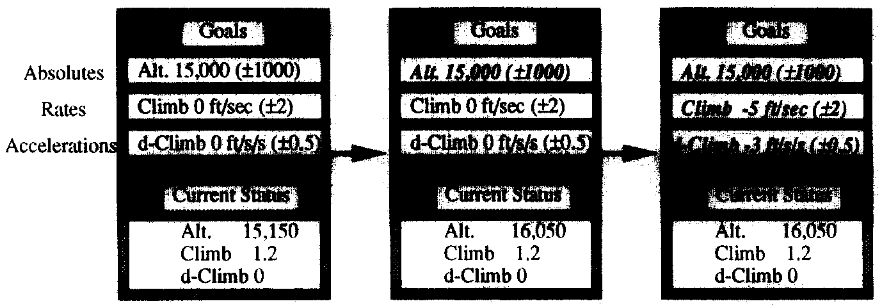


Fig. 7. Reacting to the violation of a top-level constraint.

a series of homeostatic goals for heading, altitude, turn-rate and climb-rate. Initially all of the goals are satisfied. In the second frame Air-Soar makes the decision to turn to a new heading. The new heading goal is not satisfied so the system responds in an attempt to achieve it. Air-Soar updates the desired turn-rate to 1 degree/sec, which in turn is not currently achieved (as shown in the third frame). Air-Soar continues down the goal hierarchy until it can make a change to the plane's control surfaces. The final frame represents the situation once the desired turn rate has been achieved. At this point the goal for turn rate conceptually becomes a homeostatic goal rather than an achievement goal (but notice that no modification is actually required to the data structure).

6. Multi-level reactivity

Typically many or even all of Air-Soar's levels are active simultaneously, trying to maintain or achieve their goals. The hierarchical structure and uniform representation of achievement and homeostatic goals supports reactive behavior at multiple levels of granularity. The reasoning method used to achieve a goal initially is also used to react to a goal with violated homeostatic bounds. The effects of reacting at any given level propagate down to the controls problem space, which ultimately results in commands being issued to the simulator.

Sensitivity at different grain sizes means that Air-Soar is able to respond to unexpected events at the level where the deviation is first noticed, without having to wait for changes to reach higher

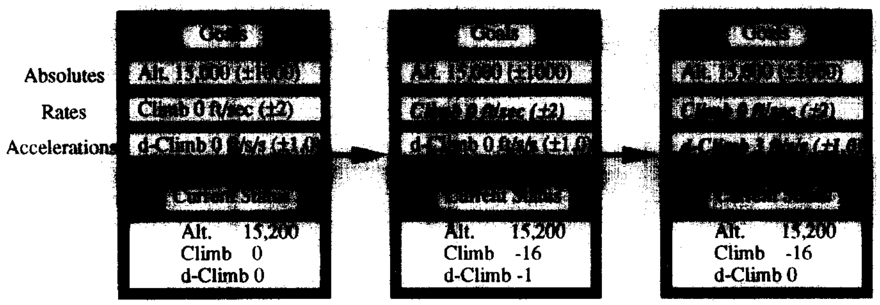


Fig. 8. Reacting to the violation of a lower-level constraint directly.

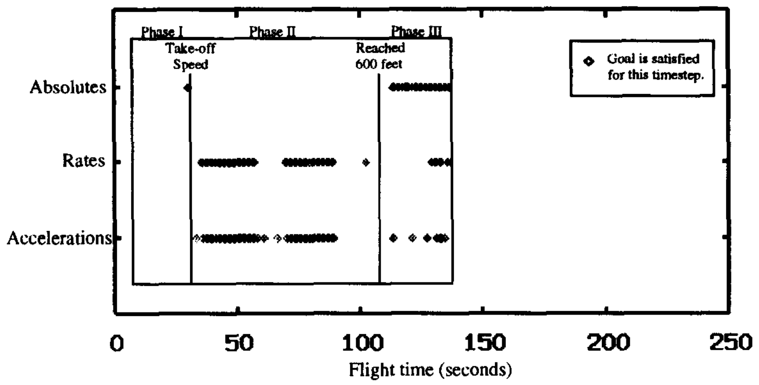


Fig. 9. Trace of satisfied goals during a flight.

levels. Although the system reasons 'top down' it does not have to return to the top level in order to react to lower level problems. Consider an example (shown in Fig. 7) where, after completing a climb, the plane is not perfectly level causing the altitude to continue to change slowly. Although the rate of climb is slow (and within the bounds), after a while (frame 2) Air-Soar notices the altitude is no longer within range and descends to correct it (frame 3).

In this case, the reasoning proceeds from the top level. Consider next the case, shown in Fig. 8. Again Air-Soar is trying to maintain a constant altitude of 15,000 feet. If a sudden downdraft hits the plane causing a steep dive, as shown in frame 2, the climb-rate goal is immediately violated and Air-Soar reacts to the sudden change in rate directly, before the altitude changes enough to be noticed (see frame 3).

An example of this behavior taken from an actual simulation run is shown in Fig. 9. Each point on the figure represents a satisfied goal (within one of the top three problem spaces). When there is no point, it indicates that the goal was not achieved at that moment. During Phase I the plane is rolling down the runway and gaining speed until it reaches take-off speed (indicated by the goal in the Absolutes space being achieved). At this point, the flight plan changes the goal to fly to an altitude of 600 feet. In Phase II (as the

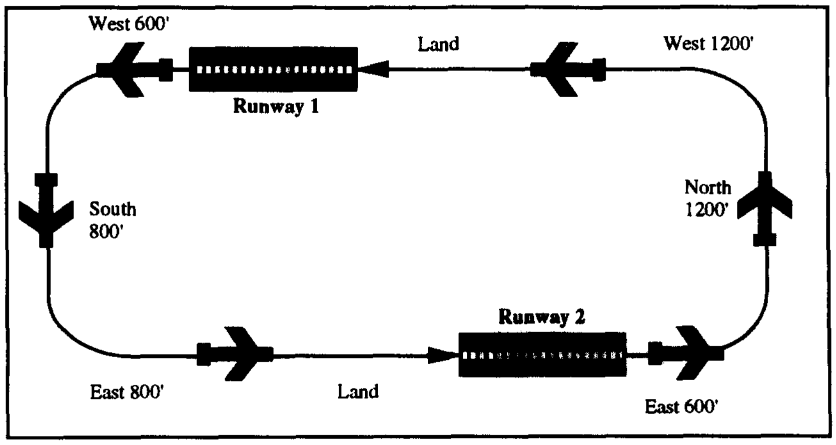


Fig. 10. Circular flight plan for Air-Soar.

plane is climbing) Air-Soar tries to maintain a target climb-rate, as shown by the goal in the Rates space. This shows Air-Soar attempting to achieve a new high level goal (the altitude of 600 feet) while maintaining a homeostatic goal of the correct climb-rate. In the third phase Air-Soar reaches the target altitude at which point turbulence is simulated by putting the plane into an unexpected dive. As the figure shows Air-Soar corrects the change in climb-rate directly, without allowing the plane to violate its homeostatic altitude goal.

The multi-level nature of reactive control in Air-Soar stands in contrast to single level approaches, such as reactive planning in Pengi [1], and reinforcement based approaches (e.g., [12]). Soar's integrated approach to reactive, hierarchical planning and execution also differs from approaches in which planner and executor are separated into different modules (e.g., [6,5,3]). In Soar, 'planning' knowledge (such as knowledge about internal simulation) and 'execution' knowledge (knowledge about how to select and carry out operators in the face of a changing environment) exist within a single architecture, allowing them to combine dynamically as needed. Air-Soar affords the possibility of dynamically creating different problem space hierarchies in response to the demands of particular tasks. This capability differs from methods that employ a static hierarchy of levels (e.g. [2]).

7. Performance and evaluation

One of Air-Soar's flight plans involves flying a circular path between two runways (taking about 30 minutes) as shown in Fig. 10. This includes a take-off, climbing to specified altitudes and turning to specified headings, searching for the next runway (which is at a known location on the ground) and landing. It then repeats the pattern endlessly. In order to land successfully, Air-Soar must line up with the runway, achieve a steady descent without allowing the plane to roll, and touch down with a low air-speed. Air-Soar has proved to be robust in normal flight even at very low altitudes (under 1000 feet) and is able to achieve stable flight within tightly constrained limits (e.g. achieving altitude to the nearest 100 feet and headings to the nearest 5 degrees), all in real time. Air-Soar routinely lands the plane successfully, and then immediately opens the throttle fully and takes off to complete another circuit, allowing the system to fly for a number of hours without the plane crashing or getting lost. We believe, although we have have not yet confirmed this, that the plane only crashes eventually as a result of particularly long network delays or interruptions to the Air-Soar process produced by swapping virtual memory.

To evaluate the effectiveness of the multi-level reactivity approach an alternative system was developed using a different reactive strategy. In this system, which we will call 'top-goal', each time it detects a constraint violation, the system reasons from the top-level goal, proceeding through each level in the hierarchy, and then issuing a command. This is similar to a system reasoning about each goal in turn, and is in contrast to Air-Soar's multi-level approach, which only reasons from the level of the violation down to the stick commands.

The two systems and a human, with experience of flying on the simulator, were first compared over the course of five circuits and judged on the

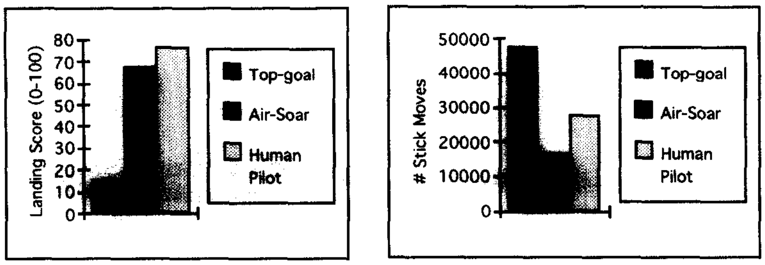


Fig. 11. Relative performance of Air-Soar and replanning approach.

quality of landings (based on factors such as distance from the center-line of the runway, amount of roll and speed of descent), and total number of stick movements, giving a measure of the quality of control during the flight (more movements indicating worse control). The results are shown in Fig. 11. The quality of Air-Soar's landings, based on the SGI Flight Simulator's evaluation system, averages more than 4 times that of the replanning approach, while requiring only about a third of the stick movements. This results from 'top-goal' unnecessarily repeating reasoning at higher levels, which Air-Soar avoids, allowing it to react faster and therefore maintain better control of the plane, both when achieving new goals and maintaining existing ones. Air-Soar's landings are only slightly worse than the human pilot and required fewer stick movements. This is because the human pilot flies the plane in a less stable manner, using steeper changes in altitude and faster turns than Air-Soar. This means more stick movements are needed to maintain control, but higher performance can be achieved.

To further evaluate Air-Soar we compared its ability to respond to simulated 'turbulence' to the alternate system and the human pilot. This was done by manually moving the mouse controlling the plane's stick a measured distance while Air-Soar was controlling the plane, putting the plane into unexpected dives, turns etc.

First the turbulence was introduced during a climb, once the plane had reached a particular altitude, (forcing the plane to turn and dive). In a second trial, the same turbulence was introduced during a turn (forcing the plane to dive and turn in the opposite direction). In the first case, the time was measured for the system (or pilot) to return the plane to the correct course and altitude. In the second case, the time for the plane to return to the original turn. The results, averaged over five runs, are shown in Fig. 12. In both cases Air-Soar performed substantially better than the alternative reactive system. Air-Soar's multi-level reactive behavior allows it to respond more quickly when goals at any level are violated. The human pilot's performance is better than either reactive system as the pilot can maintain control of the plane while performing faster turns and steeper climbs to correct for the turbulence. Air-Soar's performance is more superior in the turning case than the climbing case, because this is an inherently more unstable situation for the plane, so speed of response is particularly important to avoid losing control of the plane.

8. Conclusions and future work

The domain of flight is highly dynamic and unpredictable. Achieving intelligent control in the domain involves reasoning and reacting at multiple levels of granularity, within multiple coordinate frames, both to maintain and achieve multiple simultaneous constraints in real time. We have presented a technique for achieving this control by employing symbolic reasoning within a hierarchy of simultaneously active problem spaces. The technique is embodied in Air-Soar, a system built within the Soar architecture that controls the flight of a Cessna in the SGI flight

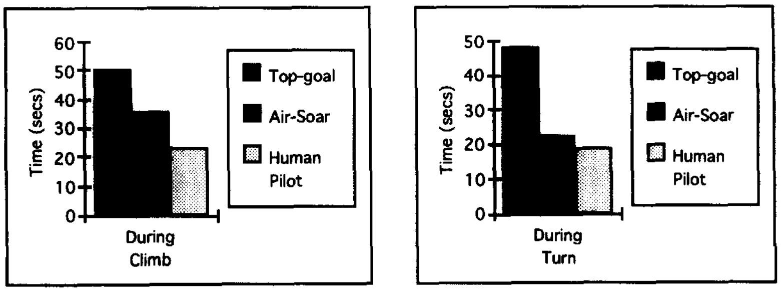


Fig. 12. Time to recover from simulated turbulence.

simulator. Air-Soar's hierarchy of problem spaces allows it to react to changes in instrument measurements at the proper level of granularity; for instance, responding to a drastic fall in climb-rate directly, without having to wait for this change to affect the plane's altitude significantly. By using a uniform representation scheme for goals, Air-Soar can smoothly integrate both homeostatic and achievable goals. The system's performance during regular flight, in response to turbulence, and in landing is comparable to that of experienced humans.

Three main areas in which this work can be extended are control, planning, and learning. First, the system's control ability is limited because it only knows a single mapping between problem spaces in its hierarchy. For instance, if the goal in the highest space (absolutes space) is to achieve a lower altitude, Air-Soar always attempts to decrease the plane's pitch, which in turn is always achieved by pushing forward on the stick. Alternative ways to reduce altitude, such as decreasing thrust or increasing the plane's roll, and alternative ways to decrease pitch, such as by lowering the flaps, are not currently considered.

Second, Air-Soar currently performs a pre-determined flight pattern. We have already integrated Air-Soar with a system that generates tactical flight plans for air combat [7], and we intend to build a more general mission planning capability, allowing the system to produce its own flight patterns from high-level specifications. Flight plans are represented as a series of local decisions rather than a single monolithic plan, allowing them to be integrated into the reactive hierarchy. If an unexpected event occurs, driving the plane away from its original plan, the reactive component returns the plane to the intended path, allowing the plan to continue once the next local decision point is reached.

Third, there are many opportunities for learning in the flight domain. We have experimented with a type of speedup learning, in which Air-Soar learns to alter control surfaces directly in response to higher level goals. For example, the system might learn to pull back on the stick to increase altitude, allowing it to bypass the intermediate levels of reasoning that led to this result. This occurs through Soar's general learning mechanism, chunking (a form of explanation-based learning [10,11]). It increases the reaction speed of the system slightly; in essence, over time it selectively compiles portions of Air-Soar's knowledge into single-level reactive rules.

In addition to speedup learning, which makes Air-Soar's reactions quicker, we plan to incorporate learning to *anticipate* the effects of actions in the environment. For instance, when performing a banked turn a plane has the tendency to lose altitude, because lift from the wings is reduced. Air-Soar can react to this altitude loss; but experienced pilots are able to anticipate and compensate for it before it happens. Learning to anticipate involves extending (and possibly altering) the system's domain knowledge within the various problem spaces.

Beyond anticipation knowledge, a pilot learns to expand the range of possible responses that are available in different situations. Earlier, we mentioned the limitation of Air-Soar's single mapping between problem spaces. This limitation could be overcome by learning mappings for a variety of responses. Such learning could occur through deliberate experimentation in the environment, or by reading textbooks or receiving tutorial instruction. One important advantage of Air-Soar's fully symbolic approach to control is that all of the agent's control structures take the form of knowledge that is open to both well-known and experimental symbolic learning algorithms.

References

[1] Phillip E. Agre and David Chapman, Pengi: An implementation of a theory of activity, *Proc. Sixth National Conference on Artificial Intelligence*, Seattle (1987) 196–201.

[2] Rodney A. Brooks, A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* RA-2 (1) (1986) 14–23.

[3] Paul R. Cohen, Michael L. Greenberg, David M. Hart and Adele E. Howe, Trial by fire: Understanding the design requirements for agents in complex environments, *AI Magazine* 10 (3) (1989) 34–48.

[4] Arie A. Covrigaru and Robert K. Lindsay, Deterministic autonomous systems, *AI Magazine* 12 (3) (1991) 110–117.

[5] M. Drummond and J. Bresina, Anytime synthetic projection: Maximizing the probability of goal satisfaction, *Proc. Eighth National Conference on Artificial Intelligence*, Boston, MA (1990) (AAAI Press) 138–144.

[6] Erann Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, *Proc. Tenth National Confer-*

ence on Artificial Intelligence, San Jose, CA (1992) (AAAI Press) 809–815.

[7] Randy M. Jones, Milind Tambe, John E. Laird and Paul S. Rosembloom, Intelligent automated agents for flight training simulators, *Proc. Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL (1993) 33–42.

[8] Leslie Pack Kaelbling, An architecture for intelligent reactive systems, in: Michael P. Georgeff and Amy L. Lansky, eds., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop* (Morgan Kaufmann, 1986) 395–410.

[9] John E. Laird, Allen Newell and Paul S. Rosenbloom, Soar: An architecture for general intelligence, *Artificial Intelligence* 33 (1) (1987) 1–64.

[10] Tom M. Mitchell, R.M. Keller and S.T. Kedar-Cabelli, Explanation-based generalization: A unifying view, *Machine Learning* 1 (1986).

[11] Paul S. Rosenbloom and John E. Laird, Mapping explanation-based generalization onto Soar, *Proc. National Conference on Artificial Intelligence* (August 1986) 561–567.

[12] Richard S. Sutton, Integrated architectures for learning, planning and reacting based on approximating dynamic programming, In *Proc. Seventh International Conference on Machine Learning* (1990) 216–224.