NASA/CR—2002–211856

CATS-based Air Traffic Controller Agents

Todd J. Callantine
San Jose State University, San Jose, California

October 2002

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

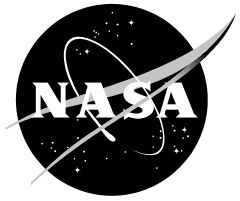
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Telephone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR—2002–211856



CATS-based Air Traffic Controller Agents

Todd J. Callantine
San Jose State University, San Jose, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

October 2002

Acknowledgments

This research was supported by the Advanced Air Transportation Technologies element of the NASA Aviation System Capacity Program, and by the System-Wide Accident Prevention element of the FAA/NASA Aviation Safety Program. It benefited from numerous useful discussions with Everett Palmer, Thomas Prevôt, and Paul Lee at NASA Ames Research Center.

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320
301-621-0390

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

Introduction

This report describes intelligent agents that function as air traffic controllers. Each agent controls traffic in a single sector in real time; agents controlling traffic in adjoining sectors can coordinate to manage an arrival flow across a given meter fix.

The purpose of this research is threefold. First, it seeks to study the design of agents for controlling complex systems. In particular, it investigates agent planning and reactive control functionality in a dynamic environment in which a variety of perceptual and decision making skills play a central role. It examines how heuristic rules can be applied to model planning and decision making skills, rather than attempting to apply optimization methods. Thus, the research attempts to develop intelligent agents that provide an approximation of human air traffic controller behavior that, while not based on an explicit cognitive model, does produce task performance consistent with the way human air traffic controllers operate.

Second, this research sought to extend previous research on using the Crew Activity Tracking System (CATS) (Callantine, Mitchell, and Palmer, 1999) as the basis for intelligent agents (Callantine, 2001). The agents use a high-level model of air traffic controller activities to structure the control task. To execute an activity in the CATS model, according to the current task context, the agents reference a 'skill library' and 'control rules' that in turn execute the pattern recognition, planning, and decision-making required to perform the activity. Applying the skills enables the agents to modify their representation of the current control situation (i.e., the 'flick' or 'picture'). The updated representation supports the next activity in a cycle of action that, taken as a whole, simulates air traffic controller behavior.

A third, practical motivation for this research is to use intelligent agents to support evaluation of new air traffic control (ATC) methods to support new Air Traffic Management (ATM) concepts. Current approaches that use large, human-in-the-loop simulations are unquestionably valuable for this purpose (e.g., Callantine, Prevôt, Smith, and Palmer, 2001; Battiste, et al., 2002; Raytheon, 2002), but pose considerable logistical, fiscal, and experimental control problems. First, data analysis is extremely complicated, owing simply to the large number of participants and data sources in such simulations. In addition, experienced human air traffic controllers working adjacent sectors tend to flexibly adapt to the evolving control problem – potentially shifting to other strategies than those under investigation. In addition, their performance is tightly coupled to the control interface, which in the development phase may support some concepts and supporting strategies better than others. A simple shift in strategy by one controller can change the character of a particular traffic scenario dramatically, which makes experimental comparison of ATC performance under different traffic scenarios difficult. Training a given team of controllers on operations under a new ATM concept for a sufficient period of time could avert such difficulties, but instituting an adequate training program is expensive and logistically difficult.

A more expeditious and inexpensive approach involves testing concepts and interfaces in a part-task setting, in which one human controller subject coordinates with agents controlling traffic in adjacent sectors. Using agents this way ensures that the traffic 'feed' to the subject controller has been 'conditioned' by controlling it

according to a specific ATC strategy dictated by developers of the ATM concept and embodied in the supporting agents. This report qualifies the current-day ('vectoring') conditions under which the agents currently operate and what additional capabilities are required to use them with new ATM concepts in this capacity.

Related Modeling Research

Modeling air traffic controller behavior has generated considerable interest in recent years. Before presenting the CATS agents, this report provides some background on related work. Modeling efforts focus on (1) understanding features of ATC as it impacts the performance of National Airspace System (NAS) ATM, (2) constructing models of operators in complex environments, and (3) embodying such models in intelligent agents.

A model that focuses on NAS performance is MITRE's Detailed Policy Assessment Tool (DPAT) (Heimerman, 1997; Schaefer and Millner, 2001). DPAT is a fast-time simulation of NAS operations; however, it does not model air traffic controller control actions. Another tool, called the Reorganized ATC Mathematical Simulator (RAMS) was developed as part of the FAA/NASA Aviation System Analysis Capability Program (Mondoloni, 1998). RAMS uses rules to resolve conflicts in en route air traffic. It selects a single resolution for a single aircraft that does not create any new conflicts with any other aircraft. RAMS' performance was compared to that of actual controllers resolving the same conflicts, and found to be agreeable. However, the global focus of RAMS is reflected in some of the resolutions it constructs.

Other research has focused on modeling the air traffic control task. Dowell (1998) developed an ecological model and used it to derive the 'cognitive costs' associated with monitoring, planning, and control incurred by a human subject controlling air traffic. Other researchers have conducted empirical analyses of how experienced air traffic controllers assess traffic situations (Niessen, Eyferth, and Bierwagen, 1999), and used them to construct a computational cognitive model of the air traffic controller's task based on the ACT-R framework (Niessen, Leuchter, and Eyferth, 1998). The resulting model (called 'MoFl') was used primarily to investigate the construction of the controller's 'picture' of the traffic situation. While the quality of ATC it simulates is not discussed, the researchers note MoFl was useful as the basis for developing a computer-based tutoring system for training situation awareness strategies (Niessen and Eyferth, 2000).

Leiden (2000) also presents a model of en route controller performance, implemented as a task network model using the MicroSaint modeling tool. The model was not explicitly evaluated for its ability to control air traffic, but instead as a tool for producing predictive human performance measures. Hexmoor and Heng (2000) developed agents for assisting a human tower controller, based on a shared control scheme in which the agents assume control when the human has allowed a situation to become critical. The agents construct prioritized cues of aircraft in a small tower simulation, and use them to detect and resolve conflicts and manage landing clearances.

Finally, ATC agents have been developed that incorporate a model of information processing, situation assessment, and decision making and procedure execution (called 'SAMPLE') to represent distributed decision making in future Air Traffic Management (ATM) systems (Harper, et al., 2002). Agents representing

pilots, controllers, and other participants in the NAS attempt to negotiate solutions to en route conflicts and airspace violations. The agents handled level-flight conflicts using heading and speed resolutions with a high degree of effectiveness.

In summary, the SAMPLE agents and the RAMS system appear to be the only computational ATC agents that have been evaluated for controlling traffic in a closed-loop simulation. Both focus on conflict detection and resolution. RAMS operates over all flight phases, while the SAMPLE agent research has focused on en route airspace, with a focus on negotiation between agents. RAMS is nominally a mathematical simulation, whereas the SAMPLE model attempts to represent skilled human behavior hierarchically, possibly providing a better approximation of human performance.

CATS-based Agents

The CATS-based agents presented in this report compare most closely to the SAMPLE agents. They incorporate a CATS model to represent the main aspects of air traffic control – situation awareness, problem identification, and clearance formulation – in terms of hierarchically decomposed activities. As the agents perform activities, they access skills and control rules, then update their representation of the current operational context, which enables them to perform their next activity.

The agents use heading, route, altitude, and speed clearances to space aircraft in an arrival flow and resolve conflicts as current-day air traffic controllers might, by applying heuristic rules to plan and issue clearances; the agents do not use global optimization methods. Agents in different sectors control traffic from cruise to meter fix crossing (no departures or overflights are as yet included in the traffic scenarios). Thus, the agents address both en route and arrival control problems, including merging traffic flows. Specifically, agents in en route sectors attempt to space aircraft a specified distance in trail (even across flows, if applicable). Agents in low altitude (feeder) sectors attempt to merge arrival traffic and achieve a specified spacing across the meter fix.

The remainder of this report is organized as follows. It first describes the CATS agent approach, and the agent coordination architecture. It then describes the CATS model that represents controller activities, discusses the skills and priorities key activities use, and the flow of control that results when the agents execute activities in real time according to the CATS model. It then describes the rules and skills used to space and separate aircraft. These activities may lead the agents to formulate plans; the report details the plans and triggers for executing a particular plan. Finally, the report presents results of applying the agents to control arrival traffic, and discusses directions for further research.

CATS Agents

CATS 'activity tracking' applications use a model of hierarchically decomposed activities to predict what activities the human operator should perform in a given operational context, and then use these predictions as the basis for interpreting actual operator actions as correct or in error. CATS-based agents are designed to supplant the human operator; they simply execute the activities predicted according the model to control a simulated controlled system (Callantine, 2001).

CATS activity tracking applications take data on the state of the controlled system, and the constraints on controlled system trajectory that define the operator's goals, and use these data to generate a summary of the current operational context. The

context represents the true state of the world, to the extent possible. CATS agents, on the other hand, maintain an internal set of 'beliefs' that may or may not (in the case of agents that err) reflect the true state of world and the attendant operational context.

The type of activity a CATS agent is executing determines how the agents processes beliefs when executing it. Performing a perceptual activity entails transforming information found in a representation of the appropriate visual or auditory 'display' into a set of 'beliefs' about the information. Performing a cognitive activity entails further assimilation of information already present in the agent's belief set, to produce beliefs at different levels of abstraction and/or aggregation, or the results of a decision making process. Manual activities entail executing the activity using a given control; verbal activities entail transmitting some information to another agent. Underlying this scheme is the theory that all salient activities involve transforming or communicating contextual information.

Multi-agent Architecture

Multiple CATS agents operating together have to date relied on a synchronous, 'tick-based' architecture to control processing. A central controller sends each agent a message on each 'tick' (typically one second in duration) that cues each agent to perform a single processing cycle. A processing cycle consists of using the representation of the current operational context to predict which activities need to be performed, and executing those activities. This works well for CATS models structured to enable agents to perform multiple activities at once, in situations where the performance of individual agents is tightly coupled. The flight crew agents discussed by Callantine (2001) provide an example: the agent that represents the pilot-not-flying can set a target value using the Mode Control Panel of the aircraft while simultaneously listening for an ATC instruction on the radio.

As the results section of this report indicates, this scheme is not ideally suited for air traffic controller agents; nonetheless, it was applied as a starting point, as depicted in Figure 1. The architecture uses an 'Agent Hub' process to connect to an Aeronautical Data link and Radar Simulator (ADRS) process. ADRS's function as simulation hubs for the overall air traffic simulation (Prevôt, Palmer, Smith, and Callantine, 2002). The Agent Hub provides four critical functions beyond synchronizing the agent processing times. First, it receives aircraft data from the ADRS and provides it to the agents each time the traffic display is updated (i.e., every twelve seconds). Second, it forwards clearances produced

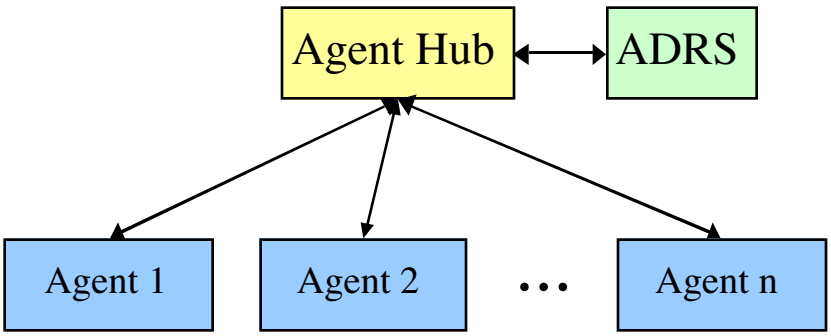


Figure 1. Generic CATS ATC agent architecture.

by the agents to the ADRS, which forwards them to the aircraft simulation. Third, it forwards handoff requests and accepts to the appropriate agent, and finally, it forwards aircraft flight plan and clearance constraint information from an 'upstream' agent to a 'downstream' agent. Thus, when an aircraft is handed off, the upstream agent makes flight plan and clearance information, traditionally recorded on 'flight progress strips,' available to the next controller agent.

CATS Agent Architecture

The CATS ATC agent architecture is derived from the CATS flight deck agents described in Callantine (2001). Figure 2 depicts how knowledge representations in the air traffic controller agents inform each other. The agent hub provides each agent with information about aircraft shown on the agent's traffic display, as well as information on aircraft flight plan constraints and handoffs. The agents have a representation of beliefs about the current task context, and a CATS model that specifies the high-level activity structure. The agent uses its task context beliefs to select an activity to perform during a given processing cycle from its activity model. The activity model represents information about how the activity transforms beliefs. The 'Belief Transformer' uses this information to access a library of skills to generate beliefs that summarize or reformulate traffic information presented on its display, or access a set of control rules to formulate a clearance to issue an aircraft. Control rules may themselves use information from the skill library to support the decision making process. Regardless of the particular activity the agent performs, the agent's belief set is in some way transformed, by adding, removing, or altering the agent's beliefs about the

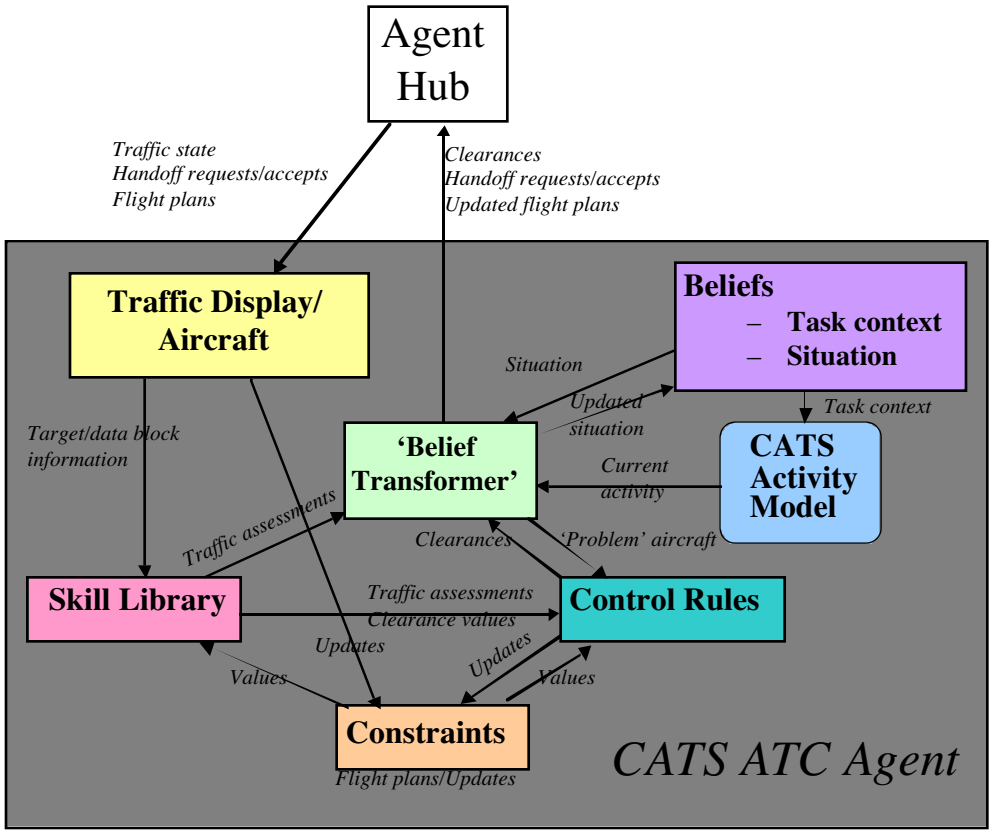


Figure 2. Information flow in CATS ATC Agents.

situation, and by changing the agent's beliefs about the task context to reflect that the particular activity was performed.

Beliefs that represent the current operational context (both the task context, and situational elements of the controlled system) are central to the CATS agent architecture. The agents also use beliefs to represent retrospective context (memory for what the agent has done) and prospective memory for planned activities. Through belief manipulations, the agents implement a prospective memory process model that resembles that of Kliegel, Martin, McDaniel, and Einstein (2002). The CATS agents must perform planning in the context of reactive control, because heuristics do not always consider the impact of aircraft just 'outside' the problem of interest. For example, an agent may identify two aircraft to be in conflict, but in applying heuristics represented in the control rules to generate a solution, the agent may not consider the impact of aircraft immediately behind a conflict aircraft. By planning to issue a clearance to solve the conflict, rather than issuing the clearance right away, the agent has the option to adapt the plan if the conditions necessary to execute it turn out not to be met by the evolving situation. Plans are stored, so that the agent can 'remember' them and evaluate the conditions for executing them.

In general, the control rules govern which clearances should be issued or planned. Agents use the skill library to assess information on the traffic display, and in some cases to formulate clearance values. Examples of skills are detecting conflicts, determining spacing relationships between aircraft, and determining the exact value of a heading vector to issue. Some perceptual activities are purely skill-based, while some cognitive activities reference control rules that require accessing skills themselves. The following sections provide additional detail on key elements of the CATS agent architecture.

- **Maintain situation awareness**
  - **Monitor traffic display**
  - **Scan aircraft**
- **Determine aircraft to work**
- **Manage handoffs**
  - **Accept aircraft**
    - **Accept handoff**
    - **Roger check-in**
  - **Initiate handoff**
    - **Inform other controller**
    - **Issue frequency change**
- **Manage descents**
  - **Issue descent clearance**
- **Manage separation**
  - **Evaluate separation clearance options**
  - **Issue separation clearance**
- **Manage spacing**
  - **Evaluate spacing clearance options**
  - **Issue spacing clearance**
- **Manage nonconformance**
  - **Re-issue clearance**

Figure 3. CATS model for ATC agents.

CATS Model

Each agent encapsulates an activity model to drive the high-level selection of activities to perform. The model represents activities hierarchically, down to the level of actions. The CATS model developed for the CATS air traffic controller agents is shown in Figure 3. The model is roughly comprised of three pieces. The first is the 'maintain situation awareness' activity, and its children, 'monitor traffic display,' and 'scan aircraft.' These activities are devoted to gathering information from displayed traffic information. A second portion is the 'determine aircraft to work' activity, which represents the activity of selecting a problem to address from those currently identified.

The third portion is a collection of 'manage' activities that are performed based on the outcome of the 'determine aircraft to work' activity. Accepting and initiating handoffs are represented by the 'manage handoffs' activity. Note that because the agents operate closed-loop via the ADRS connection, the 'roger check-in' and 'issue frequency change' activities are not required for this implementation, and therefore appear grayed out in Figure 3. 'Manage descents' is devoted to providing aircraft with a descent clearance sometime before they reach their planned top-of-descent points. 'Manage descents' uses knowledge about how far the aircraft can be cleared, given the airspace configuration, as well as control rules for providing positive altitude separation. 'Manage separation' is the activity devoted to resolving detected conflicts, while 'manage spacing' addresses aircraft that, while not technically in conflict, do violate desired in-trail spacing goals. Finally, 'manage non-conformance' addresses aircraft that are not presently in compliance with their constraints; based on the type of non-compliance (lateral, vertical, or speed), the agents re-issue a clearances as necessary to get the aircraft to comply.

Returning now to the 'maintain situation awareness' activity, its first sub-activity is 'monitor traffic display,' which simply generates a belief that reflects which aircraft are currently present in the agent's sector. The second sub-activity, 'scan aircraft,' is devoted to identifying the current control problems that exist for the sector aircraft identified by 'monitor traffic display.' Figure 4 presents a detailed picture of the control problems that the 'scan aircraft' activity identifies. When an agent executes this activity, the agent's skill library is accessed to identify each of the classes of control problems shown in Figure 4. The activity produces beliefs about the existence of various problems that are then referenced by the

Identify:

- Aircraft with plans that need to be executed
- Conflicting aircraft
- Within-flow spacing problems
- Cross-flow spacing problems
- Aircraft that need descent clearances
- Non-conforming aircraft
- Handoffs that need to be accepted
- Handoffs that need to be initiated

Figure 4. Purpose of 'Scan aircraft' activity.

'determine aircraft to work' activity. The model, as implemented, does not decompose the 'scan aircraft' activity into 'identify' activities. This cuts down processing overhead somewhat, but there is no technical reason that an agent could not concurrently execute lower-level 'identify' activities for each of the elements listed in Figure 4.

The 'determine aircraft to work' activity identifies the aircraft or set of aircraft that the controller should address next. When executed, it references the beliefs created during the 'scan aircraft' activity, then selects the aircraft to work based on the priorities shown in Figure 5. Note that these priorities are established based on the effectiveness of the mechanisms used to identify the control problems as much as how controllers are thought to prioritize control problems. For instance, an actual air traffic controller would most likely assign a higher priority to non-conforming aircraft. However, because the agents can sometimes identify non-conformance incorrectly, owing to the need for further refinements to its representation of clearance constraints (see Callantine, 2002), non-conformance is assigned a lower priority in the present implementation.

The priorities shown in Figure 5 reflect the critical importance of executing plans as soon as the conditions for doing so are met. For plans that consist of multiple steps (e.g., vector an aircraft off its route, then to a route-intercept heading, then back on its flight plan route), later steps in the plan may depend on earlier steps for their success. Thus, it is important that the early step is executed as soon as possible. Conflicting aircraft receive the second highest priority, as they may require the most radical steps to address. In the present implementation, in order for the agents to generate a vector, aircraft must be in conflict. Next are aircraft that need to be spaced. The agents use speed clearances to space aircraft, and it is relatively easy to determine safe speeds. Next after spacing aircraft are aircraft that require descent clearances. The agents are configured such that, in the absence of conflicts during the descent, they are certain to get aircraft down in time; otherwise these aircraft would also receive a higher priority. Finally, handoff acceptance, and handoff initiation receive lowest priority. This reflects observed human air traffic controller behavior, in that controllers typically do everything else that needs to be done, then take (or issue) several handoffs consecutively.

CATS Agent Beliefs

An important feature of the 'determine aircraft to work' activity is, depending on

Priority:

1. Aircraft with executable plan
2. Conflict aircraft
3. Within-flow spacing problems
4. Cross-flow spacing problems
5. Aircraft that need a descent clearance
6. Aircraft that need to be handed off
7. Non-conforming aircraft
8. Handoffs that need to be accepted

Figure 5. Priority used for selecting control problems in the 'determine aircraft to work' activity.

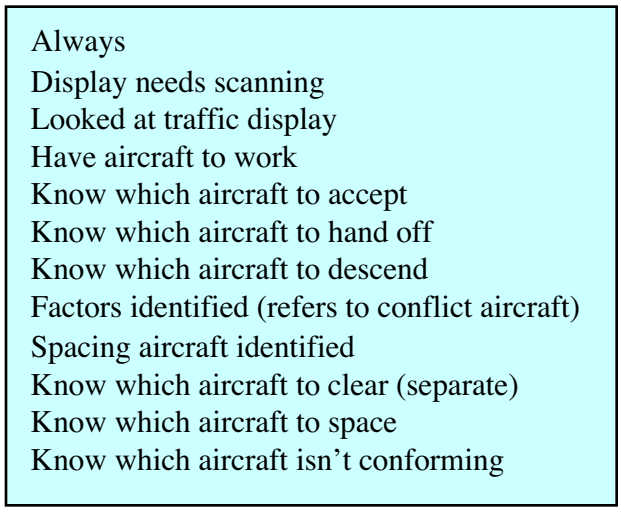


Figure 6. Task context beliefs.

the results of the assessments it performs, different beliefs are added to and removed from the agent's current set of beliefs about task context. This is an extension of the CATS agent framework presented in Callantine (2001) (see Appendix A for a discussion of the extended CATS model file specification and associated processing issues). The beliefs used to represent task context are shown in Figure 6. In essence, the last several beliefs ('know which…' and '… identified') correspond to the type of control problem identified in 'determine aircraft to work.' The 'always' belief ensures that the CATS model's top-level activity, 'control traffic,' is always active, so that the top-down search used to predict activities in CATS has the opportunity to find one.

In addition to the task context information that may be included in an agent's belief set, beliefs about the traffic control situation may also be included. Figure 7 depicts some specific beliefs about the current situation, memory for when problems were last addressed, and prospective memory for plans. Retrospective memory for when problems were last addressed is important because it takes some time for the displayed traffic information to reflect for the effects of a clearance. Because the problem may appear to continue to exist for a period of time, without the 'check' beliefs an agent will repeatedly address the same (higher priority) problem to the exclusion of other problems — even if they have actually already addressed it. Prospective memory, in the form of plans associated with particular aircraft, and especially as reflected in beliefs about an aircraft that has a plan that needs to be executed immediately, is a vital part of the CATS agent scheme. This is because control rules do not address 'other' aircraft that also impact the control problem of interest.

The justification for including beliefs in the format shown in Figure 7 is for displaying them, and also in looking ahead to a scheme for generating errors by

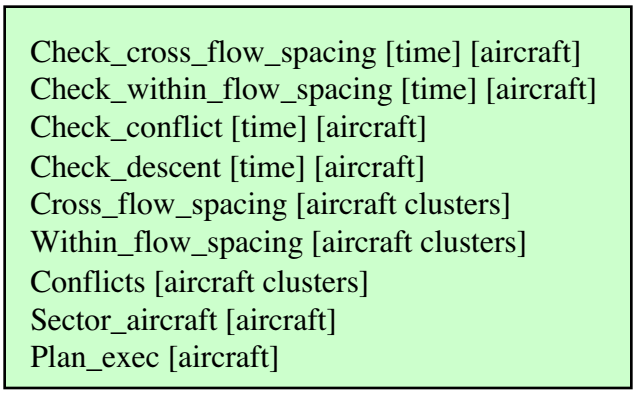


Figure 7. Traffic control situation context beliefs.

altering the contents of the beliefs. In the current implementation, however, other important information for applying the control rules, retaining the contents of an about-to-be-issued clearance, and of course, the state and constraint information for each aircraft, is maintained using representational objects and variables in code. The most important of these involves 'role bindings' for aircraft. The agents use role bindings to provide a general way to specify a frame of reference for the application of control rules (see Horswill and Zubek, 1999). When agents initially execute the 'monitor traffic display' activity (the first activity to perform after the traffic display is updated), they access their skill library to 'bind' aircraft to crucial roles (e.g., 'inFront,' 'inFrontSequence,' 'firstConflict,' etc.). For each bound role, the agents also access perceptual skills to assign a bit-vector of attributes. This information is simply too fluid and too complex to represent as a 'belief string' (and would defeat the purpose of representing it as a bit-vector to begin with). This issue will be revisited below, in the discussion of control rules.

Figure 8 summarizes how the agents work, at the high level governed by the activity model. In essence, the structure of the activity model, and the beliefs about control problems, plans, etc., yield a flow-of-control that reflects the priorities used by the 'determine aircraft to work' activity. The flow of control can be considered at least somewhat congruent with that observed in actual air traffic controllers, although further research is needed in this area. A comprehensive set of priorities from experienced controllers may be difficult to elicit at a more detailed level than that of the situation assessment studies performed by Niessen, et al. (1999). This is because the context information used is largely perceptual and likely far richer, and the prioritization process is more deeply ingrained as skill.

Control Rules

This section describes the heuristics the agents use to determine clearances in the current implementation. The agents have two top-level entry points to the rule base depending upon whether they are addressing a 'spacing problem,' related to putting aircraft a specified distance in trail,

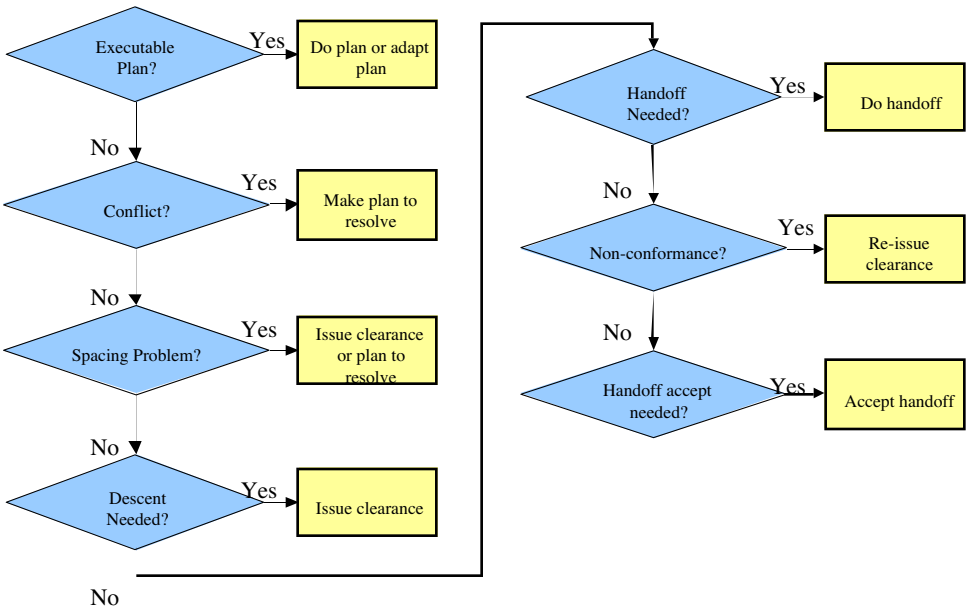


Figure 8. Flow of control resulting from the CATS model used by the CATS ATC agents.

- If excess spacing, **speed up/plan to match speeds**
- If insufficient spacing:
  - If no aircraft in front of *front* or behind *back*, **stagger speeds**
  - If no aircraft in front of *front,* but aircraft behind *back*, **speed lead aircraft up**
  - If aircraft in front of *front*, but not behind *back*, **slow back aircraft**
  - If aircraft in front of *front*, and behind *back*, require vectors (handle as conflict using separation rules)

Figure 9. CATS ATC agent spacing rules.

or a 'separation problem,' related to resolving a conflict, including those that occur at a merge point. In general, the agents solve spacing problems with speed, and separation problems with vectors. However, by adjusting what qualifies as a conflict to use the in-trail spacing requirement as the separation value, the agents can solve spacing problems by issuing vectors.

The agents use the spacing rules shown in Figure 9. Four strategies to space aircraft are incorporated in the rules: 'speed up/plan to match speeds,' 'stagger speeds,' 'speed lead aircraft up,' and 'slow back aircraft.' The rules access two skills to determine which rules to apply: 'excess spacing' and 'insufficient spacing.' When aircraft are evaluated by the spacing rules, one is designated 'front' and the other is designated 'back.' In cases where the specified spacing is insufficient, the strategy invoked is determined using the role bindings attached to the 'front' and 'back' aircraft. Each of these aircraft, in turn, has aircraft bound to roles called 'inFront' and 'behind,' and to the roles 'inFrontSequence' and 'behindSequence.' The latter role bindings refer to aircraft that are actually in front or behind, but are in 'adjacent flows' (which is the case when two traffic streams traverse a sector, to be merged in a downstream sector). By referencing the role bindings, the spacing rules can base strategy determination on the presence of 'other' aircraft in the vicinity of the spacing problem.

Agents can issue clearances immediately for all of the spacing strategies, except when there is excess spacing. In this case, the agent immediately issues the back aircraft clearance to accelerate to 'close the gap,' and establishes a plan to match the lead aircraft's speed when the spacing reaches the desired spacing. The plans that the agents use are described in the section that follows. As noted above, spacing problems that occur in the midst of other aircraft are handled by separation rules using vectors.

Figure 10 shows the separation ('vectoring') rules that the agents use. Vectoring to resolve conflicts is more complex than issuing speed clearances for spacing, because nothing can be done without prior planning. Rules are again structured to reference a 'front' and 'back' aircraft and role bindings are used to reference aircraft in the vicinity of those in conflict. Because the agents only address aircraft in arrival flows, conflict angles are small, and as such, the 'front' and 'back' designations make sense. Separation rules for 'opposite direction'

If *front* directly in front and no aircraft behind *back*:
- If merge, **plan to merge**
- Otherwise, **plan minimal offset**

If *front* directly in front and aircraft behind *back*:
- If merge, **plan to merge**
- Otherwise, **plan minimal offset** and **plan to match vectors** for aircraft behind *back*

If *front* in front sequentially and no aircraft behind *back*:
- If merge, **plan to turn in to merge**
- Otherwise, **plan to vector and turn back**

If *front* in front sequentially and aircraft behind *back*:
- If merge, **plan to turn in to merge**
- Otherwise, **plan to vector and turn back** and **plan to match vectors** for aircraft behind *back*

- Multiple aircraft conflicts
  - Only handle in cases of merge, using **plan to merge** or **plan to turn in to merge**

Figure 10. CATS ATC agent separation rules.

conflicts, and conflicts between arrivals and 'other' aircraft require a prioritization scheme for which aircraft should best be vectored, which is a subject of further research.

Several skills are accessed by the separation control rules. Specifically, agents have a skill for determining whether the conflict in question is a 'merge condition,' and skills for assigning values for each type of vectoring plan. A given vectoring strategy consists of multiple plan steps (e.g., the strategy 'plan to vector and turn back' consists of three plans: 'delay vector,' 'turn back vector,' and 'resume route'). Some separation rules also address planning for vectoring aircraft bound to roles. For example, when an aircraft is directly behind the 'back' aircraft (i.e., bound to its 'behind' role), agents apply the strategy 'plan to match vectors' to that aircraft in the same planning pass, so that the aircraft behind the 'back 'aircraft can be delay-vectored first. Experienced air traffic controllers have been observed to use this technique. Conflicts involving more than two aircraft present a special case. The agents cannot reasonably sort out how to plan to turn multiple aircraft, except when all are merging at the same point. In this case, the role bindings are used to establish a sequence, which breaks the conflict into a number of conflict pairs.

The agents use knowledge about when aircraft in their particular sector should descend in order to hand them off at the required altitude. Control rules also incorporate positive altitude separation. These rules attempt use the aircraft bound to the role 'firstConflict' to gauge whether the aircraft can be cleared all the way down to the required sector-exit altitude for arriving aircraft, or whether an intermediate altitude above the conflict aircraft is required. Agents re-address aircraft cleared to an intermediate altitude periodically (via 'check_descent' beliefs)

to determine if they can be cleared to a lower altitude, until the aircraft are cleared all the way down to the exit altitude for the sector.

Plans

From the discussion of control rules above, it is clear that planning plays a critical role in the successful application of control strategies. Agents are not capable of globally assessing clearance options under the 'application of heuristics' scheme employed in this research. Thus, plans are constructed of steps that have, based on their type, a set of conditions under which the plan should be executed or adapted (which includes abandoning the overall plan or a step of the plan altogether).

Several plans (plan 'steps') were identified for inclusion in the planning strategies used by the control rules. Figure 11 shows these plans as they relate to the lateral, vertical, and speed dimensions of control. Five were never used (shown grayed-out in Figure 11). The vertical plans are supplanted by immediate clearances (i.e., there was no perceived need to plan these actions). The lateral plans 'direct-to,' and 'meter fix direct-to' were also never needed; the plan 'return to route' covers both of these functions. Lastly, the speed plan 'allow to pass' introduced difficulties with role bindings (e.g., by its very nature, at some point the aircraft 'inFront' becomes the aircraft 'behind,' etc.). Because the agents bind roles before the plans are checked for execution, this created problems. 'Naturally faster' aircraft are therefore obliged to stay behind slower aircraft for spacing, under this scheme.

The remaining plans shown in Figure 11 combine to cover the control strategies implemented by the control rules. The agents use the lateral plans to implement strategies used in the separation rules. Lateral planning strategies entail, first, a plan to 'delay vector' (or 'match planned lead delay vector'), followed in some cases by a 'turn back vector' (or 'match planned lead turn back vector'). Finally, lateral planning strategies add a 'return to route' plan (or a 'return to heading' plan, if the aircraft has no known route to rejoin). The agents determine the values of the vectors encompassed by the plans using skills in their skill library.

The agents use speed plans to implement strategies used by the spacing rules. Speed plans are given in pairs, depending on whether a Mach number or indicated airspeed is called for. While the spacing rules shown in Figure 9 reference only the 'match lead' speed or mach plans explicitly, the agents use the remaining low-level 'accelerate' and 'decelerate' speed plans as necessary to implement the 'stagger speeds' strategy.

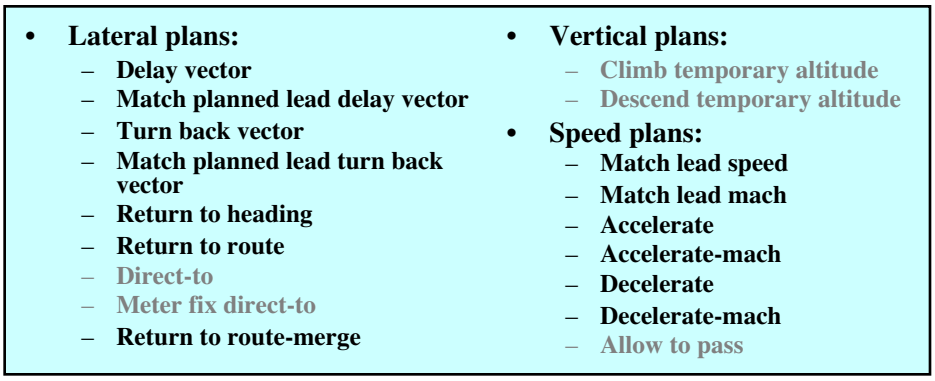


Figure 11. Plans to implement planning strategies used in control rules.

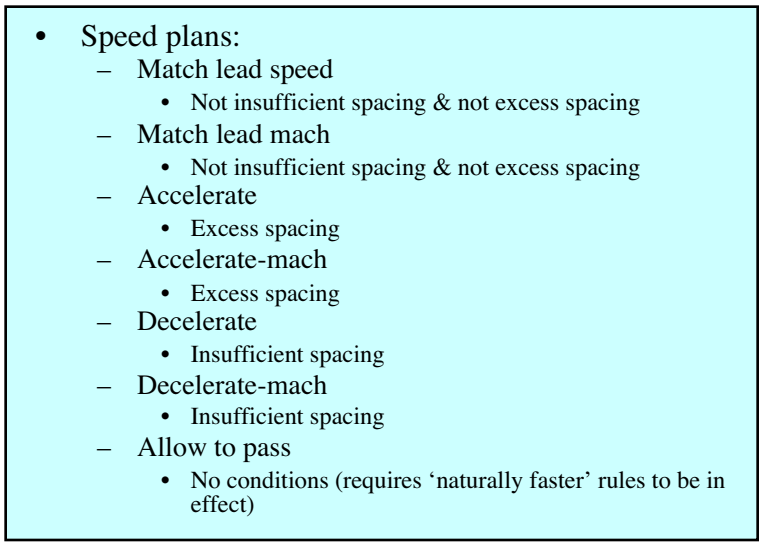


Figure 12. Execution conditions for speed plans.

A critical feature of each type of plan is the set of conditions under which the agents should execute it. Each plan has conditions for execution that relate to the control strategy that the agent was following when developing the plan for an aircraft. The execution conditions reference roles that the agents bind to the plan at the time it is developed. A plan records, for example, the 'front' aircraft against which proper spacing is to be measured. In addition, a plan records the time the agent developed it and, in the case of turn-back vectors, the value of the vector and the time at which the agent plans to execute it.

Figure 12 shows the execution conditions for speed plans. The execution conditions reference the agent's skills to detect insufficient and excess spacing. For example, when an agent implements the strategy termed 'speed up/plan to match speeds,' it clears the 'back' aircraft to a faster speed and also logs a 'match lead mach' plan for it. Each time it executes the 'scan aircraft' activity, the agent checks whether the 'back' aircraft has closed to the desired distance behind the 'front' aircraft referenced by the plan. If it has, it executes the plan to match the lead aircraft's Mach by issuing the appropriate Mach number as a clearance. As shown in Figure 13, the conditions for executing lateral plans are more complex. Typically, they include conditions for executing the plan under circumstances where vectoring skills handled the situation well, along with some that function as 'stop-gap measures.' Such conditions are required when, for example, the agents vector aircraft toward a sector boundary, or when the aircraft flies on a vector past a waypoint that was to be the point at which the aircraft rejoined the route. Some conditions are included to ensure that aircraft cross the meter fix. Still others make sure an aircraft is not handed off before it has been cleared to rejoin its filed routing, as the plans with which the agent sought to accomplish that are not transmitted to the receiving agent; only the current clearance constraints that the aircraft is following are transferred. In short, deficiencies in the agent's skill library, together with the dynamics of addressing various control problems, necessitate ways to adapt plans to ensure the agents issue reasonable clearances.

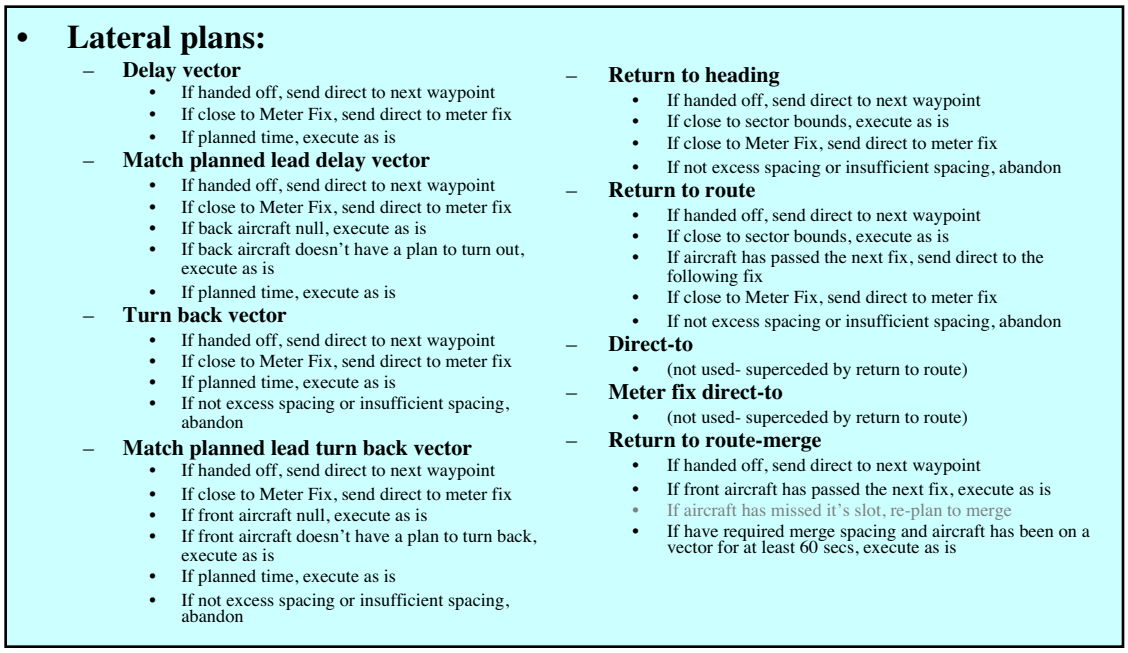


Figure 13. Execution conditions for lateral plans.

Example Operations

This section presents two examples of how the agents control traffic. The first example describes how the agents address one type of in-trail spacing problem; the second example describes a 'merge problem.'

The example spacing problem is captured in Figure 14, which depicts a situation with two in-trail flows moving roughly from left to right. After receiving a traffic update from the ADRS, the agent executes the 'monitor traffic display' activity and acquires the belief that the 'sector aircraft' include AAL497, AAL630, and AAL508 (AAL137 is not yet in this set, but will be shortly). The successful execution of 'monitor traffic display' results in the agent adding the 'task context' belief 'looked at traffic display' (see Appendix A), which enables execution of the 'scan aircraft' activity. The agent executes 'scan aircraft' on the next tick, upon which the agent accesses its skill library to assess the 'sector aircraft' traffic. The skill library checks for conflicts (i.e., separation problems) and spacing problems, and adds a 'within_flow_spacing' belief for AAL630 and AAL508. AAL630 is bound to the role 'behind' in the aircraft AAL508, and AAL508 is bound to the role 'inFront' in AAL630. Successful completion of the 'scan aircraft' activity installs a 'have aircraft to work' belief, which triggers the 'determine aircraft to work' activity.

When the agent executes 'determine aircraft to work' activity, it notes the presence of the 'within_flow_spacing AAL508 AAL630' belief in the agent's belief set. If the agent has a 'check_within_flow_spacing' belief for this aircraft pair that tells the agent when it can reasonably re-address this particular spacing problem, it will move on to other problems. Similarly if higher priority problems exist (i.e., aircraft with a plan that needs to be executed, or aircraft in conflict — see Figure 5), the agent will address those problems first. However, assuming the agent holds neither of these beliefs, the 'determine aircraft to work' activity has the effect of installing a 'spacing aircraft identified' belief in the agent's belief set. This enables the agent to

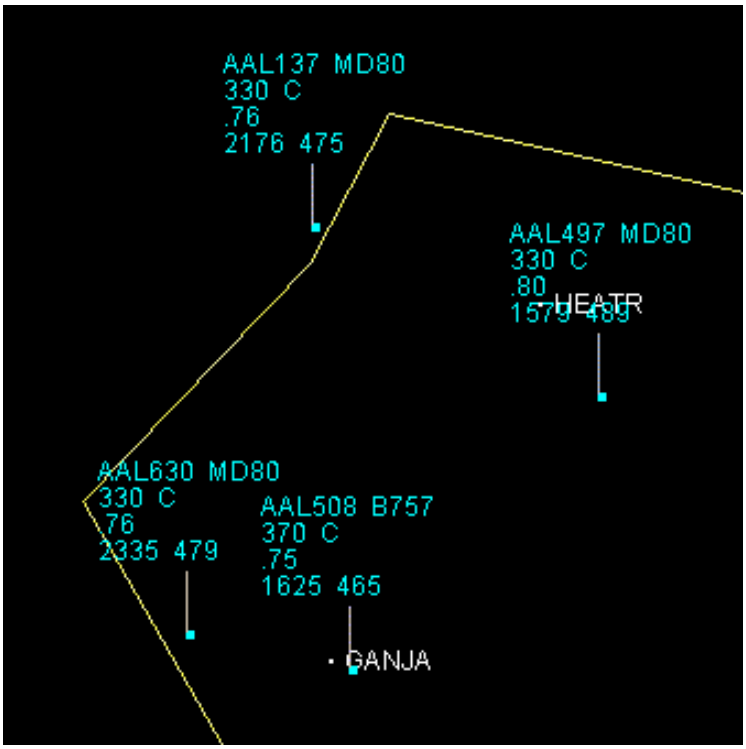


Figure 14. Example in-trail spacing problem involving AAL508 and AAL630.

execute the 'evaluate spacing clearance options' sub-activity of 'manage spacing' on the next tick.

The activity 'evaluate spacing clearance options' actually accesses the agent's control rules to determine the appropriate control strategy. Because AAL508 and AAL630 are more than ten nautical miles in trail (plus a small tolerance), the control rules determine the appropriate strategy is 'speed up/plan to match speeds' (see Figure 9). Using this strategy, the agent accesses an aircraft performance database, and sets up a clearance to accelerate AAL630 to its maximum Mach. It also logs a plan to 'match lead mach' with AAL630. It completes execution of 'evaluate spacing clearance options' by adding the task context belief 'know which aircraft to space' to the agent's current beliefs. Thus, on the next tick, the agent's model indicates that the 'issue spacing clearance' activity should be executed. When the agent executes the 'issue spacing clearance' activity, the agent sends the clearance to accelerate AAL630 to its maximum Mach to the ADRS, via the Agent Hub.

Processing then continues, with the agent checking AAL630's plan to match the Mach of AAL508 each time it executes the 'determine aircraft to work' activity. At some point, AAL630 will have closed the gap with AAL508 — the condition for executing the 'match lead mach' plan. At this time (assuming this plan is the first requiring execution that the agent finds), the agent readies the appropriate clearance for AAL630, and acquires the 'know which aircraft to clear' belief. On the next tick, the agent issues the clearance which, when AAL630 complies, results in proper in-trail spacing between AAL508 and AAL630.

The second example concerns the merge situation shown in Figure 15; the figure shows the solution of merge problem in progress. The agent first identifies AAL6080 as in conflict with UAL1114. When the agent selected this conflict as highest priority in the 'determine aircraft to work' activity, it performed the 'evaluate separation clearance options' activity. The agent accessed its control rules, and used the role bindings for each aircraft to determine that the UAL1114 is in front of AAL6080 sequentially, and that there is no aircraft immediately behind AAL6080. It also determined that the two aircraft are merging at UKW, causing it to apply the strategy 'plan to turn in to merge' (see Figure 10). When the agent applied this strategy it cleared AAL6080 to a 095 heading, and logged a 'return to route – merge' plan with AAL6080. The agent is now in the position to repeatedly evaluate the conditions for executing this plan (see Figure 13) each time it executes the 'determine aircraft to work' activity.

After AAL6080 started its turn, the agent determined that AAL6080 was also in conflict with DAL323 (see Figure 15). The agent then followed the same solution method as it did for the first conflict. When the agent applied its control rules to DAL323 and AAL6080, it again determined that these aircraft are to merge at UKW, and used the aircraft's role bindings to determine that, in this case, AAL6080 is in front of DAL323 sequentially, and there is no aircraft behind DAL323. Thus, the agent again applied the 'plan to turn in to merge' strategy (see Figure 10), which resulted in a 245 heading for DAL323, together with a plan for DAL323 to 'return to route – merge.'

Figure 15 shows the situation after both AAL6080 and DAL323 have begun to turn onto their new headings. Each time the agent executes the 'determine aircraft to work' activity, it evaluates the status of these aircraft in relation to the aircraft in front of them at the time their plans were formulated. After the plans have been in effect for sixty seconds (to avoid

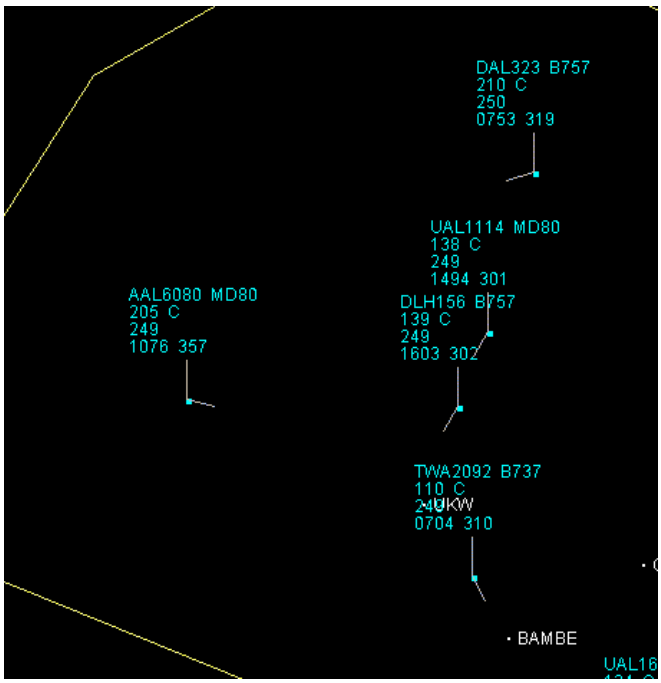


Figure 15. Example merge problem involving UAL1114, AAL6080, and DAL323.

immediate turn-backs in certain situations), the agent checks AAL6080's distance to UKW versus UAL1114's, and DAL323's distance to UKW versus AAL6080's. Eventually, the agent finds that the heading vector has produced the required merge spacing between UAL1114 and AAL6080, and executes AAL6080's 'return to route – merge' plan. This sets up a clearance for AAL6080 to proceed direct UKW. After AAL6080 has begun to converge on UKW, eventually the conditions for executing the plan for DAL323 to 'return to route – merge' will be met, and the agent will issue a clearance for it, too, to proceed direct UKW, completing the merge.

In summary, the CATS air traffic controller agents use periodically updated data on the state of arrival traffic to generate a 'picture' of the traffic situation. Agents update beliefs about the situation by executing activities represented in a hierarchical model according to beliefs about the task context. Based on assessments of what needs to be done, and a prioritization scheme for choosing the problems to address, the agents formulate clearances and issue them, or plan clearances to issue when the appropriate conditions are met. To do this, the agents reference a library of skills for assessing displayed traffic, and a set of control rules that incorporate strategies for spacing and separating aircraft, as well as employing positive altitude separation. The remainder of this report offers an appraisal of how well the agents perform, and discusses further research.

Performance Assessment

It is impossible to determine how well the agents perform, and what modifications might be necessary to improve them, without applying them to an exhaustive set of traffic flow conditions and airspace configurations. However, with an eye toward their practical use supporting studies of operational ATC concepts, a preliminary performance assessment was conducted. NASA ATM research has focused on airspace centered on Dallas-Fort Worth Center (ZFW) with arrival flows to Dallas/Fort Worth (DFW) airport (for details on the test airspace and the concepts under study, see Callantine, et al., 2001, and Battiste, et al., 2002). Thus, airspace in this region was chosen for testing the agents. Agents were implemented to control the Wichita Falls High Altitude sector (SPS), the Ardmore High Altitude sector (ARD), and the sector responsible for merging arrival flows from these sectors to cross the BAMBE meter fix, the Bowie Low Altitude sector (UKW). Figure 16 shows the architecture for testing the agent implementation. Figure 17 shows a screen snapshot of the UKW agent controlling traffic.

Traffic scenarios were derived from those currently being used as the baseline scenarios for NASA studies extending the research reported in Callantine, et al. (2001), and from current-day traffic flows. The 'new concept' scenarios constructed specifically for testing advanced ATM concepts do not necessarily reflect the traffic flows that the selected sectors would experience during normal current-day operations. On the other hand, the 'current-day' arrival rushes may be too difficult to manage without the capability to issue holding clearances. Thus, as with all ATM research, traffic scenario selection poses a problem.

Nonetheless, two 'new concept' baseline scenarios ('A1' and 'B1') were used as is, and modified (lightened) twice each, to produce six test scenarios, and three additional 'current-day' scenarios were produced by reducing the traffic in a 'current-day' scenario ('C1'), yielding a total of nine agent test scenarios. Figure 18 shows the number of arrival aircraft in each

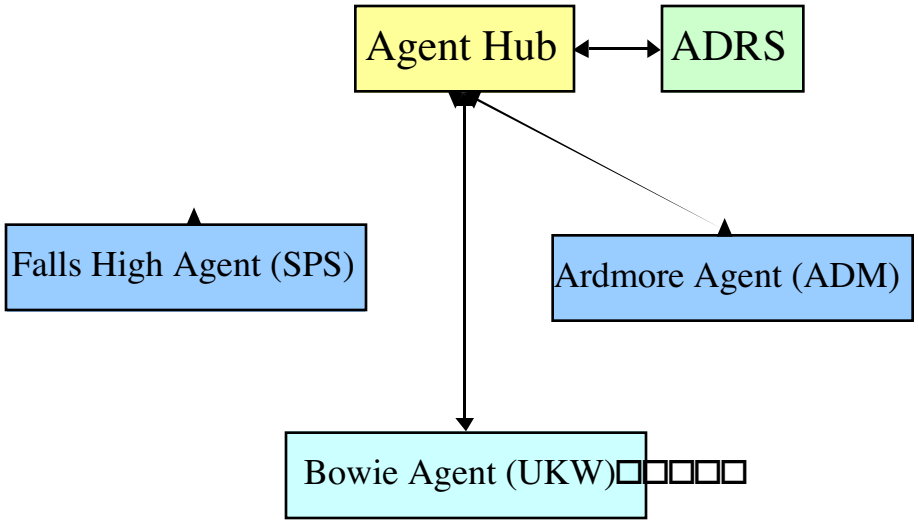


Figure 16. Three-agent test architecture.

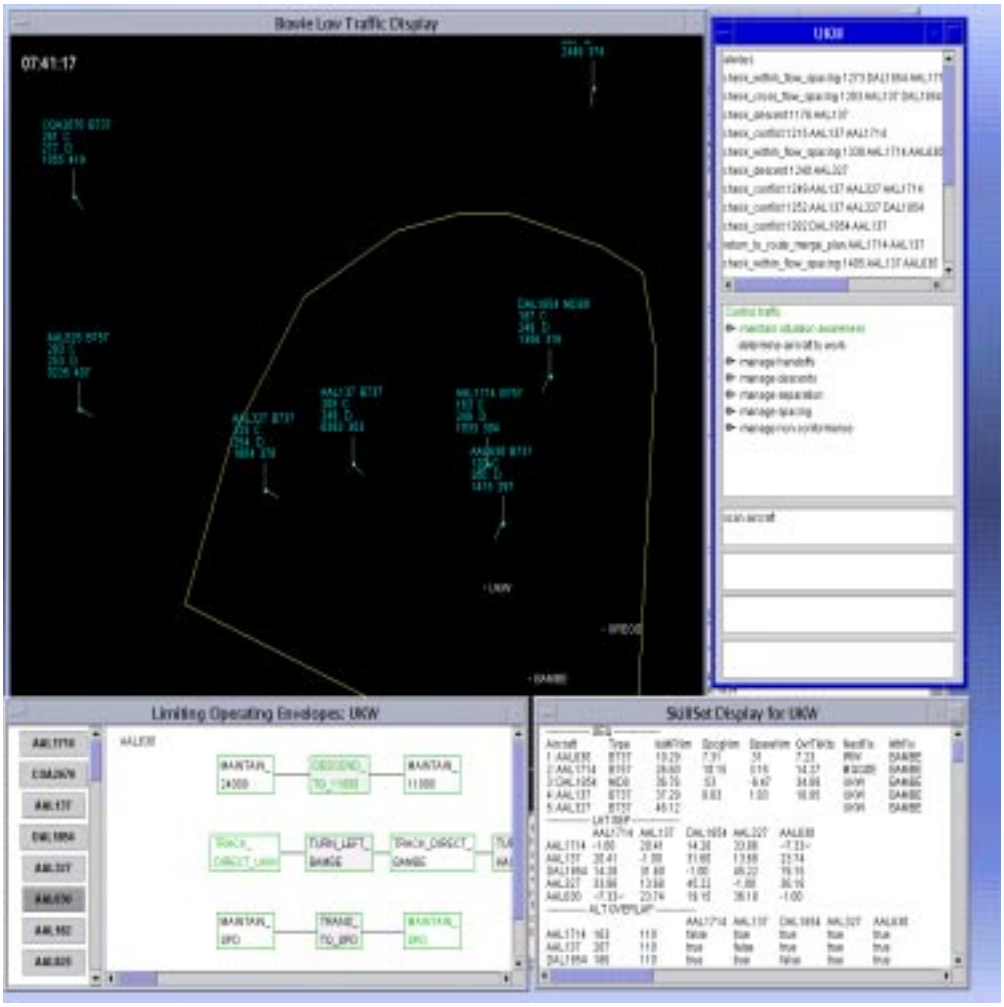


Figure 17. Screen snapshot of the UKW agent.

scenario. Figure 19 shows that the total time for aircraft to arrive in each condition was comparable, but the agents were able to compress the traffic in the 'A' group of scenarios.

For the performance assessment, each of the nine scenarios was run twice. First each scenario was run in a 'descent only' control condition with agents issuing only descent clearances to the sector exit altitude. Each scenario was then run again, with the agents controlling traffic to the fullest extent (i.e., issuing speed, heading/route, and positive altitude separation clearances). In the descent only condition, aircraft always remain on their filed flight plans but, as the results show, are still subject to some nuances of agent behavior. Traces of the traffic flows for each test run appear in Appendix B; the traces for the descent only condition help characterize the traffic flow in each scenario.

In general, the performance assessment reveals that the agents generally do a good job handling spacing problems in the high altitude sectors (SPS and ADM). The agents are less adept at handling merge problems, and even less so at handling difficult multiple merges at the meter fix. Nonetheless, in no case did the agents fully controlling traffic produce more separation violations than in the uncontrolled ('descent only') condition. The results also show that traffic with a particularly large number of simultaneous conflicts (and the relatively large numbers of plans that may need executing) can leave the agent little time to perform other, lower priority tasks. For example, the results show that when too many aircraft are merging at the meter fix, the UKW agent has trouble managing descents. The agent may descend aircraft to temporary altitudes, but may fail to issue lower clearances in time for aircraft to reach the required sector exit altitude.

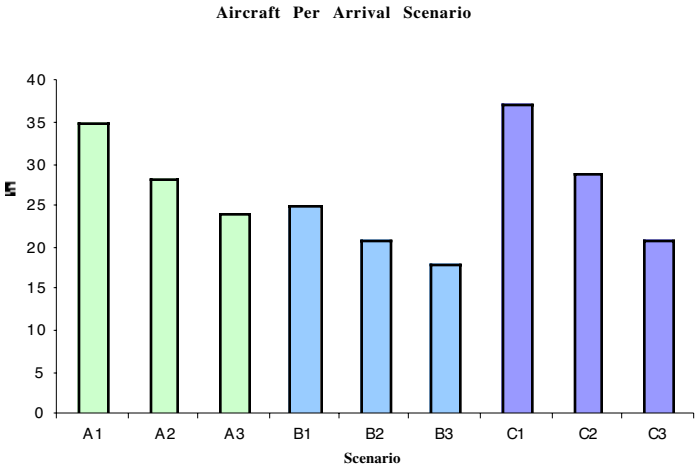


Figure 18. Number of arrival aircraft in each test scenario.

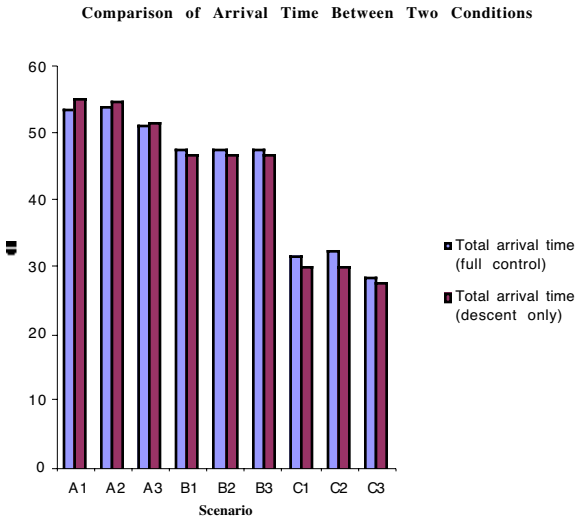


Figure 19. Time for all aircraft to cross the meter fix by scenario and condition.

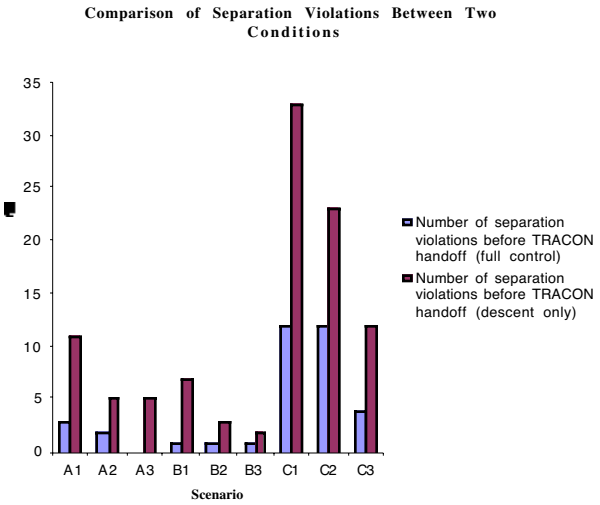


Figure 20. Number of separation violations in each scenario for each of the two conditions.

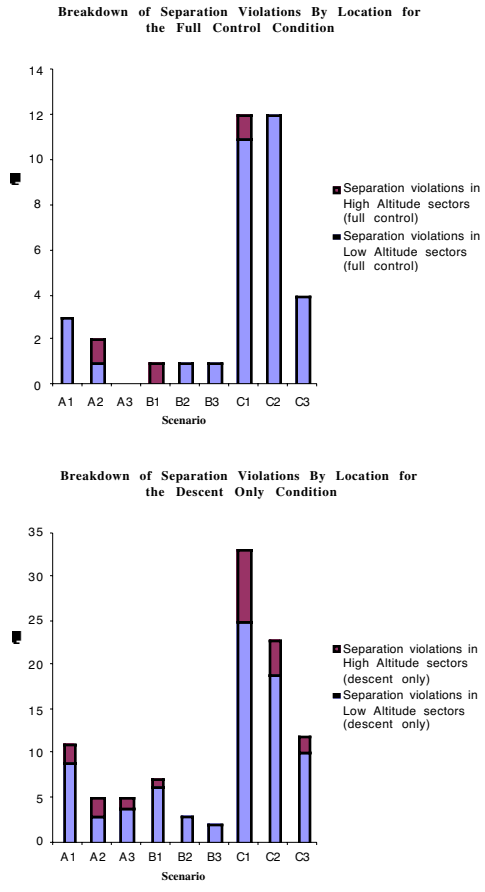


Figure 21. Location of separation violations under each condition.

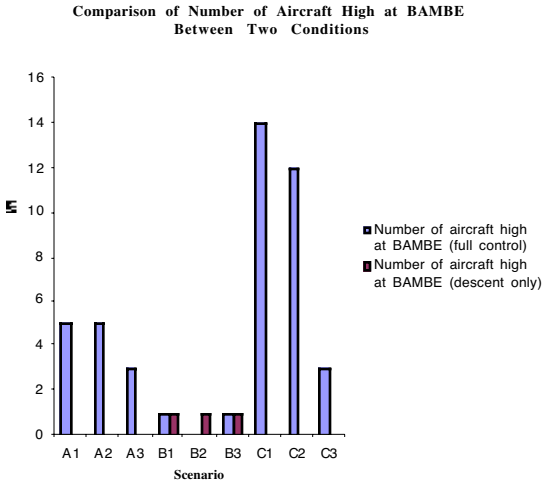


Figure 22. Number of aircraft not at the desired crossing altitude at BAMBE under each condition.

The results first focus on the number of separation violations that result under each of the two conditions. Data produced by the aircraft simulation were analyzed using a computer-based analysis program. The analysis program measures separation violations stringently. A pair of aircraft with less than the required vertical separation registers as a separation violation when lateral separation falls even a infinitesimal amount below five nautical miles (i.e., 4.98 nm counts as a violation). Figure 20 shows that, for every scenario, the full-control condition results in fewer separation violations than occur in the descent-only condition. For the A and B scenario sets, the agents created relatively few separation violations, while for the more dense traffic in the C scenario set, the agents created a considerable number of separation violations.

Figure 21 shows the agents in the full control condition created a single separation violation that involved at least one aircraft that was still in a high altitude sector in three of the nine scenarios. All three of these violations were of the 'barely below five nautical miles' variety. The remainder of the violations all

occurred in the UKW Low Altitude sector, where the aircraft must merge before crossing the meter fix. Figure 21 also shows that, for the descent only condition, a larger number of separation violations occurred in the high altitude sectors. In Appendix B, lines connect points where spacing minimums were violated (while not always clearly visible in the charts, these lines sometimes help visualize where some violations occurred).

Finally, Figure 22 depicts the problem noted earlier, in which agents can become overloaded handling conflicts. The UKW agent's performance, in particular, suffers from this problem. Figure 22 shows that aircraft failed to achieve the altitude necessary for crossing the BAMBE meter fix more frequently in the full control condition. This problem rarely occurred in the descent only condition.

Conclusions and Further Research

Overall, the agents perform well in the High Altitude sectors. However, the UKW agent can become overloaded easily; too many merge conflicts and attendant plans cause the UKW agent to ignore aircraft that need lower altitude clearances. This is likely due largely to activities taking one second to execute, as is currently required by the agent architecture. The agents also merge traffic reasonably well, but priorities should be more flexible, so that aircraft that require a lower altitude immediately are sure to receive the clearance. In general, despite these difficulties, the agents show considerable promise as tools for both understanding how air traffic controllers operate, and for supplanting expensive, variable human air traffic controllers in future ATM concept studies.

There is considerable room for refinement in several areas. First the hub-based architecture needs to support asynchronous processing, such that the agents can execute activities as time permits (currently, they wait until next 'tick' to perform the next activity). Second, the control rules and skills could benefit from some refinements. In particular, the vectoring skills, and the plans that result when they are applied, may operate more effectively if based on specific attributes of the merge that is taking place.

The notion of improving the flexibility of the agents is, indeed, overarching. Role bindings could be performed dynamically depending on the situation (e.g., spacing versus merging). Plans could benefit from more flexible, dynamic adaptation. Tolerances used by perceptual skills could be set dynamically depending on the situation. And timing values used in beliefs that dictate when a problem can be re-addressed could similarly be situation-specific. All of these areas should be refined with the aid of actual air traffic controller input, to the extent possible. One approach would be to take data on actual controller behavior for representative classes of control problems, as was done for particular conflict classes in the RAMS research (Mondoloni, 1998).

This is but one of several areas of further research. Other areas include extending the CATS agent architecture, with its mechanisms for manipulating an agent's beliefs about the task context and control situation, to enable the agents to make realistic errors. The error-making capability could be applied for safety assessment of ATM concepts. A second area of research is integrating the ATC agent capabilities into a compact module that would enable it to be integrated into a flexible controller station, in which agent control could be toggled on and off. Part of this research will entail enabling the agents to access advanced ATC automation tools that play a central role in new ATM concepts (e.g., Callantine, et al., 2001). A variety of new skills and control rules will

require development to support tool usage by the agents. For example, the agents will require access to information presented as a timeline of arrival aircraft, and functionality designed to predict precise speeds necessary to meet scheduled meter fix crossing times. Thus, this report has presented initial work toward these aims.

References

Battiste, V., Johnson, W., Kopardekar, P., Lozito, S., Mogford, R., Palmer, E., Prevôt, T., Sacco, N., Sheldon, S., and Smith, N. (2002). *Distributed air/ground traffic management – technology and concept demonstration report*. AIAA-2002-5825, Reston, VA: American Institute of Aeronautics and Astronautics.

Callantine, T. (2001). Agents for analysis and design of complex systems. *Proceedings of the 2001 International Conference on Systems, Man, and Cybernetics*, October, 567-573.

Callantine, T. (2002). A representation of air traffic control clearance constraints for intelligent agents. *Proceedings of the 2002 IEEE International Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, October.

Callantine, T., Mitchell, C., and Palmer, E. (1999). *GT-CATS: Tracking operator activities in complex systems*. NASA Technical Memorandum 208788, Moffett Field, CA: NASA Ames Research Center.

Callantine, T. Prevôt, T., Smith, N., and Palmer, E. (2001). Simulation of CTAS/FMS air traffic management, Proceedings of the 4th USA/Europe Air Traffic Management Research and Development Seminar, Air-Ground Cooperation Track, Santa Fe, NM, December, not paginated.

Dowell, J. (1998). Ecological modelling of the air traffic management task. *Proceedings of the Ninth European Conference on Cognitive Ergonomics (ECCE-9),* Limerick, Ireland: University of Limerick, 109-116.

Harper, K., Guarino, S., White, A., Hanson, M., Bilimoria, K., and Mulfinger, D. (2002). *An agent-based approach to aircraft conflict resolution with constraints*. AIAA 2002-4552, Reston, VA: American Institute of Aeronautics and Astronautics.

Heimerman, K. (1998). Air traffic control modeling. In *Frontiers of Engineering: Reports on Leading Edge Engineering from the 1997 NAE Symposium on Frontiers of Engineering*, Washington, D.C.: National Academy Press, 41-50.

Hexmoor, H. and Heng, T. (2000). Air traffic control agents: Landing and collision avoidance. *Proceedings of the 2000 International Conference on Artificial Intelligence*, Berlin: Springer-Verlag.

Horswill, I. and Zubek, R. (1999). *Robot architectures for believable game agents*. In D. Dobson and K. Forbus (Eds.), AAAI Technical Report SS-99-02, Menlo Park, CA: AAAI Press, 55-59.

Kliegel, M., Martin, M., McDaniel, M., and Einstein, G. (2002). Complex prospective memory and executive control of working memory: A process model. *Psychologische Beiträge, 44*, 303-318.

Leiden, K. (2000). *Human performance modeling of en route controllers* (RTO-55 Final Report). Boulder, CO: Micro Analysis and Design, Inc.

Mondoloni, S. (1998). *Development of an enroute conflict resolution rulebase for the reorganized air traffic control*

mathematical simulator. NARIM-A09008-01, Washington, D.C.: CSSI, Inc.

Niessen, C. and Eyferth, K. (2001). A model of the air traffic controller´s situation awareness. *Safety Science, 37*, 187-202.

Niessen, C., Eyferth, K. and Bierwagen, T. (1999). Modelling cognitive processes of experienced air traffic controllers. *Ergonomics, 42*(11), 1507-1520.

Niessen, C., Leuchter, S., and Eyferth, K. (1998). A pyschological model of air traffic control and its implementation. In F. E. Ritter and R. M. Young (eds), *Proceedings of the Second European Conference on Cognitive Modelling*, Nottingham, U.K.: Nottingham University Press, 104-111.

Prevôt, T., Palmer, E., Smith, N., and Callantine, T. (2002). *A multi-fidelity simulation environment for human-in-the-loop studies of distributed air ground traffic management*. AIAA 2002-4679. Reston, VA: American Institute of Aeronautics and Astronautics.

Raytheon (2002). *Air traffic management system development and integration: Distributed air-ground traffic management test plan*.

Schaefer, L., and Millner, D. (2001). Flight delay propagation analysis with the detailed policy assessment tool. *Proceedings of the 2001 IEEE Conference on Systems, Man, and Cybernetics*, Tucson, AZ, 1299-1303.

Appendix A

The following is the file specification for the CATS model used by the CATS ATC agents (Table A-1). Below it, this appendix provides some explanation on how it works.

Table A-1. CATS ATC agent model file specification.

```
! ATC AGT (Current-Day Operations -- put AC miles-in-trail) generic_0.4.1.proc
! This one uses MULTIPLE RETURN VALUES from act-beliefs
! Changes for beliefs-based situation representation

{ topLevel "Control traffic"
conditions "predicted" "always"

{ function "maintain situation awareness"
conditions "predicted" "always"

{ task "monitor traffic display"
conditions "predicted" "display needs scanning"
act_type "perceptual"
conditions "act-beliefs" ( visually to self "<sector aircraft>.set_value" )
conditions "rslt-beliefs-true" ( and
        ( cognitively to self "looked at traffic display" )
        ( cognitively from self "display needs scanning" ) )
}


{ task "scan aircraft"
conditions "predicted" "looked at traffic display"
act_type "cognitive"
conditions "act-beliefs"
        ( cognitively to self "<traffic>.assess" )
conditions "rslt-beliefs-false" (and
              ( cognitively from self "looked at traffic display" )
              ( cognitively to self "display needs scanning" ) )
conditions "rslt-beliefs-true" (and
              ( cognitively from self "looked at traffic display" )
              ( cognitively to self "have aircraft to work" ) )

}

} ! end "maintain situation awareness"


{ function "determine aircraft to work"
conditions "predicted" "have aircraft to work"
act_type "cognitive"
conditions "act-beliefs" ( cognitively to self "<aircraft to work>.evaluate" )
conditions "rslt-beliefs-hoa" (and
```

```
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "know which aircraft to accept" ) )
conditions "rslt-beliefs-hoi" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "know which aircraft to hand off" ) )
conditions "rslt-beliefs-td" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "know which aircraft to descend" ) )
conditions "rslt-beliefs-factors" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "factors identified" ) )
conditions "rslt-beliefs-spacing" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "spacing aircraft identified" ) )
conditions "rslt-beliefs-non-conf" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "know which aircraft isn't conforming" ) )
conditions "rslt-beliefs-exec-plan" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "know which aircraft to clear" ) )
conditions "rslt-beliefs-false" (and
              ( cognitively from self "have aircraft to work" )
              ( cognitively to self "display needs scanning" ) )
}


{ function "manage handoffs"
conditions "predicted" ( or
                     "know which aircraft to accept"
                     "know which aircraft have accepted"
                     "know which aircraft to hand off"
                     "know which aircraft is accepted" )

{ task "accept aircraft"
conditions "predicted" ( or
                     "know which aircraft to accept"
                     "know which aircraft have accepted" )

{ subtask "accept handoff"
conditions "predicted" "know which aircraft to accept"
act_type "manual"
conditions "act-beliefs"
       ( manually to "<previous controller>" "<incoming aircraft>.accept" )
conditions "rslt-beliefs-true" ( and
       ( cognitively to self "looked at traffic display" )
       ( manually to "<previous controller>" "<incoming aircraft>.accept" )
       ( cognitively from self "know which aircraft to accept" ) )
! this is in case there is <previous controller> is GHOST
conditions "rslt-beliefs-false" ( and
```

```
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to accept" ) )
}

! this has to wait for check-in from aircraft
{ subtask "roger check-in"
conditions "predicted" "know which aircraft have accepted"
act_type "verbal"
conditions "act-beliefs"
       ( verbally to "<incoming aircraft>" "<incoming aircraft>.roger" )
conditions "rslt-beliefs-true" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft have accepted" ) )
}

}

{ task "initiate handoff"
conditions "predicted" ( or
                     "know which aircraft to hand off"
                     "know which aircraft is accepted" )

{ subtask "inform other controller"
conditions "predicted" "know which aircraft to hand off"
act_type "manual"
conditions "act-beliefs"
       ( manually to "<next controller>" "<outgoing aircraft>.handoff" )
conditions "rslt-beliefs-true" ( and
       ( manually to "<next controller>" "<outgoing aircraft>.handoff" )
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to hand off" ) )
! this is in case there is <next controller> is GHOST
conditions "rslt-beliefs-false" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to hand off" ) )
}

! this has to wait for accept from other controller
{ subtask "issue frequency change"
conditions "predicted" "know which aircraft is accepted"
act_type "verbal"
conditions "act-beliefs"
       ( verbally to "<outgoing aircraft>" "<outgoing aircraft>.freq_change" )
conditions "rslt-beliefs-true" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft is accepted" ) )
}

}
```

```
}


{ function "manage descents"
conditions "predicted" "know which aircraft to descend"

{ task "issue descent clearance"
conditions "predicted" "know which aircraft to descend"
act_type "verbal"
conditions "act-beliefs"
       ( cognitively to self "<descent aircraft>.assign_alt" )
conditions "rslt-beliefs-true" ( and
       ( verbally to "<descent aircraft>" "<descent aircraft>.descend" )
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to descend" ) )
conditions "rslt-beliefs-false" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to descend" ) )
}

}


{ function "manage separation"
conditions "predicted" ( or
                     "factors identified"
                     "know which aircraft to clear" )

{ task "evaluate separation clearance options"
conditions "predicted" "factors identified"
act_type "cognitive"
conditions "act-beliefs"
       ( cognitively to self "<separation aircraft>.evaluate_and_set" )
conditions "rslt-beliefs-true" ( and
       ( cognitively to self "know which aircraft to clear" )
       ( cognitively from self "factors identified" ) )
conditions "rslt-beliefs-false" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "factors identified" ) )
}

{ task "issue separation clearance"
conditions "predicted" "know which aircraft to clear"
act_type "verbal"
conditions "act-beliefs"
       ( verbally to "<aircraft to clear>" "<aircraft to clear>.clear" )
conditions "rslt-beliefs-true" ( and
       ( verbally to "<aircraft to clear>" "<aircraft to clear>.issue" )
```

```
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to clear" ) )
}

}


{ function "manage spacing"
conditions "predicted" ( or "spacing aircraft identified"
                           "know which aircraft to space" )

{ task "evaluate spacing clearance options"
conditions "predicted" "spacing aircraft identified"
act_type "cognitive"
conditions "act-beliefs"
       ( cognitively to self "<spacing aircraft>.evaluate_and_set" )
conditions "rslt-beliefs-true" ( and
       ( cognitively to self "know which aircraft to space" )
       ( cognitively from self "spacing aircraft identified" ) )
conditions "rslt-beliefs-false" ( and
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "spacing aircraft identified" ) )
}

{ task "issue spacing clearance"
conditions "predicted" "know which aircraft to space"
act_type "verbal"
conditions "act-beliefs"
       ( verbally to "<aircraft to space>" "<aircraft to space>.clear" )
conditions "rslt-beliefs-true" ( and
       ( verbally to "<aircraft to space>" "<aircraft to space>.issue" )
       ( cognitively to self "looked at traffic display" )
       ( cognitively from self "know which aircraft to space" ) )
}

}



{ function "manage non-conformance"
conditions "predicted" "know which aircraft isn't conforming"

{ task "re-issue clearance"
conditions "predicted" "know which aircraft isn't conforming"
act_type "verbal"
conditions "act-beliefs"
       ( verbally to "<non-conforming aircraft>" "<non-conforming aircraft>.re-clear"
)
conditions "rslt-beliefs-true" ( and
```

```
        ( verbally to "<non-conforming aircraft>" "<non-conforming aircraft>.issue" )
        ( cognitively to self "looked at traffic display" )
        ( cognitively from self "know which aircraft isn't conforming" ) )
}

}

}
```

The file specification specifies information about the hierarchy of activities represented in the model. It also specifies the conditions under which they are 'predicted' (i.e., executed, in the case of CATS agents), what underlying skills and control rules an activity must access to execute, and what the agent's task context beliefs should reflect after they have executed an activity. The model specifies the activity hierarchy using 'curly brackets.' Activities are designated as "function," "task," etc., for no particular reason, except that past CATS implementations have used such designations (see Callantine, Mitchell, and Palmer, 1999, for discussion of the CATS model's roots in Operator Function Model methodology).

After an activity's name is information about the activity's type ('cognitive,' 'perceptual,' 'verbal,' or 'manual'), and a series of 'conditions' expressions. The first (order doesn't really matter, but an order is maintained for readability) is the conditions under which the activity is 'predicted.' These conditions reference task context beliefs. The second is 'act-beliefs.' This is the key expression for accessing underlying knowledge about how various beliefs get transformed by the so-called 'Belief Transformer' module in Figure 2 of the main text. Inspection of the model finds examples such as "<sector aircraft>.set_value", "<aircraft to work>.evaluate", "<separation aircraft>.evaluate_and_set", etc. These identifiers cue methods in the Belief Transformer module to perform the indicated manipulations by accessing the agent's skill library and control rules. The belief transformer installs the required beliefs in the agent's belief set, and returns a value that tells the agent how to adjust its task context beliefs.

The best example of this is what happens when the Belief Transformer is sent the cue "<aircraft to work>.evaluate" when an agent executes the activity 'determine aircraft to work.' Based on the priorities described in the main text (Figure 5), the Belief Transformer finds, first, whether there are any beliefs in the agent's belief set that say an aircraft with an executable plan exists ('plan_exec [aircraft]'). If so, the Belief transformer returns a PLAN_EXEC value, which in turn tells the agent to adjust its task context according to the 'conditions "rslt-beliefs-exec-plan"' clause in the model, and so on, for all the other outcomes. In this example, the task context information 'know which aircraft to clear' gets added to the agent's belief set. An interesting twist is, in evaluating plans to execute, the control rules actually set the value of the aircraft to clear, and the clearance value specified by the plan, so that when the activity 'issue separation clearance' fires (because its 'predicted' conditions are 'know which aircraft to clear'), and sends the cue "<aircraft to clear>.clear"' to the Belief Transformer, the Belief Transformer 'knows' which aircraft and what clearance is intended.

While a bit confusing, this scheme allows knowledge that resides in code to operate in concert with knowledge provided by the CATS model file specification. Earlier CATS agent applications (Callantine, 2001) use a similar scheme. Activity tracking applications (Callantine, Mitchell, and Palmer, 1999) are able to encapsulate all the required knowledge except rules for activating 'context specifiers' in the model file specification.

Appendix B

This appendix shows traffic flows for each of the scenarios in each of the two conditions. The traffic traces are shown in pairs, with the full control condition on the top, and the descent only condition on the bottom.

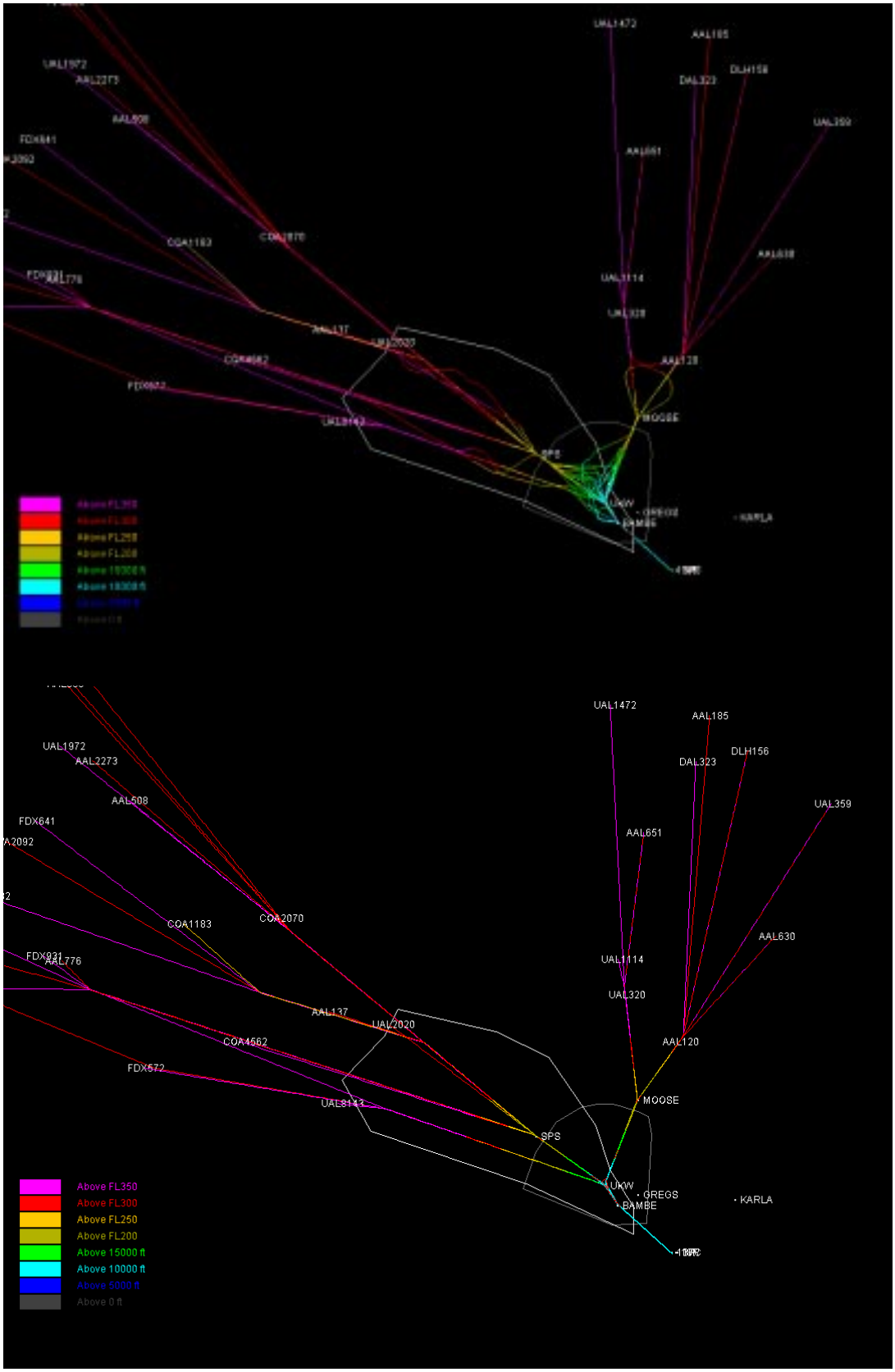


Figure B-1. Flows for scenario A-1.

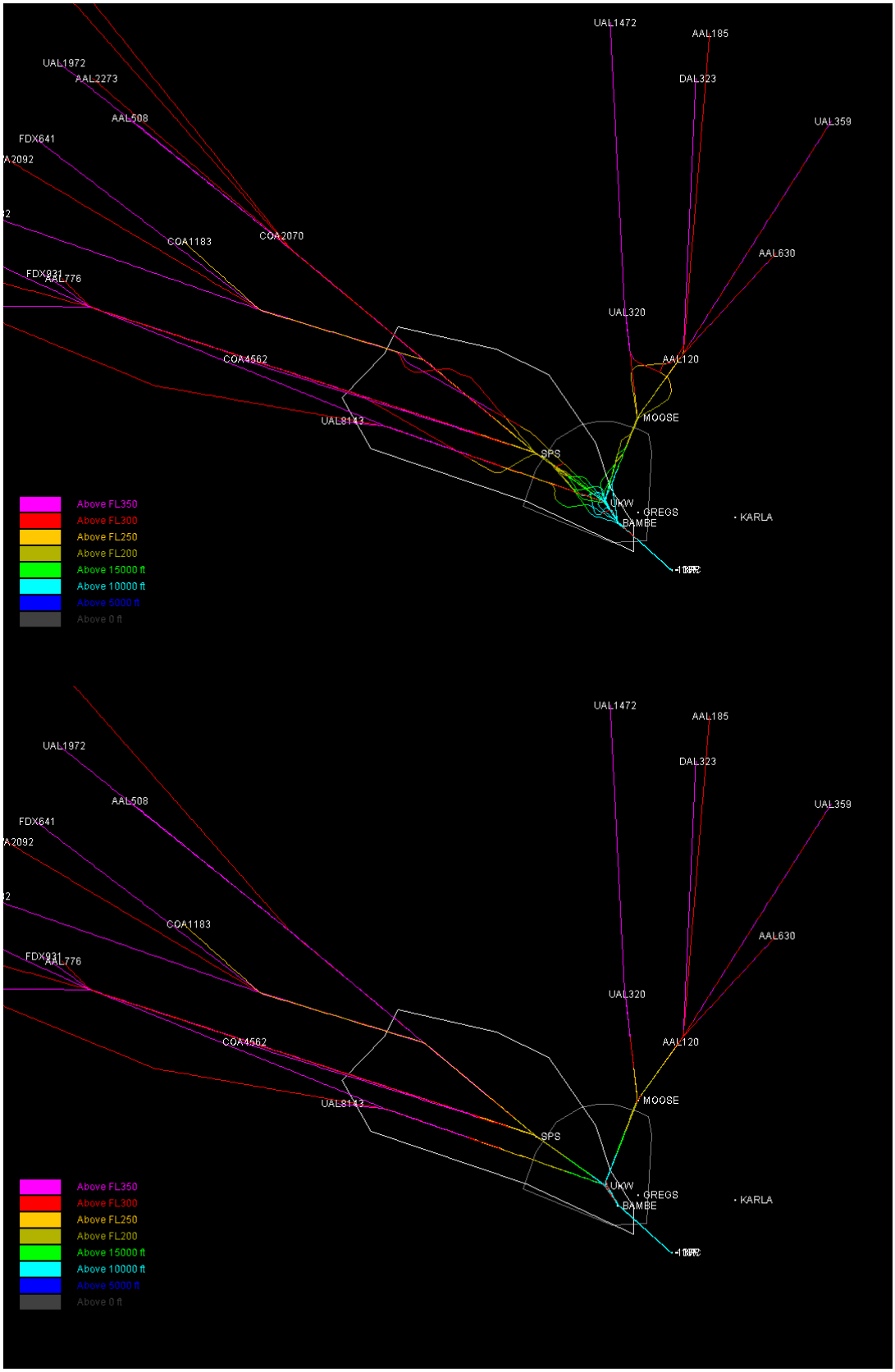


Figure B-2. Flows for scenario A-2.

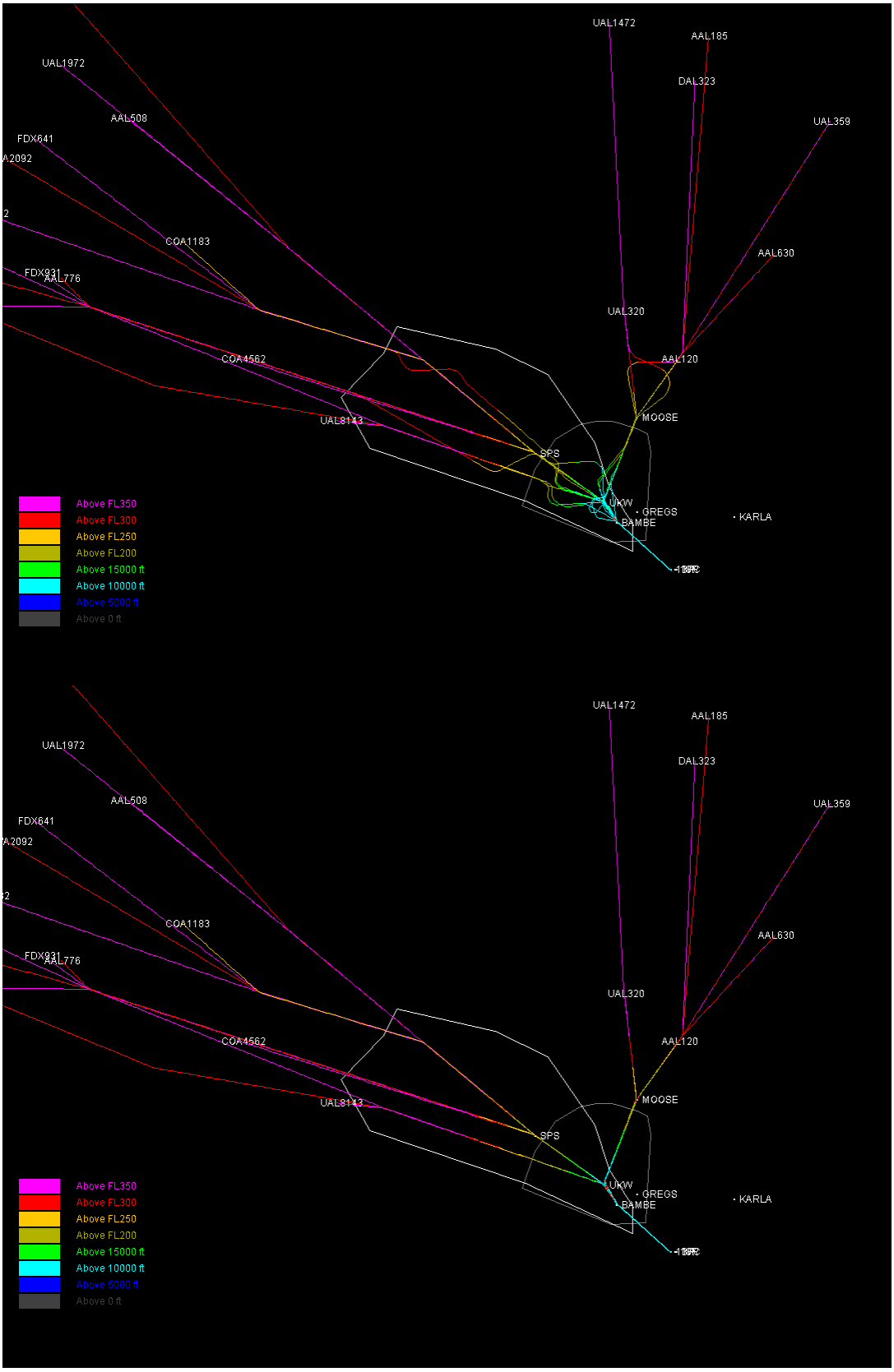


Figure B-1. Flows for scenario A-3.

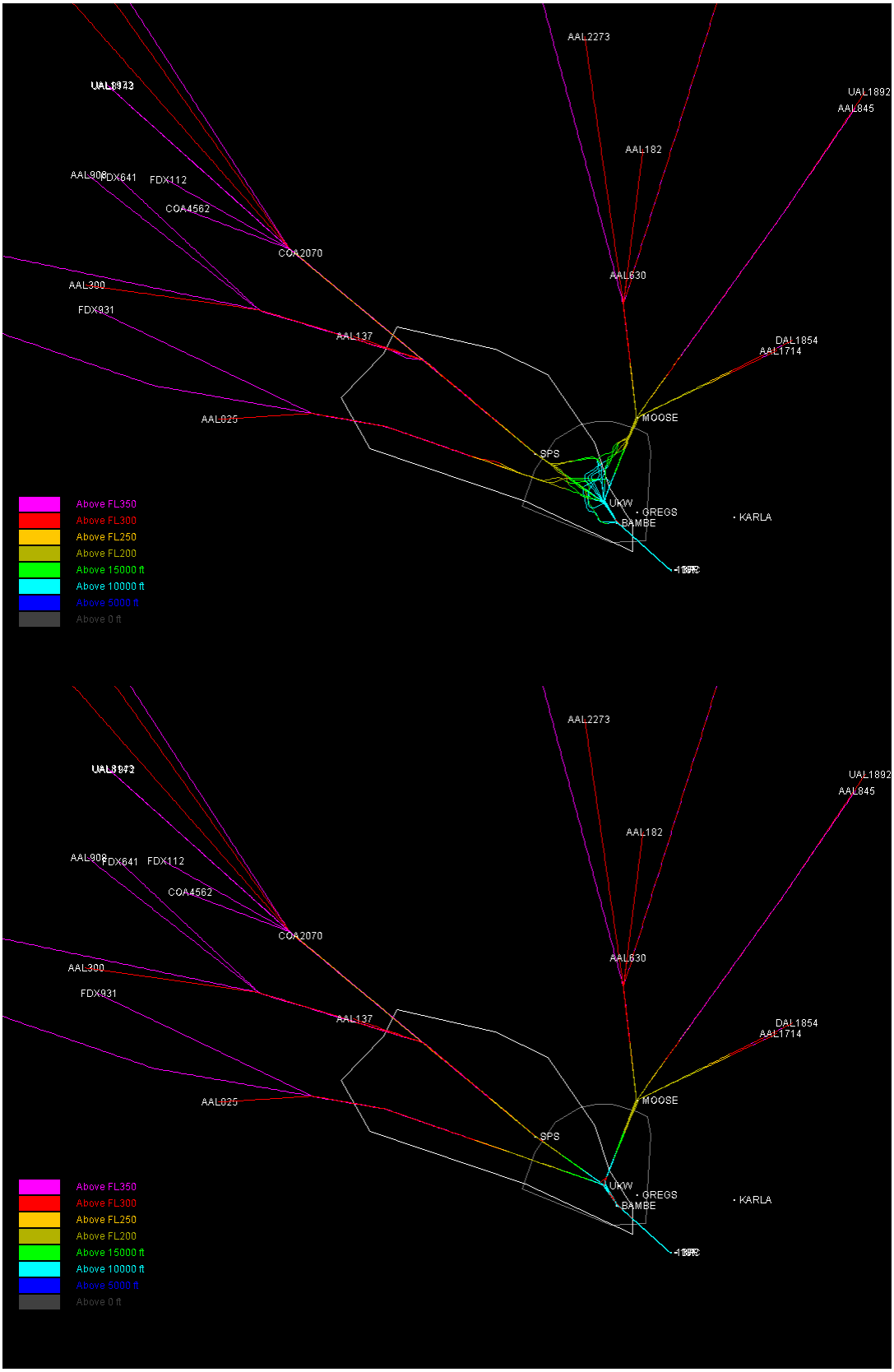


Figure B-4. Flows for scenario B-1.

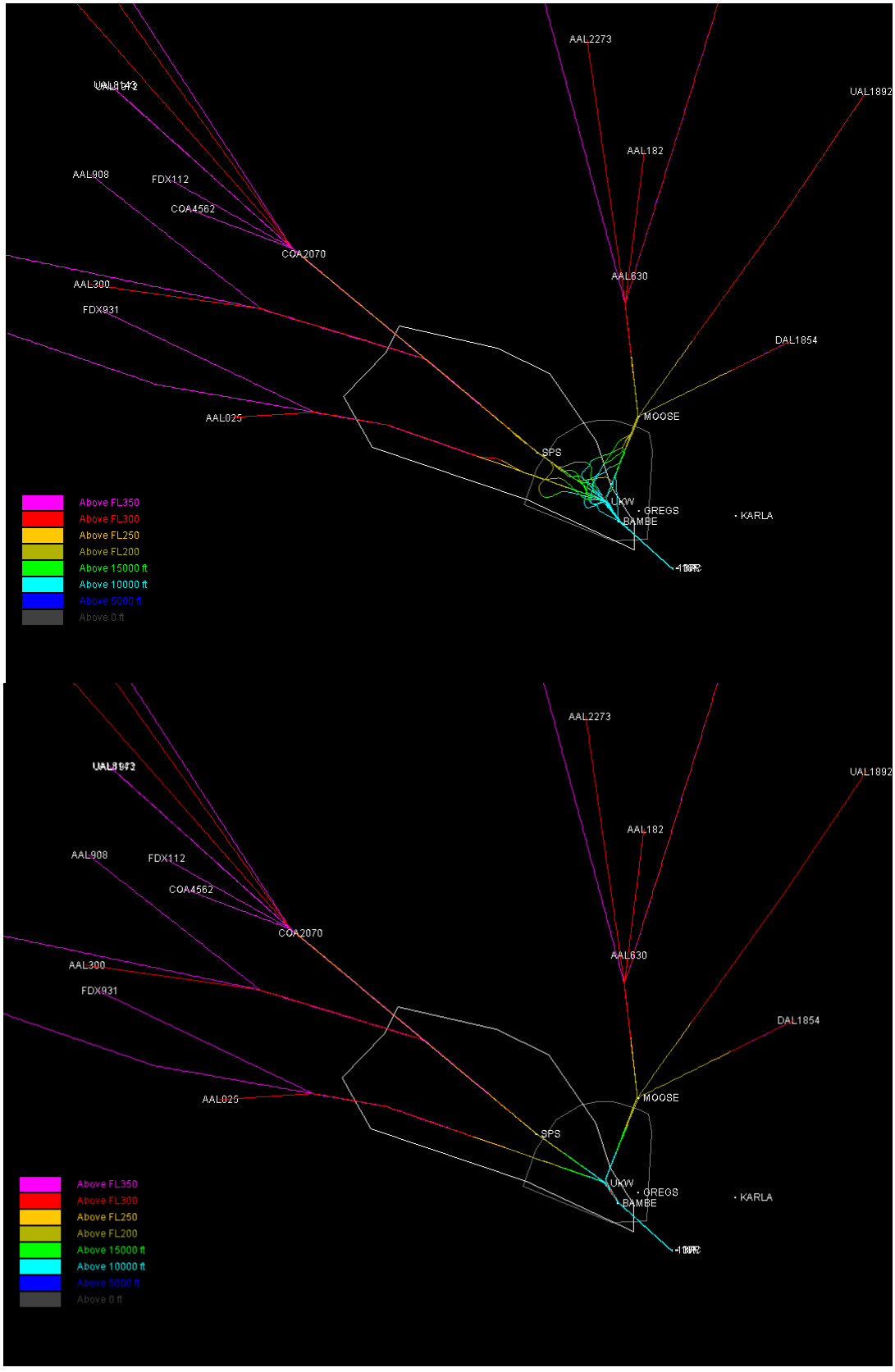


Figure B-5. Flows for scenario B-2.

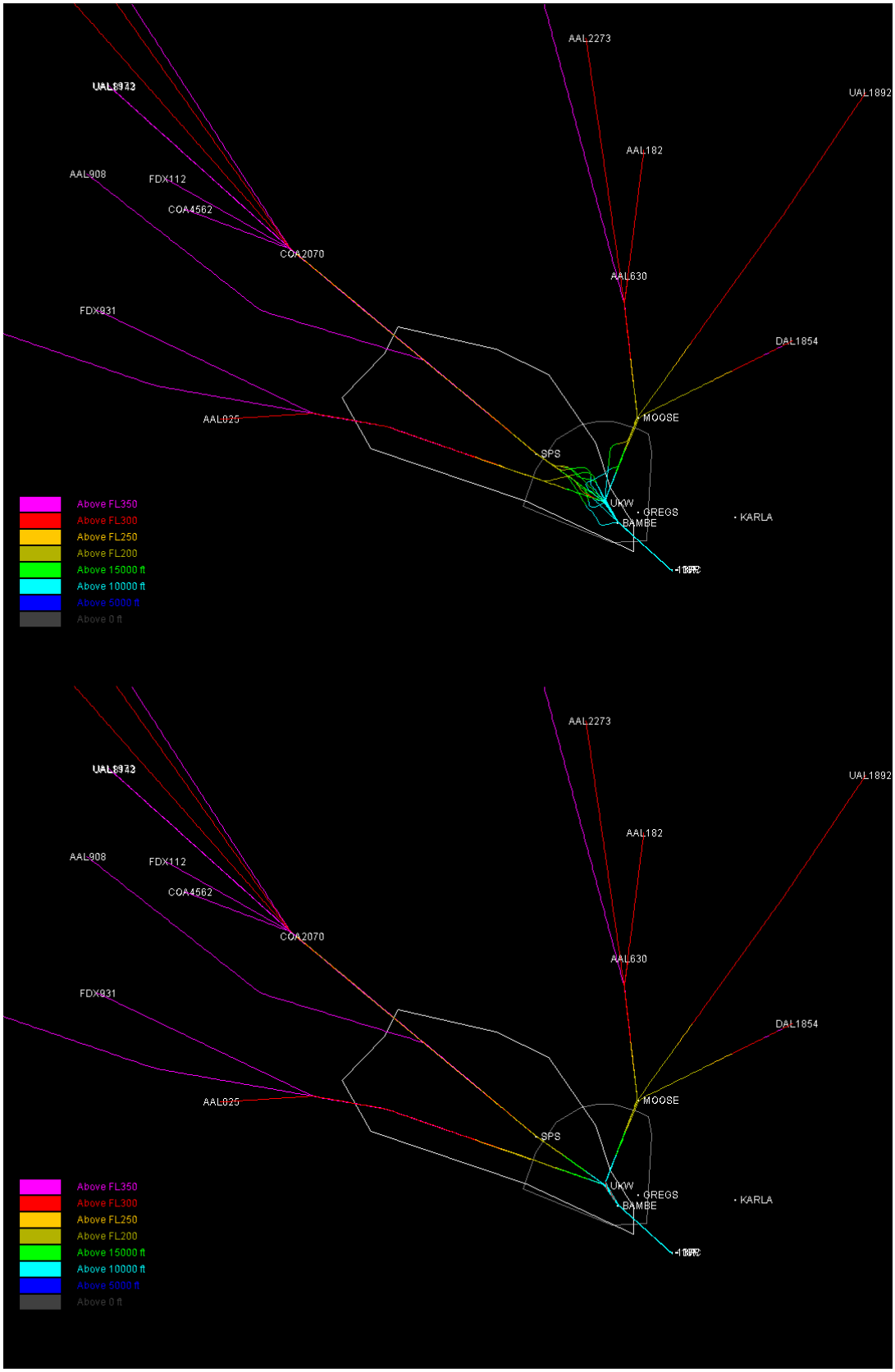


Figure B-6. Flows for scenario B-3.

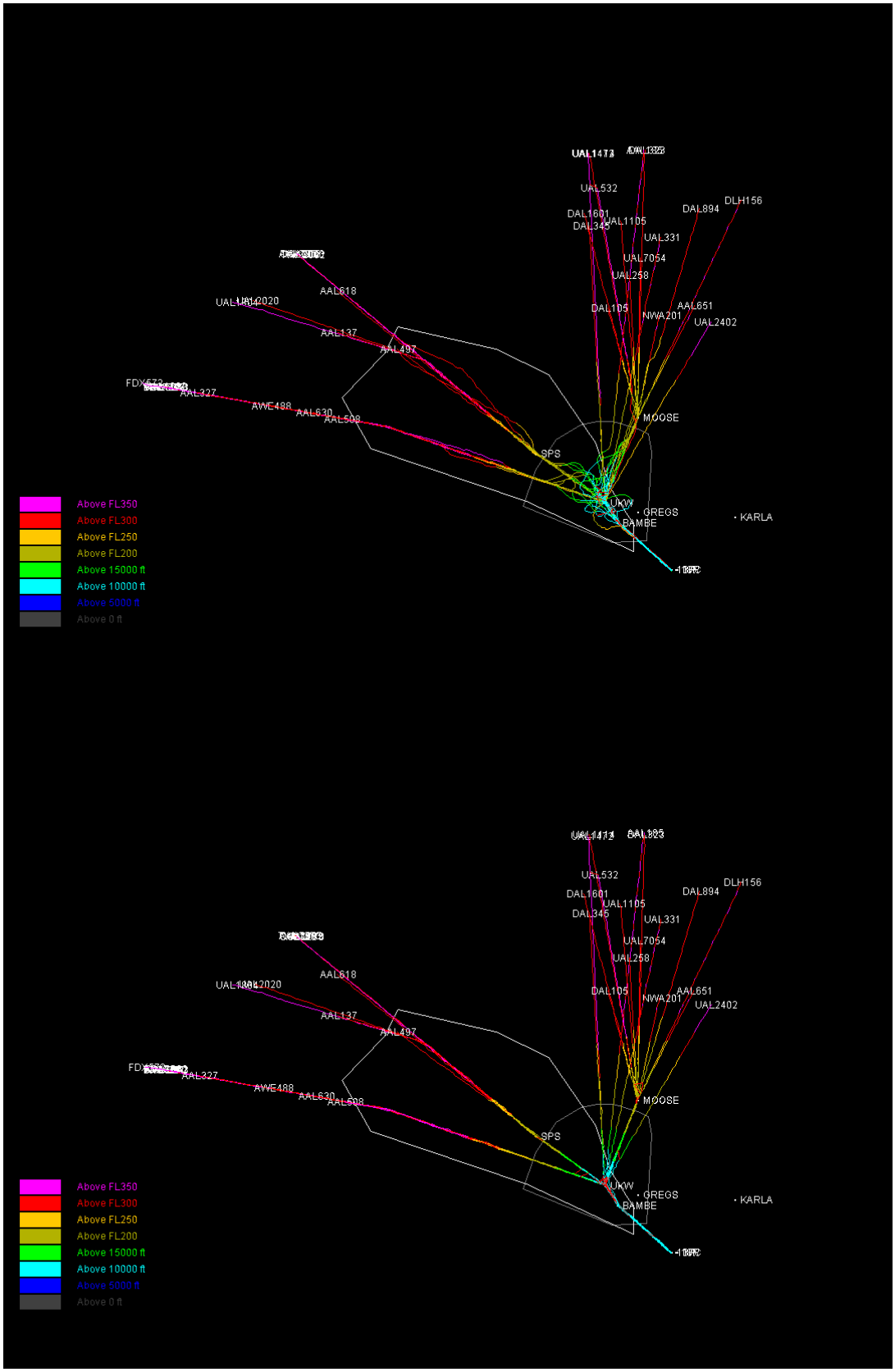


Figure B-7. Flows for scenario C-1.

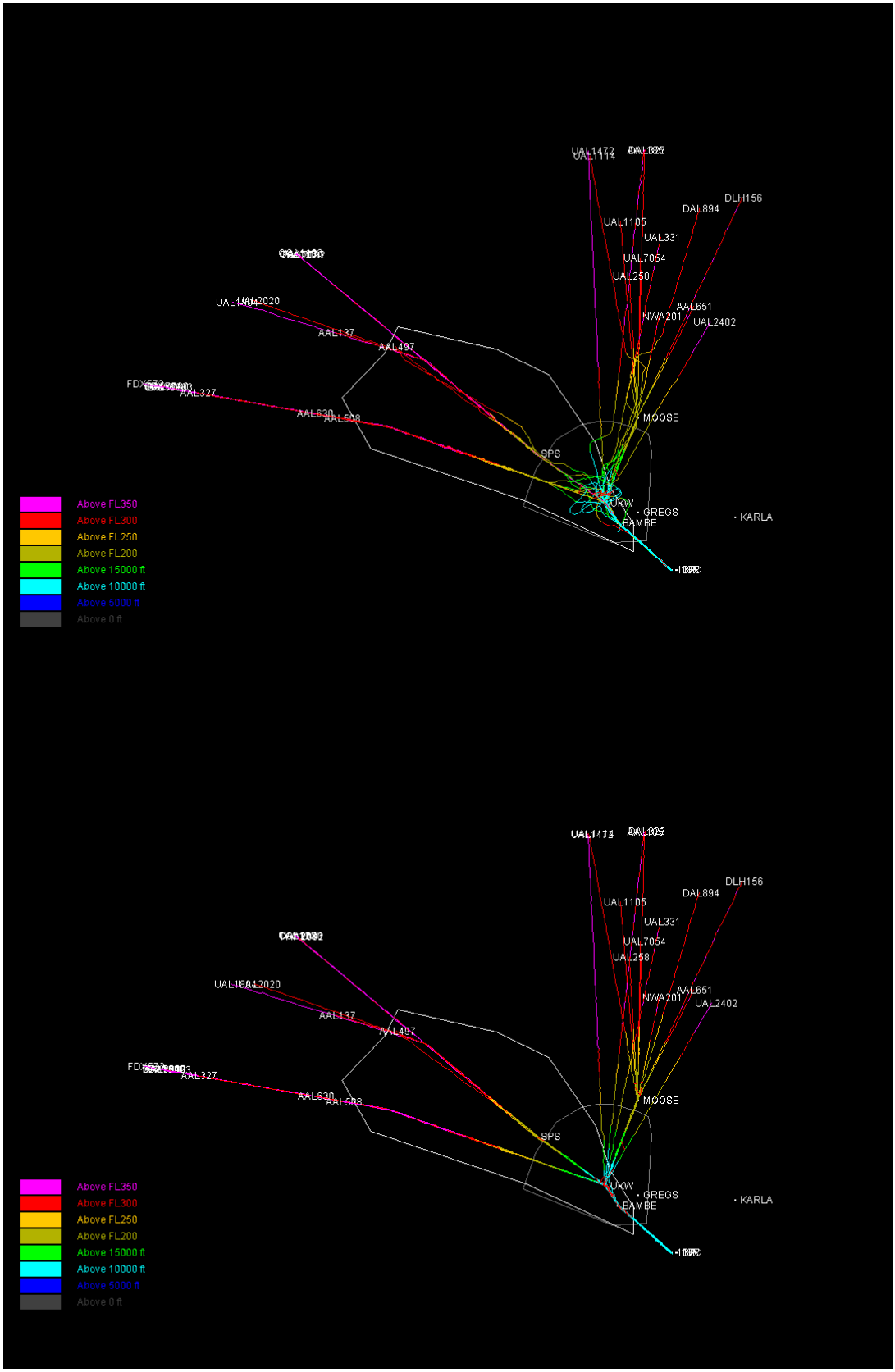


Figure B-8. Flows for scenario C-2.

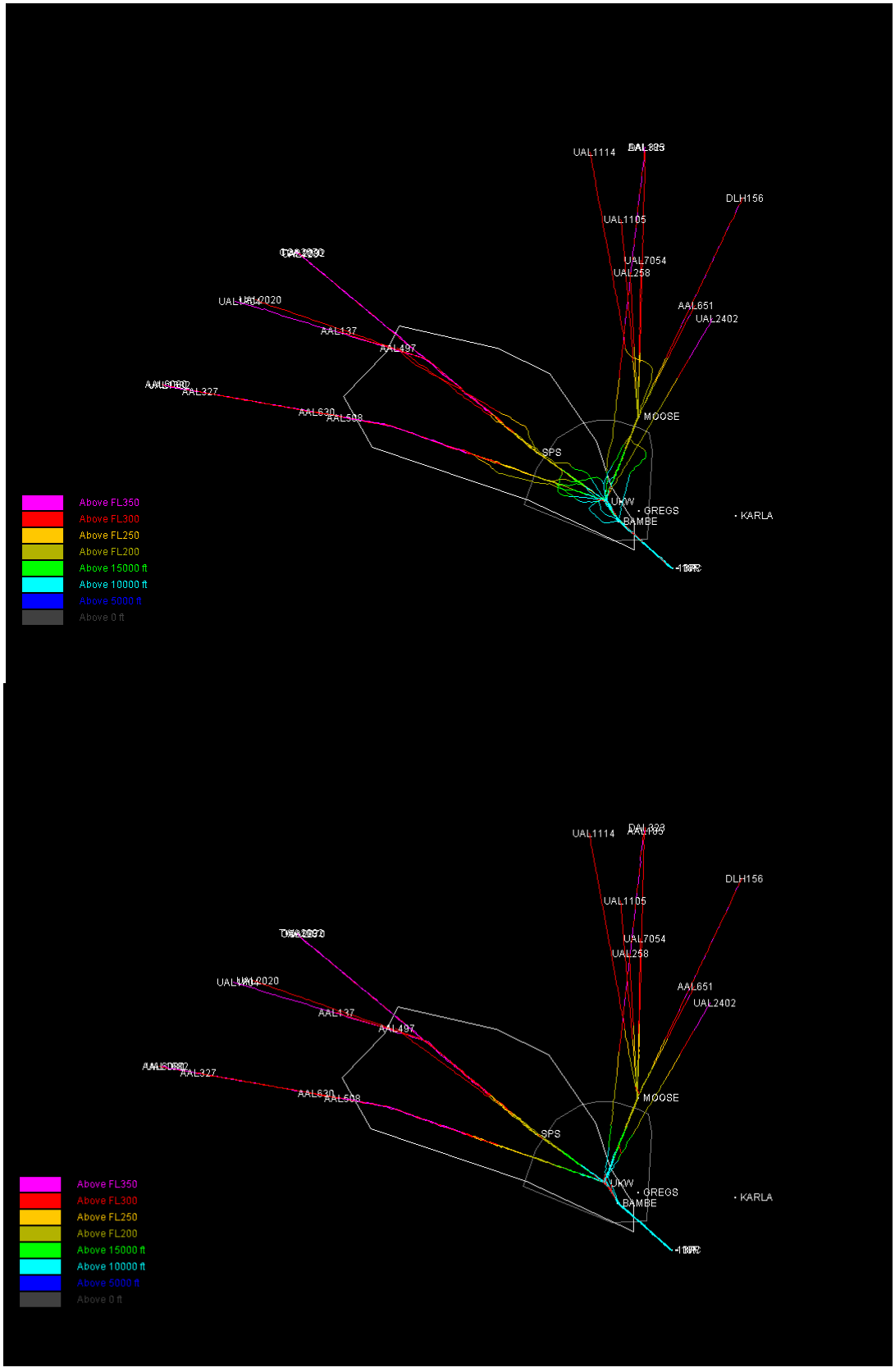


Figure B-9. Flows for scenario C-3.