FTL REPORT R85-1

RULE-BASED IMPLEMENTATION OF AUTOMATION AIDS FOR ATC CONTROLLERS

Antonio L. Elias

March 1985

MIT

DEPARTMENT OF AERONAUTICS & ASTRONAUTICS

FLIGHT TRANSPORTATION LABORATORY
Cambridge, Mass. 02139

FLIGHT TRANSPORTATION LABORATORY REPORT R85-1

RULE-BASED IMPLEMENTATION OF AUTOMATION AIDS FOR ATC CONTROLLERS

Antonio L. Elias

March 1985

RULE-BASED IMPLEMENTATION OF AUTOMATION AIDS FOR ATC CONTROLLERS

Antonio L. Elias

Flight Transportation Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

28 March 1985

Abstract

Development of Air Traffic Controller automation aids is frequently hampered by the mathematical nature of the algorithms they are based upon. These limitations are: lack of adaptability to local conditions; high development, testing, and modification costs; and low end-user confidence on the algorithm's behavior. Research in Artificial Intelligence has produced systems whose logic is implemented by means of rules which may be defined by the end-user without explicit programming. Such rule-based systems may provide a flexible, low-cost alternative to mathematical algorithms. Since the end-user can exercise significant control over the behavior of such logic, automation aids built using these methods can be tailored to the user's environment, preferences, and experience more readily than if built around a classical mathematical algorithm. While this is theoretically possible, present rule-based systems technology is insufficient to allow practical implementation of an ATC aid today. An experimental new rule-based core system has been developed which overcomes some of these obstacles, but a number of problems, including that of poor hardware performance, remain outstanding as topic for continuing research.

1. BACKGROUND

In January of 1984, the Flight Transportation Laboratory of M.I.T. began a program of research on the impact of recent developments in Computer Sciences to Air Traffic Control automation. Included in this research are new computation models, such as symbolic manipulation, object-oriented programming and logic programming, non-alphanumeric forms of input/output (e.g., icon-based), knowledge-based systems, and other topics related to the field of Artificial Intelligence. The object of this research is to identify potential functional improvements, software techniques, and the impact on hardware of these approaches as they may impact future ATC procurements. One of the application studied is the use of rule-based systems – a key component of the so-called knowledge-based systems – as an alternative to mathematical algorithms to implement ATC automation aids.

The first section of this paper presents the rationale for alternatives to the algorithmic approach of implementing ATC automation aids. The second part illustrates a specific application where a rule-based system may prove superior to algorithms. While this application has not yet been implemented or tested, the third part of the paper reports the actual progress of the work, including the difficulties found in attempting to implement these ideas, and the steps that are contemplated to overcome them.

2. PROBLEMS WITH TRADITIONAL ALGORITHMS

The dynamic nature of Air Traffic Control, and the procedures that both pilots and controllers have developed over the years combine to make the implementation of automation aids for air traffic controllers a very difficult proposition. A number of such aids have been developed, such as En-Route Metering, Conflict Alert, Minimum Safe Altitude Warning, Automated Metering and Spacing and, more recently, the Automated En Route ATC (*AERA*) system, which was originally conceived as an "Automatic Control" environment (1) but which, after significant scrutiny (2), developed into a collection of more loosely coupled automation aids capable of independent, incremental implementation (3). There is a consensus among all interested parties that aids such as these are highly desirable to reduce the frequency of operational errors, increase the throughput of critical elements of the national airways system (4), improve the operational economy and comfort of air transport (5), and, last but not least, increase the productivity of the ATC system (6).

With all these potential advantages, it is surprising that so few, if any, of these automation aid have actually gained operational acceptance. Part of this problem can be attributed to the inadequacy of the present ATC computational hardware, and to the exhorbitant cost of software modifications within the NAS system, factors recognized by the FAA and directly addressed in their system facilities improvement plans (7). This handicap, however, does not fully justify the lack of acceptance by the part of operating controllers of the few automated aids that *have* reached the floor of the ARTCC's, aids that were designed and implemented with sound design and engineering practices. We would like to propose as a possible contributing factor to this lack of success a fundamental conflict between the algorithmic nature of these automation aids, and the ad-hoc nature of most ATC situations. Algorithms, the very essence of traditional computer programming, are created by mapping physical entities into mathematical objects, such as scalar and vector variables; in turn, these objects are manipulated by mathematical means to yield values which we then offer to controllers as useful information upon which to base their control decisions. A typical example, and possibly the most successful algorithm in the entire NAS system is the radar tracking logic, where noisy measures of successive positions of aircraft are transformed into better estimates of current position, speed and direction by modeling both the dynamics of that aircraft[1] and the sources of measurement noise by means of mathematical entities.

In most instances, we find that the nature of ATC problems and the nature of mathematical algorithms conflict in the following ways:

1. Algorithms require simply-stated problems, whereas interesting ATC problems are ill-defined. For an algorithm to be functionally successful, the problem it models must be either simple (in structure), or capable of being simplified. This does not imply that it is simple to *solve*, but only simple to *state*, including that the problem be well-known and capable of being quantified. While estimating the

[1]In the simple α-β tracker, the assumed aircraft dynamics are simple straight, constant-velocity flight

most probable location of a radar target satisfies this requirement, we find that most interesting ATC automation problems are hard to simplify, or, when simplified, lose too much of their usefulness.

2. ATC problems tend to be specific to individual positions in individual sectors, which makes development of universal algorithms uneconomical. For an algorithm to be economically effective, it must be generic in its applicability; it is just not feasible to develop a different algorithm for every individual use of the aid, both in initial development cost, validation and verification, and further maintenance (or modification) costs. Similarly, development of an algorithm with sufficient special cases so that a single copy can cope with the variety of environments it may find in actual usage (perhaps by simple changes in algorithm parameters) would result in a large, cumbersome piece of software large portions of which will remain unexercised in any individual ATC position.

3. When used to make or propose decisions, algorithms behave in a way which may look counter-intuitive to a controller. Human thought processes, particularly when performing real-time control operations, rarely utilize the same abstract models as algorithms, designed in the quiet of an engineer's office; while mental extrapolation of target position's can be thought of as a mental vector operations, other mental processes involve pattern-matching, inference, intuition, and other elements which are normally not found in mathematical algorithms. While it may be perfectly feasible to design an algorithm that produces a correct control decision, it may be difficult for a human whose task is to use, monitor or validate that decision to understand *why* that result was produced. In an all-automatic ATC environment, such as the one ultimately sought for the high-altitude airspace in the AERA program, this may not be a significant problem; for a hybrid environment where automation aids are simply aids to the human who bears ultimate responsibility for control decisions, it is important that the information produced by the aid be perceived as "logical", or intuitively correct by the controller.

3. ADVANTAGES OF RULE-BASED LOGIC

While traditional approaches to computer software have relied on the availability and applicability of good algorithms (8), new developments in Computer Sciences, in particular symbolic manipulation and knowledge-based systems, are providing useful alternatives to traditional algorithms (9). While these developments have been brought up mainly by research in the field called Artificial Intelligence, we prefer to view them simply as new tools for developing functionality in computers, rather than as some kind of synthetic replacement for human skills and judgement.

"Rule-based" systems (RBS), sometimes called "Production Systems", implement logic by means of *condition-action* pairs. Such a system is composed of three parts (10):

1. A *rule base* of such condition-action pairs; (also referred to as the *ruleset*).

2. A *context*, composed of data structures which capture the state of the problem that is both referenced by, and modified by, the rules; and

3. A *controller*, (sometimes called *interpreter*), which embodies the logic by which the

rules influence, and are activated by, the context.

The software that embodies the controller and the data structures required to represent the context and the rule set is called a *core* or an *empty system* (that is, a rule system devoid of actual rules).

The traditional application of rule-based systems is in *expert systems*; in these systems, the rules implement the know-how and decision rules of very experienced (but not necessarily talented) individuals or *experts*. It is in the recognition that expertise is more often than not the result of experience, rather than talent, that makes successful expert system possible. Unfortunately, expert systems have received so much publicity in recent times that it is popularly believed that they can somehow behave in a talented manner. Although the original concept of a rule-based system can been traced as far back as 1943, practical applications of such systems are relatively recent, and had to wait for the development of efficient symbolic computation software (11) and hardware (12) systems. Other necessary developments were the refinement of the *process* of designing the form of context and rules to fit the problem at hand – a discipline known as *knowledge engineering* – and new methods of of controlling the triggering of individual rules – the essence of the *inference process* embedded inside the Rule-based system.

More recently, attempts have been made to develop general-purpose Rule-Based System "cores" capable of being adapted to a large variety of problems, thus reducing significantly the knowledge engineering effort required to develop an expert system. These systems attempt to be general enough to be applicable to a large variety of knowledge domains and problems, while at the same time being relatively simple to learn and efficient to use with conventional computer hardware.

After examining some of the most successful rule-based expert systems in operation, we believe that the use of this technology as an alternative to traditional algorithms to implement the logic of ATC automation aids may bring the following benefits:

1. Rules may be better at capturing qualitative aspects of ATC problems that may be hard to cast into conventional mathematical models. In particular, *rules about rules*, that is, rational self-modification of the logic, are much easier to implement in rule-based systems than in conventional algorithms.

2. The rule-based logic can be made more site-specific, user-specific, or case-specific than an algorithm, for the same expenditure of time and manpower, since the adding of a new rule or set of rules is a much less laborious process than adding the equivalent logic in a traditionally programmed algorithm. This makes automation software more easily adaptable to local circumstances which may change over short and longer terms.

3. The behavior of a rule-based system can be perceived as being closer to that of a human – in particular the individual or individuals who created the rules in the ruleset – than an algorithm.

4. The superior readability of rules as compared with coded algorithms makes verification and confidence-building easier and cheaper.

4. A SAMPLE POTENTIAL APPLICATION

A number of potential ATC automation aids are being considered as candidates for potential application of rule-based logic. Some of these aids address problems that were previously not considered good targets for automated aid, such as "Tower Chief", an expert system that advises an airport's local control teams on the effects of weather and other airports' traffic conditions on the choice of runway configurations and on the appropriate times to change configurations. Other aids address what are considered "classic" problems; let us look in detail at one of these to find out where and how rules could be applied to implement its logic.

Consider the problem of merging two converging streams of aircraft; although the *average* separation between aircraft on each stream should be sufficient to insure adecuate separation on the merged stream, the randomness of the incoming stream arrivals, and the lack of correlation between the streams are such that uncontrolled merging of the aircraft is likely to produce separation violations.

Figure 1 shows the basic geometry of the problem: two streams of aircraft, **A** and **B**, follow airways that merge at a nominal point, labelled **Bravo** in the figure. If no lateral maneuvering (e.g. corner-cutting) is allowed, the point Bravo is also the exit point of the merge operation: aircraft leave this point with the desired separation. If lateral separation is allowed, aircraft may not actually be on the merged path at Bravo; in this case, there is an ultimate merge point, labelled **Alpha** in Figure 1, after which the aircraft flows must have been completely merged. In either case, we call the point at which the merging process must be completed the *merge point*.

A simple way of visualizing the merging process is shown in Figure 2. In it, actual and desired aircraft positions are indicated in *time to the merge point* **(TTMP)**. Points to the right of the origin indicate positive TTMP, that is, aircraft which have not yet reached the merge point, while negative TTMP's indicate that the aircraft has already passed the merge point. The first horizontal line at the top of Figure 2 indicates a set of "desired" positions of the aircraft, in time-to-merge-point space, and is usually referred to as the "conveyor belt"; if each aircraft followed exactly the motion of one of the "buckets" in the conveyor belt, (with aircraft from both streams cooperating not to occupy the same bucket), then the merge problem would be solved, in that simple flight down the airways will produce a perfectly merged stream at point Bravo[2].

Unfortunately, the real position of the aircraft in "time-to-merge-point" space may look more like the second horizontal line of Figure 2: aircraft are not perfectly synchronized with the buckets and, as a matter of fact, may even be on top of each other in time-to-merge-point space. Under these circumstances, the TTMP of each aircraft may have to be shortened or lengthened to achieve a conflict-free merge at the merge point.

The limits to the maneuvering authority available to the controller to bring a particular aircraft aircraft inside the desired bucket can be summarized by a series of marks preceding and following the aircraft. Whereas the symbol indicating the "position" of the aircraft

[2]It is assumed in the figure that uniform spacing of the aircraft after the merge is desired; this may not be always the case, such as when interleaving arriving and departing traffic at an airport, or where there may be further merge points downstream of the current merge

measures the "nominal" time to the merge point of that aircraft[3], the "earliest possible" mark indicates the earliest time to merge that the particular aircraft may achieve using all available options (airspeed changes, changes in a descent profile, changes in the ground track). By comparison, "latest possible" marks must be qualified: if holding is allowed, there is no limit to how much an aircraft can be delayed, except, perhaps, fuel endurance. Thus, we should consider several "latest possible" times, each associated with a type of control action, with the understanding that their relative desirability decreases with increasing delay time[4].

Having thus abstracted the *Time-based Metered-merge* problem, we find opportunities for rule-based logic in the following parts of the problem:

1. Determination of the set of maneuvering options available to each aircraft to reduce and increase their TTMP. While in conventional algorithm this may have to be a fixed list, perhaps filtered by some site-dependent parameters, rules can be developed for individual sites to reflect such details as "If today is Saturday, the Restricted Area is not active, and this aircraft may be able to fly direct from Fox to Alpha, thus decreasing its TTMP by 10%".

2. Determination of a *compatible*, that is, conflict-free subset of the maneuvering options set. Although an algorithm similar to that used in AERA could be constructed to flag combinations of individual aircraft maneuvers that would result in a conflict, a rule-based may be significantly cheaper in computer time and more flexible in its operation (for example, it may apply different rules about the desired maneuvers by airline or aircraft type).

3. Determination of the nominal TTMP and the maneuvering limits. Although this seems to be an ideal application of traditional mathematical algorithms, there is still the question of "secondary effects" (such as uncertainty in the modeling of the wind profile) which impair an algorithm's capability to predict flight paths. This warrants further investigation of the performance of rule-based logic for flight path extrapolation.

4. Determination of bucket spacings in the conveyor belt: if the desired post-merge spacing is simple, as in the case of Figure 2, there may not be need for any logic; on the other hand, any variation from uniform spacing is likely to be site-specific, and thus an excellent candidate for rule-based implementation.

5. Assignment of individual aircraft from each stream to individual buckets in the conveyor belt. Again, this is a problem which could have a clear algorithmic solution: for example, assignments could be made to minimize the rms sum of the resulting deviations from the nominal TTMP, while not exceeding any aircraft's maneuvering limits. On the other hand, the resulting assignments may not be intuitive to the controller, or may no take into consideration exceptions that cannot

[3]Nominal in that a certain flight velocity profile is assumed, although it may include changes in ground-relative velocity brought about by changes in airspeed, altitude, or wind.

[4]At the scale used in Figure 2, max holding time may very well be off the figure, and thus is not portrayed

be included in the maneuver set logic of step 2.

6. Control of the aircraft to match and track their assigned bucket's time to merge. Here, rules could be used to suggest ATC commands that would maneuver each aircraft to its assigned TTMP bucket. Rules would be superior to continuous-logic guidance algorithms in that they could select the commands from a set of "standard procedures" used by controllers in that position to change the TTMP of aircraft. The resulting commands would look familiar both to monitoring controllers and to pilots.

Even reduced to this simple form, the problem is not static, but dynamic: in addition to the natural variations in wind and predicted aircraft performance that may invalidate and aircraft's earliest and latest times – which may necessitate a re-evaluation of the aircraft-bucket assignements – we also have the potential interaction between the speed of the buckets and the particular aircraft assigned to that bucket: if the nominal speed of the aircraft is not uniform, assigning a slower or faster aircraft to a particular bucket will change the shape of the conveyor belt, for a given desired outbound spacing[5].

5. PRACTICAL IMPLEMENTATION PROBLEMS

As part of its Advanced ATC Automation research effort, the Flight Transportation Laboratory is developing a real-time multiple-actor ATC simulation in Lisp. The overall configuration of the simulator is shown in Figure 3: two VAX 11/750 processors are used, running M.I.T.'s *NIL* (New Implementation of Lisp) system (13), a software package that simulates the Lisp environment of a Lisp Machine (12) on a Vax computers under the VMS operating system. A mixed interpreted-compiled system, we believe NIL to be the fastest, most complete implementation of Lisp outside of a Lisp machine[6]. High performance vector (Sanders G7) and raster (DEC VS-100) graphic output devices are accessible from the Lisp environment, as is an analog-to-digital I/O subsystem that allows a two-pilot moving-base twin jet simulator to fly in the simulated ATC environment alongside with the synthetic aircraft.

This Lisp-based environment is ideal to test and evaluate Rule-based systems. Initially, it was thought that an existing RBS core could be used to implement a sample RBS-logic aid by simple construction of an appropriate ruleset. Eight published RBS cores were examined (14), and sample rules were written for some of them, even only one of them (Emycin) was actually available at the ATC simulator facility. It was extremely difficult, and in some cases impossible, to cast ATC rules in the form required by these RBS cores, and expect them to be correctly triggered using the control logic available. Specifically, the following difficulties were encountered:

[5]In the case of non-uniform nominal speeds, there is also the additional complication of reversed positions in time to merge space and in geometric space: a slower aircraft may be physically *closer* to the merge point than an earlier faster aircraft.

[6]In spite of the high efficiency of the NIL implementation, performance of Lisp in conventional computers, such as the VAX series, is and will remain extremely poor.

1. Most existing RBS – and the expert systems they implement – operate on a single object with many attributes (for instance, an infection whose diagnosis is sought), or on multiple individual objects with similar structure (15). In contrast, most ATC rules end up referring to *classified* sets of objects, such as "an aircraft" and "an airway", which are instances of two classes of generic objects.

2. As a consequence, triggering of the rules must include automatic scanning over members of each object class mentioned: when a rule establishes a relationship between, say "aircraft1", "aircraft2", and "airport1", this rule can potentially affect any combination of two aircraft and one airport from the set of aircraft and the set of airports. No RBS core examined had this capability.

3. Useful ATC rules require the use of both boolean (logic) and quantitative (numerical) attributes of the objects. Some RBS cores offer this capability, while the logic of others intrinsically denies the use of quantitative attributes.

4. It seems impossible, at least at the present level of development, to eliminate completely numerical computations and small algorithms from the ruleset. Only a few RBS cores allow "programs" to be embedded in the rules.

5. There are two basic forms of controlling the activation of the rules in an RBS: in the *forward-chaining* method, known facts are used to trigger rules which establish more facts which, in turn, may trigger even more rules until either a desired conclusion is reached, or all possible true conclusions are established; in the *backwards-chaining* approach, a hypothetical conclusion is established true or false by traversing the ruleset tree in the reverse direction (*then-if*). Exhaustive forward-chaining seems to be the correct inference mechanism for the implementation of ATC logic.

6. Most RBS systems assume a "batch" type of operation; this does not mean that they cannot operate interactively, but rather that they assume that all the information about the context of the problem is available before the inference proces begins. It is impossible, for example, to incrementally add new information, as would be the case on a "live" ATC aid.

7. The notion of "time" is not well handled in most RBS core systems; although some allow for the context to be changed from the "outside", in addition to the changes produced by rule-firing[7], the mechanisms available for handling context changes and trigger rules based on changes in the context are not adequate for real-time systems. The mere concept of "past", for example, has not been adequately addressed: in a radar-based system, where an aircraft's reply may be missed during certain antenna sweep, what is the "latest" raw radar position of that aircraft? Is it the position measured during the last sweep (therefore "unknown"), or the position measured during the last antenna sweep in which a positive reply was received? Which meaning of "past" is desired depends in the intended use of the information, and thus there are at least two different notions of "past

[7]E.g. by implementing it as a series of *frames*.

information".

Although some of the RBS cores evaluated contained a number of desirable features, no single one combined all the functionality listed above, an deemed necessary for ATC applications. Consequently, a new RBS core, called RS-1, has been developed, which embodies all of the features identified above. As such, it represents a significant advance in the state of the art for Rule-Based Systems, and is a necessary first step towards applying RBS technology in ATC automation.

At the present time, only a small set of rules have been tested under RS-1, mostly to verify the operation of the core. Rulesets for functional ATC automation aids have yet to be developed. The major concern at the present time is execution speed: when fully compiled, RS-1 executes a small ruleset (5 rules) in a small context (3 aircraft) at an effective rate of 1 complete inference pass a second on a VAX 11/750 using NIL. This is amply sufficient to execute in real-time in synchronization with an ARTS-type radar antenna sweep, which is assumed to be the main source of context data. On the other hand, it is doubtful that a practical problem with some 35 to 50 rules and 10 to 15 aircraft in its context could execute in real time on a VAX/750 already burdened with the simulation of the ATC environment. We expect the first simulation test to run slower than real time until high-performance (e.g. Lisp machines) are available at the facility. Use of "large" rules sets (i.e. hundreds of rules) in real time is expected to require symbolic computation power exceeding that of current Lisp machines.

6. CONCLUSIONS

Rule-based systems seem to offer economic, functional and human-factors related advantages over conventional mathematical algorithms for the implementation of ATC automation logic. These advantages are evident at the level of individual automation aid implementation, rather than at some higher abstract "expert system" level. The principal advantage seems to be the capability of a rule-based system to adapt to the individual requirements of a sector or a controller, and to more complex problems than economically feasible with mathematical algorithms. Present Artificial Intelligence technology is not mature enough to realize these benefits, since A.I. research has never addressed the ATC field. As a consequence, existing A.I. tools are not appropriate to implement even a simple ATC automation aid, and new tools, such as the RS-1 Rule-Based System core, will have to be developed. It also seems inevitable that high-speed dedicated symbolic manipulation hardware will be required to implement a useable rule-based ATC automation aid.

7. ACKNOWLEDGEMENT

The work reported in this paper was performed under sponsorship of the Federal Aviation Administration as part of its Advanced Computer Sciences program. The contract was administered by the Transportation Systems Center, Cambridge, Massachusetts. The author wishes to recognize the support and technical suggestions of Messrs. Ron Rutledge and Richard Wright of TSC, and Mr. Paul Newman of the FAA Advanced Automation Office.

References

[1] Rucker, Richard A. *Automated En Route ATC (AERA): Operational Concepts, Package 1 Description, and Issues.* Technical Report MTR-79W00167, The MITRE Corporation, McLean, Va., May, 1979.

[2] Wesson, R., Solomon, K., Steeb, R., Thorndyke, P. and Wescourt, K. *Scenarios for Evolution of Air Traffic Control.* Technical Report R-2698-FAA, The Rand Corporation, Santa Monica, Ca., November, 1981.

[3] Zellweger, Andres G. *Advanced Automation System (AAS) Transition Strategy.* Technical Report DOT/FAA/AP-83/1, Advanced Automation Program Office, F.A.A., Washington, D.C., April, 1983.

[4] Dear, Roger D. *The Dynamic Scheduling of Aircraft in the Near Terminal Area.* Technical Report FTL-R76-9, Flight Transportation Laboratory, M.I.T., September, 1976.

[5] Rucker, Richard A. *Potential Fuel Savings of Specific ATC System Improvements.* Technical Report MIR-81W275, The MITRE Corporation, McLean, Va., February, 1982.

[6] Lewis, David L. *Improving the Air Traffic Control System: An Assessment of the National Airspace System Plan.* Technical Report, Congressional Budget Office, Washington, D.C., August, 1983.

[7] U. S. Department of Transportation, Federal Aviation Administration. National Airspace System Plan: Facilities, Equipment and Associated Development.
April, 1983

[8] Knuth, D. *The Art of Computer Programming.* Addison-Wesley, Reading, Ma, 1969.

[9] Winston, Patrick Henry. *Artificial Intelligence.* Addison-Wesley, Reading, Ma, 1977.

[10] Barr, Avron., and Feigenbaum, Edward. A. (editors). *Handbook of Artificial Intelligence, Vol. 1.* William Kaufman, Los Altos, Ca., 1981.

[11] Winston, P. H., and Horn, B. K. P. *LISP.* Addison-Wesley, Reading, Ma, 1981.

[12] Stallman, R., Weinreb, D., and Moon, D. *Lisp Machine Manual* Sixth Edition (system Version 99) edition, Artificial Intelligence Laboratory, Massachusetts Institutue of Technology, Cambridge, Ma, 1984.

[13] Burke, G.S., Carrete, G. J., and Eliot, C. R. *NIL Reference Manual* Release 0.286 edition, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Ma, 1984.

[14] Hayes-Roth, F, Waterman, D. A., and Lenat, D. B. (editors). *Building Expert Systems.* Addison-Wesley, Reading, Ma, 1983.

[15] Pope, David E. An Expert System for Airborne Object Analysis. Master's thesis, Massachusetts Institute of Technology, January, 1985.

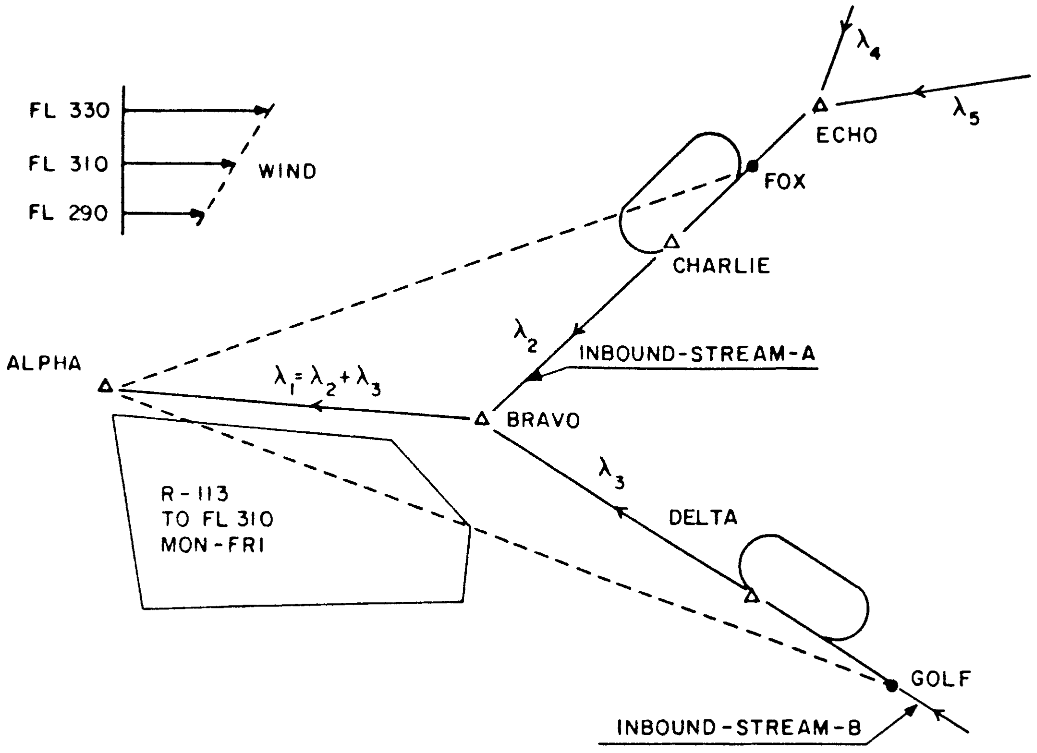

Figure 1.- Basic Geometry of two-stream merge problem.

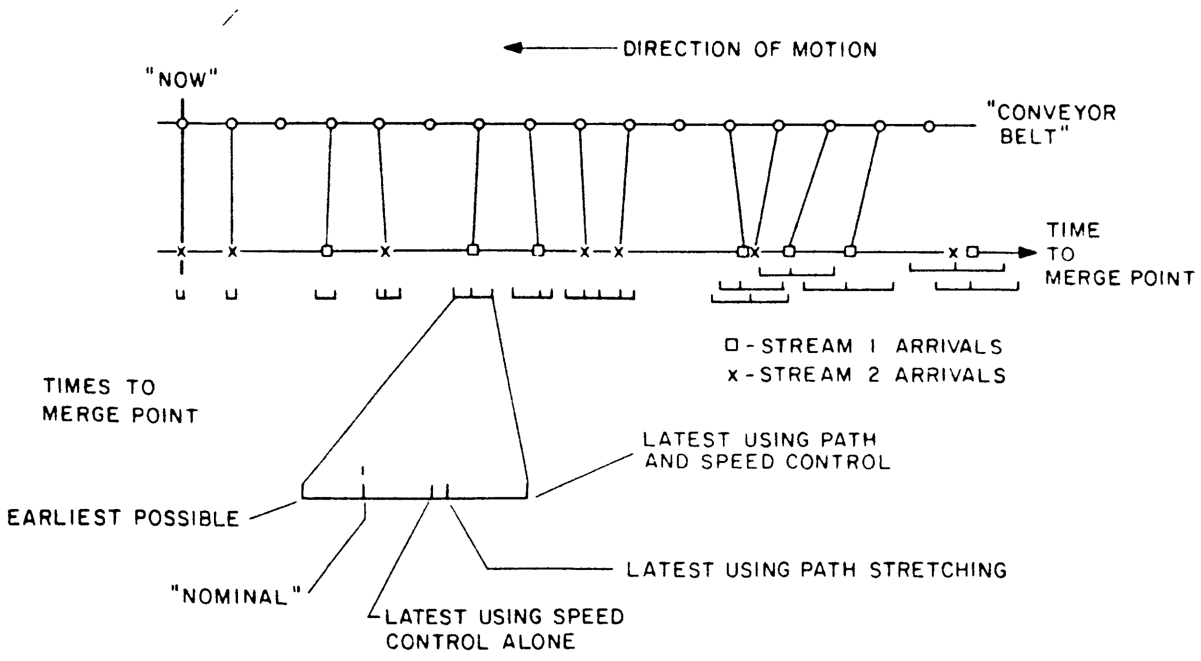

Figure 2.- Time-based Metered-merge formulation of merge problem.

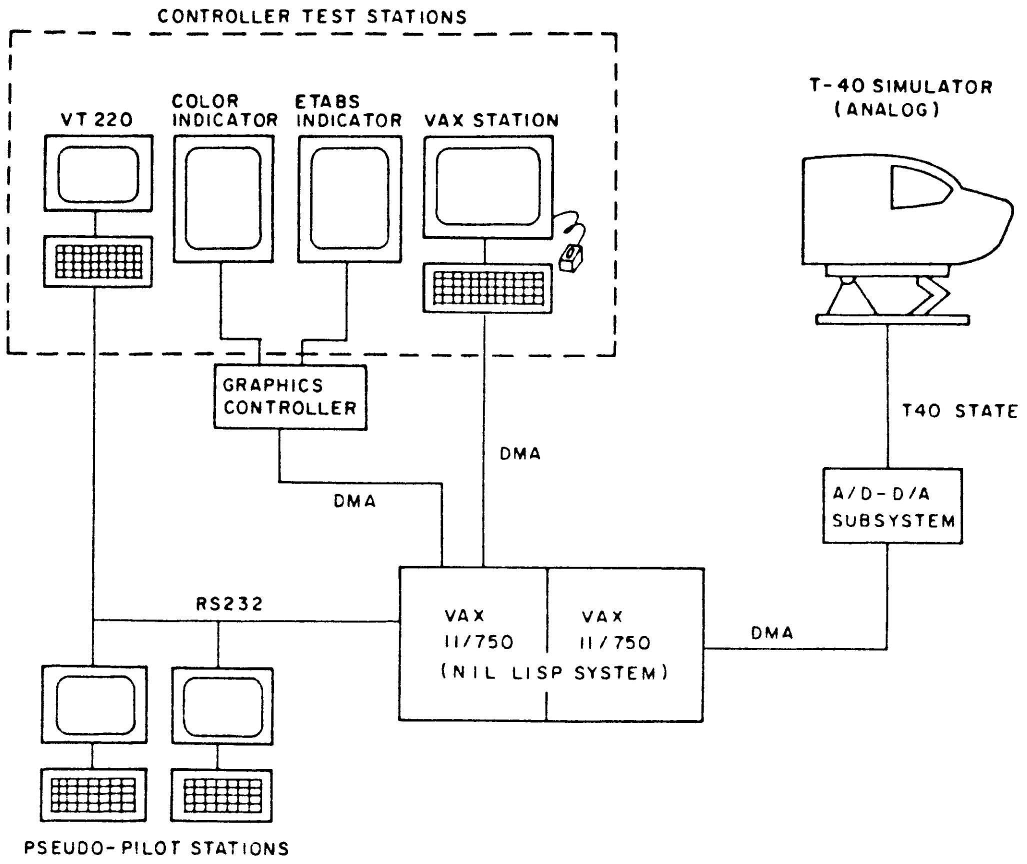

Figure 3.- Overview of the A.I. oriented Air Traffic Control Simulator at the Massachusetts Institute of Technology