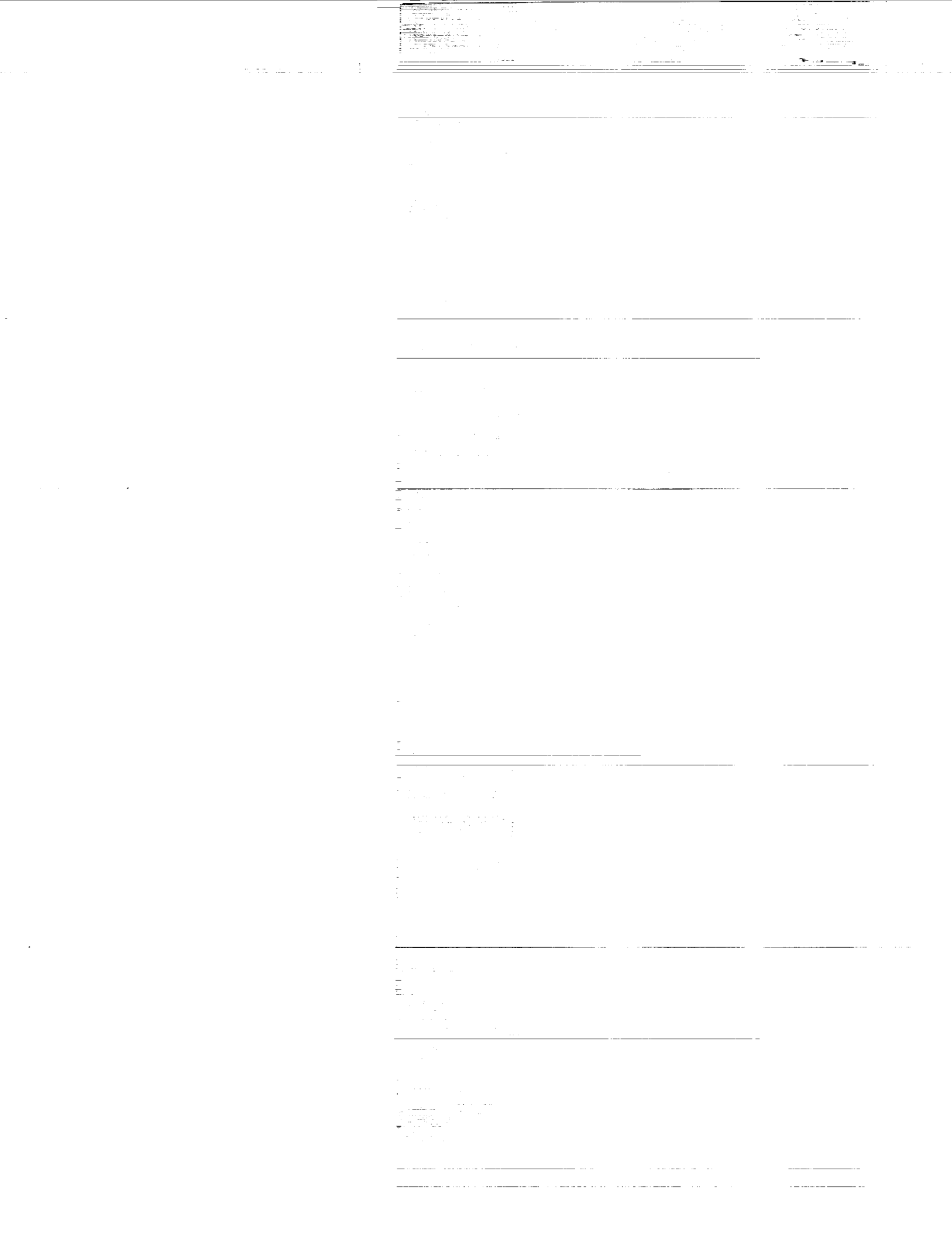
NASA Contractor Report 4507

Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System

Alan R. Chappell

CONTRACT NAS1-19000
MAY 1993



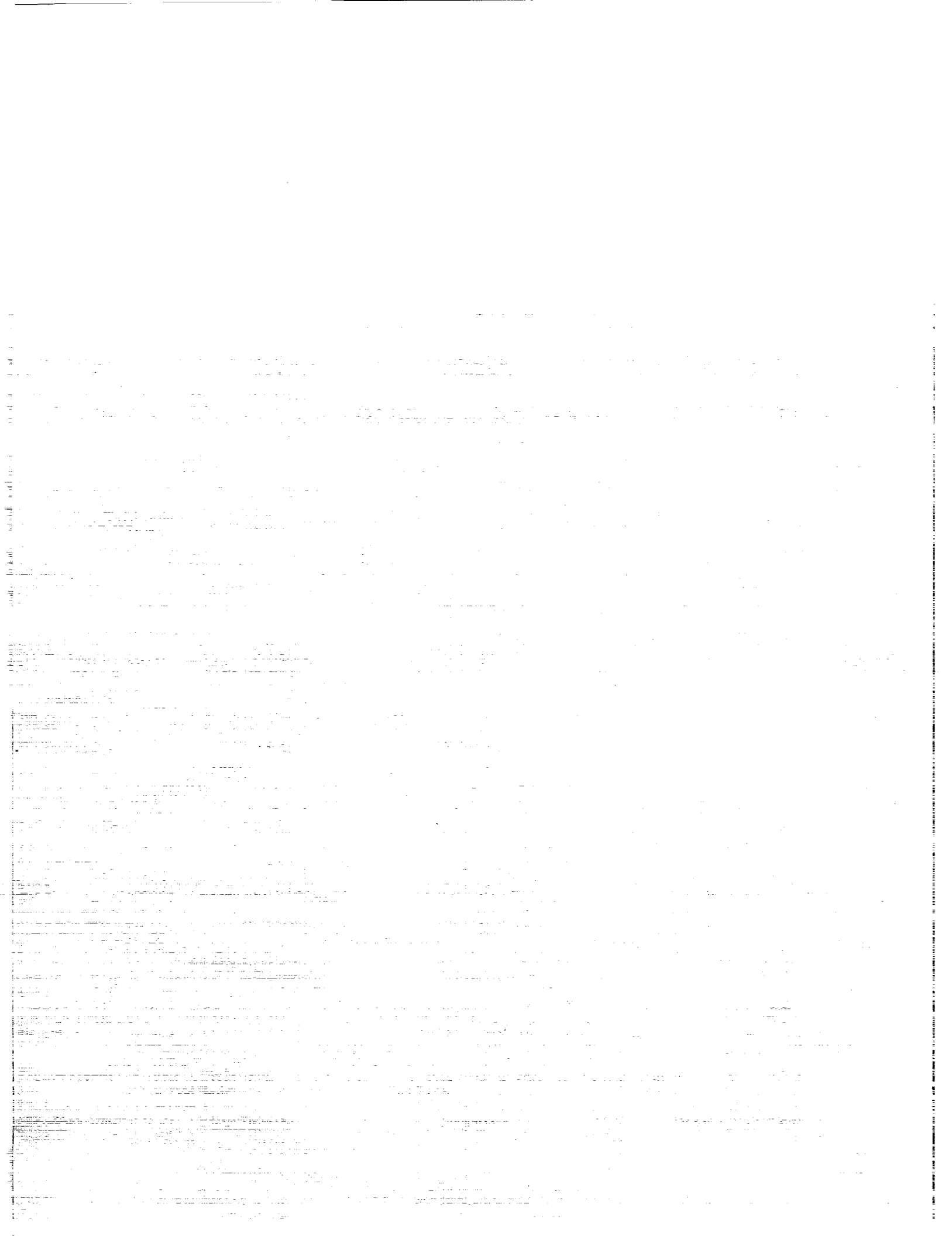
NASA Contractor Report 4507

Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System

Alan R. Chappell
Lockheed Engineering & Sciences Company
Hampton, Virginia

Prepared for
Langley Research Center
under Contract NAS1-19000

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1993



KNOWLEDGE-BASED REASONING IN THE PALADIN TACTICAL DECISION GENERATION SYSTEM

Abstract

A real-time tactical decision generation system for air combat engagements, Paladin, has been developed. A pilot's job in air combat includes tasks that are largely symbolic. These symbolic tasks are generally performed through the application of experience and training (i.e. knowledge) gathered over years of flying a fighter aircraft. Two such tasks, situation assessment and throttle control, are identified and broken out in Paladin to be handled by specialized knowledge-based systems. Knowledge pertaining to these tasks is encoded into rule-bases to provide the foundation for decisions. Paladin uses a custom built inference engine and a partitioned rule-base structure to give these symbolic results in real-time. This paper provides an overview of knowledge-based reasoning systems as a subset of rule-based systems. The knowledge used by Paladin in generating results as well as the system design for real-time execution is discussed.

Introduction

Modern air combat simulations must perform in a greatly expanded and rapidly changing tactical environment. Such a simulation system must be able to model new aircraft and their advanced capabilities. The system should have a modular software structure so that new weapons systems or aircraft subsystems (e.g. sensors or propulsion systems), modifications to aircraft control systems, or changes to the aircraft configuration can be easily incorporated. In support of the study of aircraft with enhanced maneuverability at the Langley Research Center (LaRC), a Tactical Guidance Research and Evaluation System (TiGRES) is being developed. The design and development of TiGRES as well as its relationship to past and current air combat simulation systems is described in detail in (Goodrich, 1990).

The TiGRES system is designed to allow researchers to develop and evaluate aircraft systems in a tactical environment. The three main components of TiGRES are a Tactical Decision Generator (TDG), the Tactical Maneuver Simulator (TMS), and the Differential Maneuvering Simulator (DMS).

A TDG is an intelligent system that selects the combat maneuvers to perform throughout an air combat engagement. Both the TMS and the DMS use a TDG as the automated opponent. A version of the TDG known as Paladin currently is used in TiGRES research.

The TMS (Goodrich, 1992) provides a high-fidelity batch air combat simulation environment for the development and testing of various guidance and control strategies. The researcher defines the initial conditions of the air combat engagement, and the TMS then controls the aircraft using either simple trajectory commands or a TDG. The main elements of the TMS are a high-fidelity, nonlinear six degree-of-freedom (d.o.f.) rigid-body aircraft dynamic model, including the control system, a TDG, and a user interface.

The DMS consists of two 40' diameter simulation domes and one 20' diameter dome. The facility is intended for the real-time simulation of air combat engagements between piloted aircraft. By using a TDG to control one of the airplanes, it is possible to test the TDG against a human opponent. This feature allows the guidance logic to be evaluated against one or more unpredictable and adaptive human opponents.

The Paladin System

Paladin is a knowledge-based TDG. Paladin is implemented using artificial intelligence (AI) techniques and a large amount of information about aircraft, flight dynamics, and air combat so that the system can provide insight into both the tactical benefits and costs of enhanced maneuverability.

Paladin uses an object-oriented programming approach (Meyer, 1988) to represent each aircraft in the simulation. Each aircraft object includes information on the current state of the aircraft's offensive systems (e.g. guns, missile systems, fire control radars, etc.), defensive systems (e.g. electronic counter-measures, chaff, etc.), and

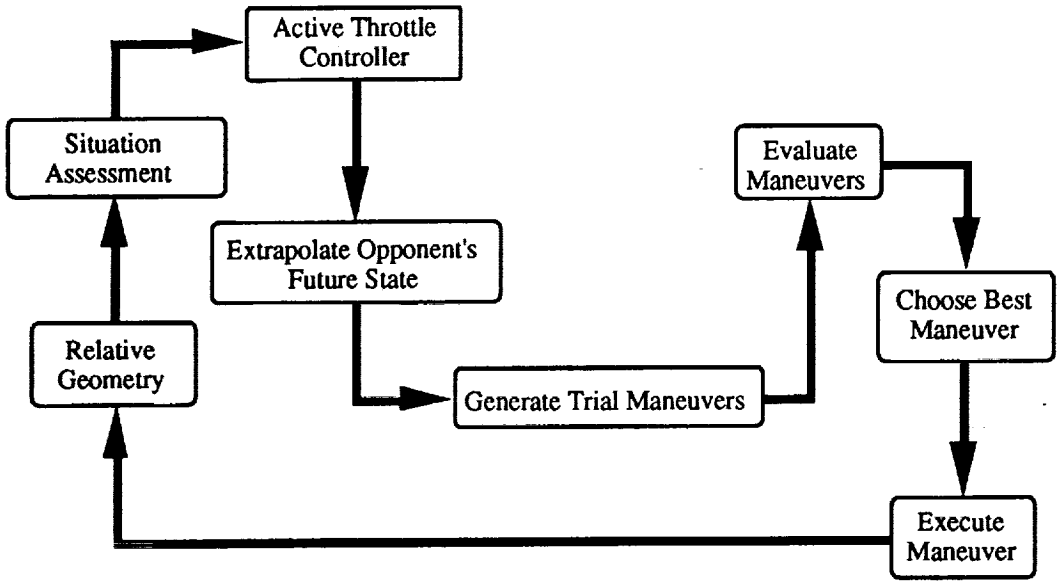


Figure 1. Schematic of The Paladin System

propulsion system. This state information is used to help guide Paladin's reasoning process.

Paladin utilizes modular software subroutines and specialized computer hardware. The separation of the aircraft simulation and decision logic components allows each module or knowledge source to be designed and implemented using the hardware and programming techniques specifically suited for its function. The use of highly specialized and independent knowledge sources also provides for modular protection (Meyer, 1988), confining the effect of an error occurring in a module at run-time to that module, or to a small set of neighboring modules in the program. The confining effect of the modular protection was used to aid in the design and debugging process of Paladin. Each knowledge source was developed and tested independently before it was incorporated into the system.

The independence of the knowledge sources also increases the efficiency of Paladin by allowing knowledge sources to be distributed across a network of several heterogeneous processors. The network currently consists of a Symbolics 3650* workstation, a Symbolics MacIvory* workstation, and four Vax 3200† class workstations. Communication between the distributed knowledge sources is achieved using customized DECNet† based client/server software developed in-house for TiGRES. This software allows for synchronization, communications, and data sharing between heterogeneous computers running the DECNet communications protocol. Paladin is currently implemented as a serial blackboard system (McManus, 1990), so no serialization or concurrency related software is required. Each knowledge source requests all of the data required to perform its computation from the blackboard at the start of its execution cycle, and posts its results to the blackboard at the end of its execution cycle.

Paladin models a combat engagement as a series of discrete decisions. At temporally regular decision points (0.25 second intervals under most circumstances), the system chooses the "best" tactical maneuver to follow until the next decision point. To make this choice, Paladin uses information about its own state, information about the opponent's position, and estimated data about the opponent's orientation to calculate the relative geometry between the two aircraft. This relative geometry is used to perform a situation assessment and to select a new throttle position. After extrapolating the opponent's state a short time into the future, Paladin generates a situationally dependent set of trial maneuvers (Chappell, 1992). A future engagement state is predicted for each of the trial maneuvers. These future engagement states are passed through a group of scoring

* Symbolics 3650 and MacIvory are registered trademarks of Symbolics Incorporated.

† Vax 3200 and DECNet are registered trademarks of Digital Equipment Corporation.

functions that evaluate various aspects of the tactical situation. The results of the scoring functions are weighted, based on the mode of operation, to compute the current best maneuver. This maneuver is then used to direct the aircraft until the next decision interval. Figure 1 is a schematic of Paladin.

From this list of tasks, two have been identified as largely symbolic in nature. Situation assessment and selecting a new throttle position primarily involve classification of the current situation. Paladin performs these tasks, in real-time, through specialized knowledge-based systems. The design and implementation of these knowledge-based systems are the focus of this paper. First, however, a brief general discussion on knowledge-based systems and their relation to the more well known expert system is presented.

Knowledge-Based Systems

Knowledge-based systems, along with expert systems, are rule-based (or production) systems (Brownston, 1985). As rule-based systems they share many characteristics, including structure and control mechanisms. All rule-based systems have three basic parts, working memory, rule-base, and inference engine. Working memory is the data structure that holds the known facts defining the current situation. The rule-base is a group of rules (productions) that represent the available knowledge about the problem. The inference engine is the control mechanism for executing the rules. All rule-based systems run by matching rule conditions to the facts in working memory, choosing one rule from those with all conditions satisfied, executing that rule to change the situation or perform a function, and starting the cycle again.

The differences between knowledge-based systems and expert systems are largely in their problem solving strategy. An expert system is so named because it attempts to solve the problem in the same manner as the human expert. The system should follow similar lines of reasoning and arrive at the same conclusion as the expert that is being used as a knowledge source. A knowledge-based system on the other hand is based on general information about the problem domain. Although input from experts can be incorporated, they are not the primary source and no attempt is made to mimic their actions. Knowledge-based systems can be more robust than expert systems (i.e. give better coverage of the problem domain) but also may lack the learned heuristic response of a human.

Paladin was implemented as a knowledge-based system for several reasons. First, air combat experts (i.e. fighter pilots) are not readily available, so building an expert system would be more difficult. More importantly, though, is the application for which Paladin is designed. Paladin needs to be able to exploit new aircraft capabilities before the new system or aircraft is built. No real world expertise exists on which to build such an expert system. Therefor, Paladin is based on general air combat principles that help to develop new tactics and maneuvers that can utilize hypothetical capabilities.

As a rule-based system, Paladin incorporates the three components listed previously. The working memory is stored using an object oriented approach as discussed in the previous section. The design and implementation of the inference engine and the rule-bases are of particular interest in this paper.

Inference Engine

The inference engine of a knowledge-based system controls the execution of rules from the rule-bases. The inference engine must determine which rules are active, evaluate the preconditions of the active rules, collect the rules with all preconditions satisfied, choose one rule to execute through conflict resolution (Brownston, 1985), and execute that rule. Types and implementations of inference engines vary widely, with equally varying performance results.

Paladin uses a custom built inference engine that was designed to support real-time execution of knowledge-based systems. The inference engine uses a depth-first evaluation strategy (Barr, 1981) to search the active rule-bases. This results in a branch of the decision tree being followed until that branch fails or succeeds, before the next branch is examined. A rule priority conflict resolution scheme is used, with rule order taken as implied rule priority. Hence, the first rule that can be executed, is executed and that inference cycle is ended. This forces the rule-base designer to prescribe execution priority off-line, instead of using limited computation resources for a search of the entire decision tree followed by complicated algorithms to choose one rule for execution.

Paladin's inference engine supports partitioning of rule-bases using meta-rules to guide partition activation. When a meta-rule is encountered that activates a new partition, it has the effect of initiating the inference process on a new decision tree and a depth-first search starts down that new tree. The importance and use of partitioning in Paladin will be discussed in the next section.

The inference engine can be stated in psuedo-code as follows:

```
(subroutine inference-engine (rule-list)
    (if rule-list empty -> quit)
    (if preconditions of first rule in rule-list
        are satisfied -> execute rule action
        and quit)
    (otherwise -> call inference-engine on
        rule-list with first rule removed) ) .
```

The input to the inference engine is a list of rules that includes each rule in the rule-base in priority order. Each rule is expected to have a set of preconditions and an associated action. This action can be any computable function, including a call to the inference engine with another rule-base (meta-rule activation of a new partition). The inference engine is written in Lisp and runs on an AI workstation.

Rule-Bases

A knowledge-based system stores the knowledge used to solve a problem as rules in a rule-base. The knowledge is used by finding a rule that matches the current conditions (preconditions satisfied) and executing its associated action. The design and set-up of this rule-base greatly impacts the performance of the system.

Paladin's rule-bases are expressed in two formats: interpreted lists of condition action pairs used during the design stage, and compiled lists of in-line function definitions used in the final, real-time version of the system. The interpreted lists are used to develop and debug the initial versions of the rule-base. Most existing rule-based systems stop development at this point and implement the interpreted rule-bases. The use of the interpreted rules severely limits the execution performance of the inference engine, thus restricting the real-time usage of this type of system. To overcome this problem and allow real-time execution, Paladin's rule-bases are "compiled" into a list of in-line functions. The compiled rule-bases execute approximately 90 to 100 times faster than the interpreted rules. On a Symbolics 3650™, the inference engine executes a representative test rule-base consisting of 40 rules in the interpreted format in 170 milliseconds. The inference engine executes the same rule-base in the compiled format in 1.9 milliseconds.

There is a direct relation between the length of the rule-base's longest execution path and the knowledge-based system's execution time. The shorter the execution path is, the shorter the execution time. The rule-bases used by Paladin have been partitioned to increase system performance by grouping related rules into small partitions and using meta-rules to link the partitions. This partitioning decreases the number of rules that are active, and decreases the length of the worst-case execution path through the rule-base. The rule-base partitioning allows the designer to calculate the longest and shortest path through the rule-base and compute both a maximum and minimum knowledge source execution time. The knowledge source's maximum execution time can be used to insure that the system will meet real-time execution requirements. If the maximum execution time exceeds the allocated execution time, the designer may be able to repartition the rule-base until real-time execution requirements are achieved.

Paladin makes use of two rule-bases: a mode selection rule-base used by the Situation Assessment Module, and a throttle control rule-base used by the Active Throttle Controller. The design and knowledge used in these rule-bases are described in the following sections.

Mode Selection Rule-Base. Six modes of operation have been incorporated into Paladin. These modes are aggressive, defensive, evasive, ground avoidance, neutral, and disengage. The Situation Assessment Module is used to model a pilot's situational awareness and changing problem solving strategies. Just as a pilot will recognize the difference between an aggressive and an evasive situation and react accordingly, the Situation Assessment Module provides information allowing Paladin to adapt its problem solving strategy based on the current situation. The determination of the current mode of operation is based on the aircraft's current mission, the current state of the aircraft's

Table 1. Mode Selection Decision Matrix

Mission	Position Classification			
	Evasive	Neutral	Defensive	Aggressive
Evasive	Evasive	Evasive	Evasive	Evasive
Neutral	Evasive	OU: Neutral OD: Neutral or Disengage	DU: Defensive DD: Evasive	Aggressive
Defensive	Evasive	Aggressive	DU: Defensive DD: Evasive	Aggressive
Aggressive	Evasive	OU: Aggressive OD: Aggressive or Disengage	DU: Aggressive DD: Evasive	OU: Aggressive OD: Aggressive or Neutral

OU: Offensive systems up (active)
OD: Offensive systems down (not active)
DU: Defensive systems up
DD: Defensive systems down

systems, and the relative geometry between the aircraft and its opponent.

The Situation Assessment Module is the knowledge-based system which uses the mode selection rule-base. The mode selection rule-base consists of five partitions and contains nineteen rules. The shortest execution path in this rule-base results in a single rule being evaluated, while the longest path results in twelve rules being evaluated. Table 1 is a representation of the decision matrix on which the mode selection is based. Mission is the combat mission assigned to the aircraft, as set prior to the engagement. Position classification is an evaluation, made by the knowledge-based system, of the current relative geometry between the aircraft. Unless failures or situation related problems (e.g. low fuel) take precedence, this matrix dictates the selected mode. In the matrix elements, when a down system can generate two results, the first is selected if the down system is simply turned off, while the second is selected if the system is not functional. A complete version of the mode selection rule-base is presented in (McManus, 1992).

Throttle Control Rule-Base. A rule-based Active Throttle Controller determines throttle and speed brake settings based on the engagement situation. The throttle controller can set the throttle to any position between idle and full afterburner, and the speed brake to any position between fully retracted and fully extended. The throttle controller uses the current mode of operation and the relative geometry information to select one of four operational modes. These four modes are: go to best corner speed, go to best range for firing, close/separate quickly, and force overshoot. Each mode has a set of specific throttle control rules that are used to maximize system performance in that throttle mode.

The Active Throttle Controller uses the throttle control rule-base. This rule base consists of eleven partitions and contains 34 rules. The shortest execution path in this rule-base results in two rules being evaluated, while the longest path results in thirteen rules being evaluated. Figure 3 is a pictorial view of the decision tree for choosing the throttle control operational mode. The complete throttle control rule-base is presented in (McManus, 1992)

Example Cycle

The complete knowledge-based systems are significantly more complicated than the limited sections shown in Figure 3 and Table 1. A large amount of preliminary classification is done to make the code more readable, understandable, and maintainable. In order to show the full range of function, an example cycle is given.

Choosing a point somewhere in the middle of a hypothetical engagement, the following describe the pertinent aspects of the current situation. The Paladin controlled aircraft is behind, a little below, and to the right of the opponent at a range of 9000 ft. The closing rate is 400 ft/sec. with the throttle currently set at 1.5 (halfway between military

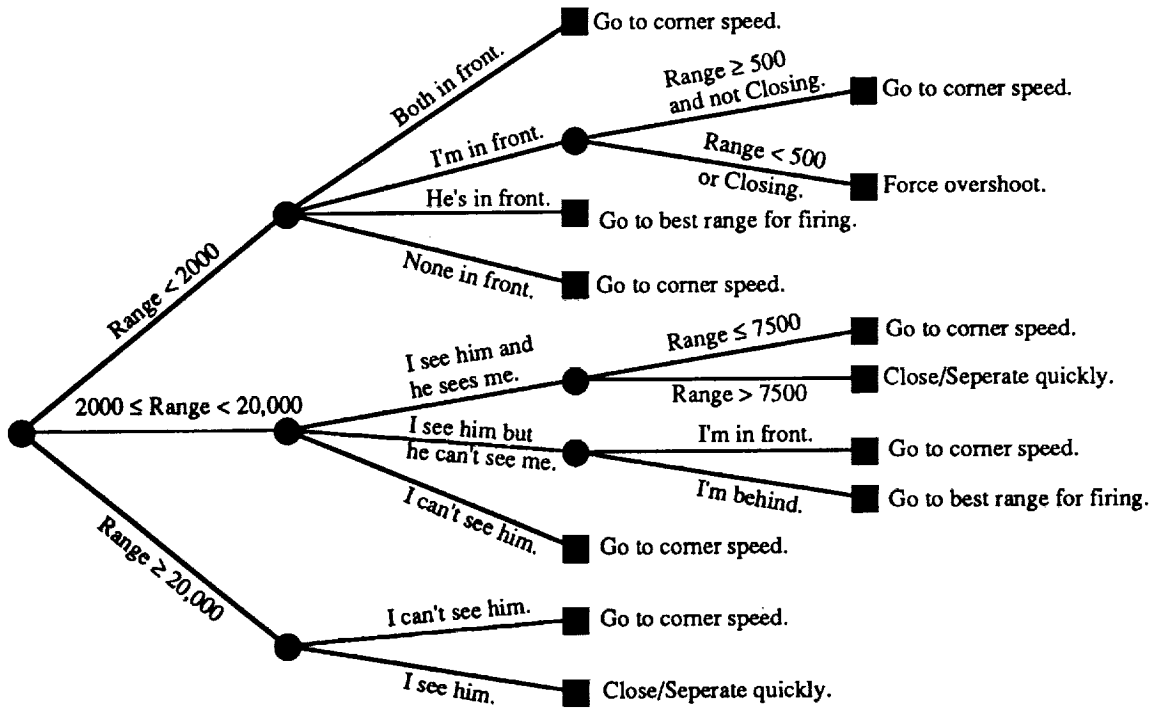


Figure 3. Throttle Mode Decision Tree

power and maximum afterburner). As for the aircraft, both the offensive and defensive systems are operational, and the engine/fuel level are registering satisfactory. Paladin's mission has been preset to aggressive.

Given this state, Paladin performs the following analysis. First, using the relative positions and the angular measurements between the aircraft, a knowledge source decides that Paladin can see his opponent while the opponent can not see Paladin. Similarly, another knowledge source indicates that Paladin is behind his opponent and that the opponent is not behind Paladin. Then, the closing rate is classified as "closing fast".

The primary partition of the throttle controller is then activated. The first applicable rule classifies the range as medium distance and so this meta-rule activates the medium range partition. Given the preliminary classification that Paladin can see the opponent, the opponent can not see Paladin, and that Paladin is behind the opponent, another meta-rule fires activating the set range partition, which corresponds to the throttle mode indicated in Figure 3. The set range partition is only responsible for setting the best range for firing the weapons. Since 9000 ft. is a good firing range, a meta-rule activates the hold range partition. Since Paladin is "closing fast", a rule fires setting the new throttle to 95% of the old throttle in order to reduce the closing rate.

Next a knowledge source classifies the tactical position of Paladin. Seeing that Paladin is in missile range but the opponent does not have a firing solution, and that Paladin sees the opponent while the opponent cannot see Paladin, the position is labeled aggressive.

The primary partition of the situation assessment rule-base is then activated. Since none of the special case rules fire (i.e. not in ground avoidance, not low on fuel, no engine problems, not in evasive position, ...) control is left to four meta-rules based on whether the offensive and defensive systems are operational. In this situation both systems are up. Therefor, the meta-rule fires which activates the offensive systems up partition and the defensive systems up partition in that order (if no rule fires in the first partition, search will continue into the second). With both the mission and tactical position labeled aggressive, a rule fires to set the mode of operation to aggressive. This corresponds to the matrix entry shown in Table 1.

Thus, the knowledge-based systems finish this hypothetical cycle, leaving an aggressive mode of operation and a new throttle position of 1.425. This cycle is repeated each time Paladin makes a maneuver decision to ensure that situational awareness is incorporated into the decision process.

Conclusions

Paladin, a computerized air combat tactical decision generator, has been developed to study air combat engagements. The system incorporates modern sensor and weapon system models, and aircraft simulation techniques. Paladin uses artificial intelligence techniques to address air-to-air combat and agile aircraft in a clear and concise manner. The Differential Maneuvering Simulator offers a unique opportunity to evaluate the performance of the Paladin software in a real-time tactical environment against human pilots.

Paladin models aspects of the decision-making processes used by human pilots through the application of knowledge-based systems. Complete dependance on the past experience of fighter pilots (expert system) would produce a rule-based system so locked into current capabilities and tactics that it would be unable to thoroughly test new aircraft capabilities. Paladin avoids this difficulty by relying on general sources of information about aircraft and air combat (knowledge-based system). Results are produced in real-time through rule-base partitioning and compilation.

References

Barr, Avron, and Edward A. Feigenbaum (eds.), *The Handbook of Artificial Intelligence, Vol. I*, William Kaufmann, Inc., 1981.

Brownston, Lee, et al., *Programming Expert Systems in OPS5*, Addison-Wesley Publishing Co. Inc., 1985.

Chappell, Alan R., Dr. John W. McManus, and Kenneth H. Goodrich, *Trial Maneuver Generation and Selection in the Paladin Tactical Decision Generation System*, AIAA Paper #92-4541, August 1992.

Goodrich, Kenneth H., and John W. McManus, *An Integrated Environment For Tactical Guidance Research and Evaluation*, AIAA Paper #90-1287, May 1990.

Goodrich, Kenneth H., Dr. John W. McManus, and Alan R. Chappell, *A High-Fidelity Batch Simulation Environment for Integrated Batch and Piloted Air Combat Simulation Analysis*, AIAA Paper #92-4145, August 1992.

McManus, John W., "A Parallel Distributed System for Aircraft Tactical Decision Generation," *Proceedings of the 9th Digital Avionics Systems Conference*, 1990, pp. 505 - 512.

McManus, John W., Alan R. Chappell, and P. Douglas Arbuckle, "Situation Assessment in the Paladin Tactical Decision Generation System," *AGARD Conference Proceedings 504: Air Vehicle Mission Control and Management*, March 1992, pp. 8:1-8:16.

Meyer, Bertrand, *Object-oriented Software Construction*, Prentice Hall International Ltd, 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1993	Contractor Report

4. TITLE AND SUBTITLE
Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System

5. FUNDING NUMBERS
C NAS1-19000
WU 505-64-30-01

6. AUTHOR(S)
Alan R. Chappell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Engineering and Sciences Company
Langley Program Office
144 Research Drive
Hampton, VA 23666

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-4507

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Frederick R. Morrell
Final Report

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified-Unlimited

Subject Category 08

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A real-time tactical decision generation system for air combat engagements, Paladin, has been developed. A pilot's job in air combat includes tasks that are largely symbolic. These symbolic tasks are generally performed through the application of experience and training (i.e. knowledge) gathered over years of flying a fighter aircraft. Two such tasks, situation assessment and throttle control, are identified and broken out in Paladin to be handled by specialized knowledge-based systems. Knowledge pertaining to these tasks is encoded into rule-bases to provide the foundation for decisions. Paladin uses a custom built inference engine and a partitioned rule-base structure to give these symbolic results in real-time. This paper provides an overview of knowledge-based reasoning systems as a subset of rule-based systems. The knowledge used by Paladin in generating results as well as the system design for real-time execution is discussed.

14. SUBJECT TERMS
knowledge-based controls, rule-based systems, air combat simulation, distributed computing, real-time, situation assessment, tactical decision generation

15. NUMBER OF PAGES
8

16. PRICE CODE
A02

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 728-064/66036