N88-17217

AN INTELLIGENT TUTOR FOR THE SPACE DOMAIN

*Dr. Kathleen Swigger**
Harry Loveland

North Texas State University
Computer Sciences Department
Box 13886
Denton, Texas 76203

ABSTRACT

This paper describes an intelligent tutoring system for the space domain. The system was developed on a Xerox 1108 using LOOPS and provides an environment for discovering principles of ground tracks as a direct function of the orbital elements. This paper also looks at some of the more practical design and implementation issues associated with the development of intelligent tutoring systems. It attempts to offer some solutions to the problems and some suggestions for future research.

INTRODUCTION

The Intelligent Tutor for Ground Tracks (nicknamed OM) is designed to teach students how to "deduce" a satellite's orbital elements by looking at a graphic display of a satellite's ground track. In order to help the student understand these relationships, the system was given a special interface that allows student to freely investigate different options and "discover" relationships between various parameters. If, however, the student does not "discover" these principles and concepts, then OM intervenes and directs the student toward specific goals.

One of the basic missions for space operations personnel is the continuous monitoring of the exoatmospheric arena through ground and space surveillance. For example, NORAD, through its Space Defense Center, maintains a worldwide network that senses, tracks, and analyzes the characteristics of orbiting systems. In order to monitor and plan for satellite missions, space operations crews must be able to read and understand ground tracks. Ground tracks are two-dimensional displays that show the portion of the earth that a satellite covers in one orbit. The ground track is a direct function of the orbital elements, so proper understanding of these functions and their effect on the shape of the ground track is critical for anyone interested in satellite operations.

One way to teach students how to deduce orbital elements from a satellite's ground track is to present the various formulas that are used to compute the orbital elements and then show students how to apply these formulas to situation-specific tracks [Bates et al., 1971; Astronautics, 1985] In contrast to this approach, we discovered that experts store ground tracks as graphical representations, indexed by feature and shape. Based on previous experience, experts learn how to detect any features such as size, number of loops, direction, etc., and then use this information to "estimate" the orbital elements. In order to duplicate this process, we decided to build a qualitative model of how the expert predicts orbital elements, given specific shape descriptors, and then use this model as a basis for teaching students the effects of different orbital parameters on the shape of the ground track.

*This research was supported, in part, with a grant from the Office of Scientific Research which was conducted at the Air Force Human Resources Laboratory (AFHRL), San Antonio, Texas.

STUDENT/COMPUTER INTERACTION

As previously mentioned, the microworld for the Ground Track problem offers a number of online tools that permit students to discover relationships between orbital parameters and ground tracks. This environment consists of an elaborate ground track display (Figure 1) and a number of interactive tools designed to encourage systematic behaviors for investigating ground track related problems. The student initiates a discovery activity by changing one or more orbital parameters or changing the injection parameters. This task is accomplished by positioning the cursor over the individual parameters and pressing the left mouse button to increase the value or the middle button to decrease the value. The injection point is changed by positioning the cursor over a particular point on the map and pressing the left mouse button, which automatically sets both the longitude and latitude. A student can observe the results of these changes by selecting "Generate Ground Trace" from the main menu. After investigating the effects of changing different parameter values for different ground tracks, the student can advance to the Prediction window where he can make a hypothesis regarding the particular shape of a ground track.

In the Prediction portion of the program, the system displays a list of words that describe various features about ground tracks such as shape, size, and symmetry (Figure 2). From this list of descriptors, the student selects the words that "best" describe the current ground track under discussion. The student then tests his prediction by selecting this option from the menu and comparing the inputs to the Expert's conclusions. The student can then interrogate the Expert System by placing the cursor over any of the descriptors and pressing the left mouse button. A "why" pop-up menu appears on the screen which enables a student to receive an explanation of the Expert's reason for the correct descriptor. A student may also interrogate his own selections by placing the

cursor over his "input" selection and pressing the right button. In this instance, a "why not" pop-up menu appears and displays the reasons why a particular descriptor was an inappropriate selection. The student can continue in this manner until he understands the various relationships between the shape of a ground track and the different orbital parameters.

After making several successful predictions, the student enters an Orbit Prediction environment which is designed to check the student's predictive powers by asking him to perform a task in the reverse order of the one described above. The student is shown a specific type of ground track and asked to enter actual orbital descriptors of the ground track. If the student is successful, then he can continue to explore different types of ground tracks. If the student is unsuccessful, then he receives information about why his answers are incorrect.

TOOL DESCRIPTION

There are three major online tools that can be used by the student to gather information and to understand concepts and principles about ground tracks. These tools are a) a History tool that allows the students to overlay previously generated ground tracks and note relationships between parameters b) an Orbit window that displays a two-dimensional representation of the orbit (Figure 1); and c) a Definition/Example tool which displays factual information about different orbital parameters (Figure 1) and orbital descriptors.

The History tool is specifically designed to help students recognize relevant patterns between and among previously generated ground tracks. As the student generates various ground tracks, the system collects and stores each transaction. The student can retrieve any of this data by selecting the History option from the main menu. A list of the past twenty ground tracks appears on the screen from which the student can select one or more related ground tracks. The system then overlays the selected ground tracks onto a single map. Again, the student observes the results of this exercise.

For any given set of orbital parameters, the student can obtain a two-dimensional display which shows the position of the satellite in relationship to the earth. The student selects the option labelled Orbit window and gains immediate access to this particular display. The Orbit window is especially useful for demonstrating the relationship between the ground track and the actual orbit and for illustrating the effect of perigee on elliptical orbits.

The Definition/Example tool provides the student with the factual knowledge about the domain. A student can obtain definitions and examples for orbital parameters and the shape descriptors by simply placing the cursor over the keyword in question and pressing the right mouse button. A pop-up menu appears on the screen from which the student can select definitions, examples or explanations. The explanations for the orbital parameters are generated according to the context that they appear.

Thus by using the available tools, a student can obtain facts about the orbital world (through the Definition/Example tool), see relationships between different ground tracks (through the History window), and understand certain principles about satellite operations (through the Orbit window). A student has the option of using any of these tools at any time during the computer/student interaction. If, however, the student is not making sufficient progress, the system interrupts and directs the student to use a specific tool to achieve an objective.

DESIGN OF THE SYSTEM

Overview

Although the system is composed of five logical units (an Expert system, a Curriculum, a State Module, a Student Model, and a Coach), the Tutor is actually implemented as a series of LOOPS classes and objects. Thus, the Tutor's logical units do not necessarily correspond to specific programming segments. The Expert, for example, is implemented as a series of Shape Objects* which contain both the rules and inference procedures used to deduce shape descriptors from a set of orbital parameters. These Shape Objects also contain the major concepts associated with the ground track curriculum (the Curriculum). The State Module contains a list of appropriate behaviors for exploring the microworld. There is also a series of methods** used to evaluate the student's answer, analyze student errors, and update counters. The Student Model resides within the Expert Objects in the form of counters and threshold values which reflect the student's current state of knowledge of both ground tracks and effective tool use. Finally, there are a series of Coaching methods that tell the system when to intervene. The system makes its decision based on information regarding the student's current state of knowledge. A more detailed description of each logical unit is presented below.

The Expert

This module contains the rules and procedures used to deduce shape descriptors (e.g., closed-body, symmetrical, vertical; compressed, lean-right, hinge-symmetry, with loops) from a set of orbital parameters (eccentricity, period, semi-major axis, argument of periapsis, inclination). The Expert is invoked only when the student is making a shape or orbit prediction. The general problem solving strategy employed by the Expert is to determine a shape descriptor by examining a specific orbital element. If this fails, then the system looks at another shape descriptor and attempts to find its value, or looks at a combination of two or more orbital elements to see if the system can deduce a shape descriptor. For example, the Expert determines the symmetry shape goal by asking whether this is a circular orbit. If the orbit is classified as a circular orbit, then its eccentricity must be equal to zero. If the orbit is elliptical then its eccentricity is not equal to zero and the Expert must look at the orientation descriptor, which in turn must look at the argument of periapsis. In this manner, the Expert Module can determine a set of shape descriptors for a given set of orbital parameters (and vice versa). During the process of deducing shape descriptors, the Expert also determines the optimal "procedure" for deriving the shape descriptors. Thus both declarative and procedural knowledge is available to the rest of the tutor.

*Objects is a trademark for data types in the LOOPS programming environment, Xerox, Corp.

**Methods is a trademark for procedures in the LOOPS programming environment, Xerox, Corp.

At the implementation level, the Expert shape descriptors are organized as classes and sub-classes (Figure 3). The Expert operates by calling a "metal-rule" that sends a message to all the objects to test the rules associated with each of the objects and return the values from the rules that are true. Along with the Expert's If...then rules, each object contains the definition, explanation templates, examples, special counters indicating the number of times the student predicts the shape descriptors correctly and incorrectly, tutoring strategies, conflict resolution strategies, and special buggy rules. This particular data proved to be a very effective way of organizing the knowledge.

Another function of the Expert is to deduce parameter descriptors (such as a Circular, Synchronous orbit) at the same time that the system is deducing the shape descriptors. These parameter descriptors are used to determine the essential skills that are necessary to understand a given ground track. Since the rules for determining the Curriculum are used by the Expert rules, we now describe the organization of the Curriculum.

The Curriculum

Along with knowledge about shape descriptors for ground tracks, a student must also understand how this information relates to specific orbit types. For example, an orbit which has a semi- major axis equal to 42,250 kilometers is said to be in a synchronous orbit. This term applies to all ground tracks that have a semi-major axis equal to 42,250 kilometers, regardless of the numbers that might appear for the other orbital parameters. Thus it is important that students recognize the relationship between the specific domain knowledge and the qualitative model produced by the Expert. Therefore the System organizes this knowledge in the Orbit Objects (Figure 4) which contain the specific content that is used to categorize different orbit types. The knowledge stored in the Orbit Objects is then used (and deduced) by the Expert. For example, the Expert System determines whether an orbit is circular or elliptical as it deduces the symmetry goal. The knowledge about shapes and orbit types are an integral part of the Expert.

This particular organization also provides a very powerful tool for relating the content areas and for determining various levels of difficulty. For example, the rules that determine the shape descriptors associated with circular orbits tend to have fewer constraints attached to them, and also tend to be fired first, and, as a result, tend to be easier for the student to learn. The hierarchy of orbit types as represented in the Orbit Objects shows both the order that the knowledge should be learned and the relationships between the knowledge. This information can be used by to recommend easier problems whenever the student becomes confused.

The State Module

The State Module contains a list of goals and subgoals which presumably indicate acceptable procedures for exploring the microworld. As the student proceeds through each of the states, the tutor records his/her actions. The authors have hypothesized that a student indicates appropriate experimental behaviors if explores a microworld by generating ground traces. The student then moves on to "making predictions," followed by testing and validating tests, and then generalizing these principles. Each one of these states, in turn, has separate subgoals which may or may not be met. The Tutor uses the State Module in two ways. First, if the student is performing poorly, then the Tutor checks to see if the student has proceeded through each state in an appropriate manner. Second, the system uses the State Module to reflect different "instructional" strategies. For example, if the student is conducting experiments (as defined as "making predictions") then the system gives a higher status to using tools correctly. If the student is "testing," then OM will switch its strategy and try rules that check for skill deficiencies.

The Student Model

The Student Model in embedded within the Expert and Orbit shape Objects as a series of counters that reflect the student's current understanding of both the domain knowledge and investigative behaviors. Whenever the student tests a prediction, OM records a list of the rules that the student understands. The Student Model maintains a series of counters for each rule indicating the number of times a rule is used appropriately, inappropriately, or ignored (a "missed-opportunity" as defined in Carr and Goldstein, 1977). If the missed-opportunity counter exceeds the used-appropriate counter, then the Coach recommends intervention.

The system also records the number of times that an online tool is invoked. In addition to this counter, an effectiveness measure is maintained for the History tool, the Orbit window, and the Definition/Example tool. If the student demonstrates inefficient behavior as indicated by one of the effectiveness measures, then the system intervenes and offers advice.

The Coaching Strategy

OM also maintains a series of rules and procedures that direct the teaching portion of the Tutor. The Ground Track Microworld is designed for two major purposes: 1) to teach students about the relationships between/among orbital elements and ground tracks, and 2) to teach students how to use systematic behaviors to investigate this domain. Thus, the system intervenes when either one of these conditions is not satisfied. The system monitors the student's actions and determines when the student needs advice. Intervention occurs only when the student is making erroneous predictions for either the Shape or Orbit descriptors.

The general or high-level teaching strategy is as follows:

If the student has made No errors
and if the student is completing
curriculum materials efficiently
then record progress

If the student has made No errors
and if the student is NOT completing
the curriculum materials
efficiently
then recommend an easier curriculum

If student has made error
then
a) Check ruleset for
satisfaction of preconditions
b) Check ruleset for
Correct Tool Use
c) Check ruleset for
Skill remediations

The author made the general assumption that when the student is in the Prediction Mode, then the system should help students discover the objectives by having them use the tools correctly. If this fails, then the system should address individual skill errors. This strategy is reversed whenever the student enters the Orbit Prediction Testing State.

The system's overall intervention strategy is to check whether the student has completed the necessary preconditions (as determined by the values stored in the State Module). If the student has satisfied all the preconditions for an exercise, then OM checks the measures for effective inquiry skills. The list of effective inquiry skills as originally defined in Shute and Glasser [1987] include skills such as: Systematic experimental behaviors such as making Predictions, asking for Definitions and Examples or accessing the Orbit window.

Every time a student enters a prediction for Shape descriptors or Orbital parameters, the system evaluates the student counters and determines if intervention is required. If the student's effectiveness measures are low, then the Coaching methods propose possible remediation and offer assistance. In the event that the student fails to attain a level of proficiency after receiving instruction on effective tool use, then the system addresses the student's domain knowledge inadequacies.

At the present time, OM uses the information stored in both the Tool Objects and the Expert Objects to advise the student concerning errors. Initially, the system suggests that the student use one of the available tools to correct his errors. If the student continues to have difficulty, then the system may display the definitions, examples or explicitly state the relationships between various parameters.

Whenever you design and implement a computer system, especially an AI system, you always discover some interesting things about the problem that you wish to share with colleagues and friends. This particular project proved NOT to be the exception to this rule. As the Tutor took shape, and as we better understood the domain, we learned several things that will be helpful as we develop the NEXT tutor. What follows is a discussion of some of these ideas.

Don't be afraid to admit that you are ignorant.

Unfortunately, most people tend to believe that they are all- knowing or, at the very least, too proud to admit that they are not all-knowing. This can be a real problem if your job is to design an expert system. One of the reasons that you need to perform knowledge elicitation is that someone or something has more information (and procedures) than anyone else. Thus, you must perform the painful task of questioning the Expert in order to "discover" the knowledge and procedures that are necessary to perform the task. This requires that the Knowledge Engineer admit ignorance, ask stupid questions, and generally try to become student-like. This is a humbling experience and a difficult one at best.

It is better to be a software engineer than a hacker...even in AI environments.

There is a basic myth among people that the words "AI programmer" and "Hacker" are synonomous. Despite such myths, we learned that it was absolutely necessary to use "good" programming practices throughout the development of the Tutor. Thus, the Expert System was changed five times in pursuit of just the "right" data structure. At each stage, we looked at the code and asked if it could be easily maintained, documented and understood. The final system fulfills all of these requirements.

Experts do not always make good teachers.

We initially assumed that if our expert knew how to solve the problem, then she would also know how to teach people how to solve problems. This is not necessarily true. One of the most important qualities of a good teacher is that they are able to reduce most complex problems to a series of very simple, clear procedures. Most good teachers have mastered the art of explaining even the most complex of ideas. They have also mastered the art of knowing what to say when students make mistakes. In short, a good teacher can make sense and order out of chaos. This particular quality is not always present in most experts. They may have performed a task or job because it "feels right", or "looks right." Also, they don't always know what to say to a student when they get the problem wrong. When this occurs, it is necessary to go find a teacher who can tell you how to teach.

Computer programmers, not educators, develop intelligent tutors.

There is a recurring theme in computer education literature that the teacher should be able to sit down at the computer and develop lessons, create interesting curriculum, and program a computer to interact with the student. This is what sold most people on the idea of Authoring Systems for Computer Based Education. It has also been proposed for the creation of Intelligent Tutoring Systems. It is a worthy dream. Yet, the reality of the situation is that programmers, not educators, develop courseware. This is true for traditional intelligent computer systems. Hopefully, this will change at some future time. At the moment, we are stuck with the fact that programmers, not educators, develop curriculum.

SUMMARY AND FUTURE DIRECTION

The current ground track microworld uses a qualitative model to teach the basic concepts of orbital mechanics. This microworld provides the student with a discovery environment which allows him to explore relationships between orbital parameters and ground tracks. The microworld also has intelligence. It knows about the domain, about how to estimate orbital parameters from a ground track, and about how to use the inquiry tools effectively to achieve goals. As a result, if the student fails to make satisfactory progress toward the stated goals, then the system intervenes and offers appropriate assistance. This type of intelligent simulation provides a more active and adaptive environment for reinforcing training skills.

C-2

The initial prototype is now complete and has been formatively evaluated by members of the NORAD crew and instructors at the Space School. The authors performed further tests during the Spring Semester of '87 with students from the Space School at Lowry Air Force Base to determine if the Tutor is more effective than traditional classroom experience. This data will also be used to improve the diagnostic portion of the tutor.

Several areas of research are currently being investigated. Because one of the primary purposes for developing this Tutor was to create a vehicle for testing hypotheses for training effectiveness, we want to investigate specific questions dealing with this area such as: What happens in an instructional environment when you vary the order of the State Module? (Is it better to state a hypothesis and then conduct experiments?) What happens in the instructional environment when you vary the order of remediation? (Tool use versus Skill Diagnosis?) Finally, how can the information we obtain from these studies be made a dynamic part of the system so that it can adapt to individual student's needs? These and other issues will be explored in the coming months and should contribute to our understanding of how to build more effective training systems.

ACKNOWLEDGEMENTS

The authors would also like to thank the instructors and students at the Unified Space Training School (UST) for their assistance with this project.

REFERENCES

Astronautics 332. USAF Academy, Colorado, 1985.

Bates, Roger, Mueller, Donald and White, E. Fundamentals of Astrodynamics. Dover Publications, New York, 1971.

Carr, B., Goldstein, Ira. Overlays: a theory of modelling for Computer Aided Instruction. MIT AI Memo 406 Memo 40, 1977.

Shute, Valerie, and Glasser, Robert. An Intelligent Tutoring System for Exploring Principles of Economics. In Press.

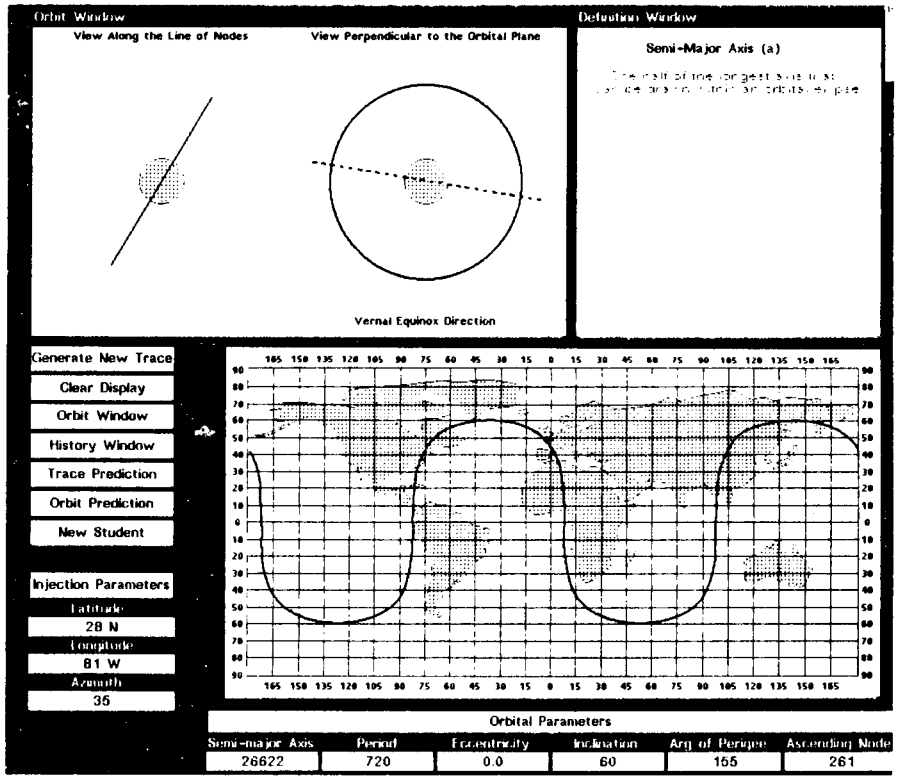

Figure 1: Generate Trace Window

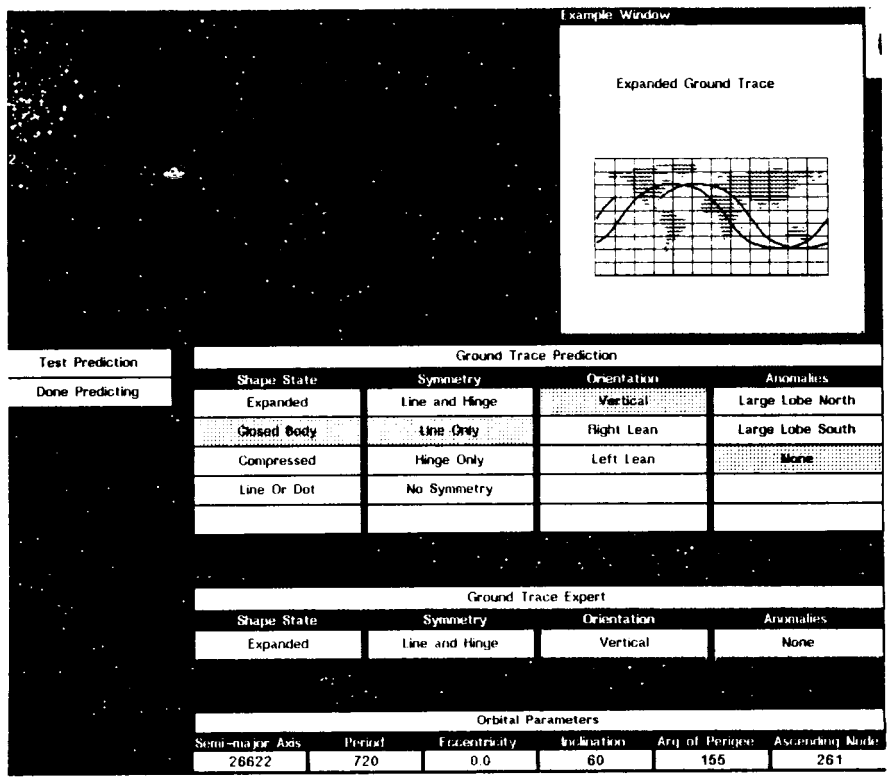

Figure 2: Trace Prediction

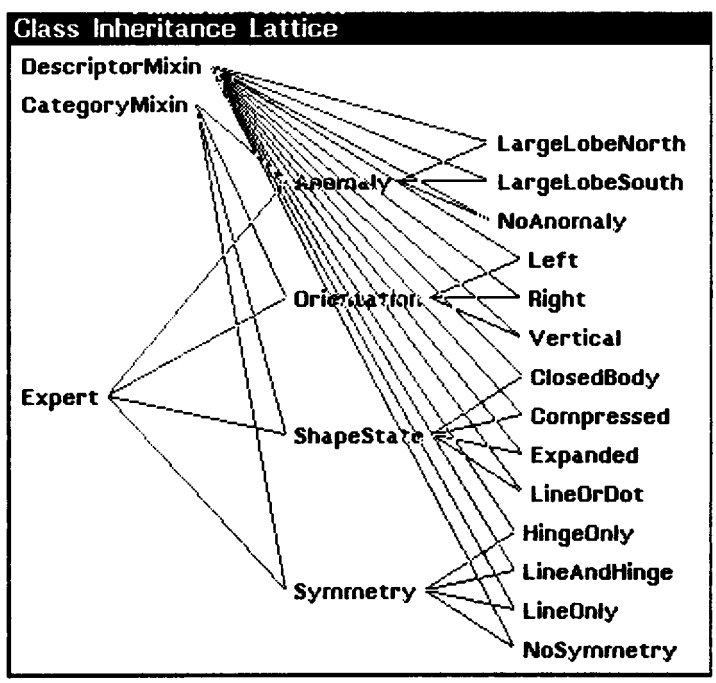

Figure 3: Shape Objects

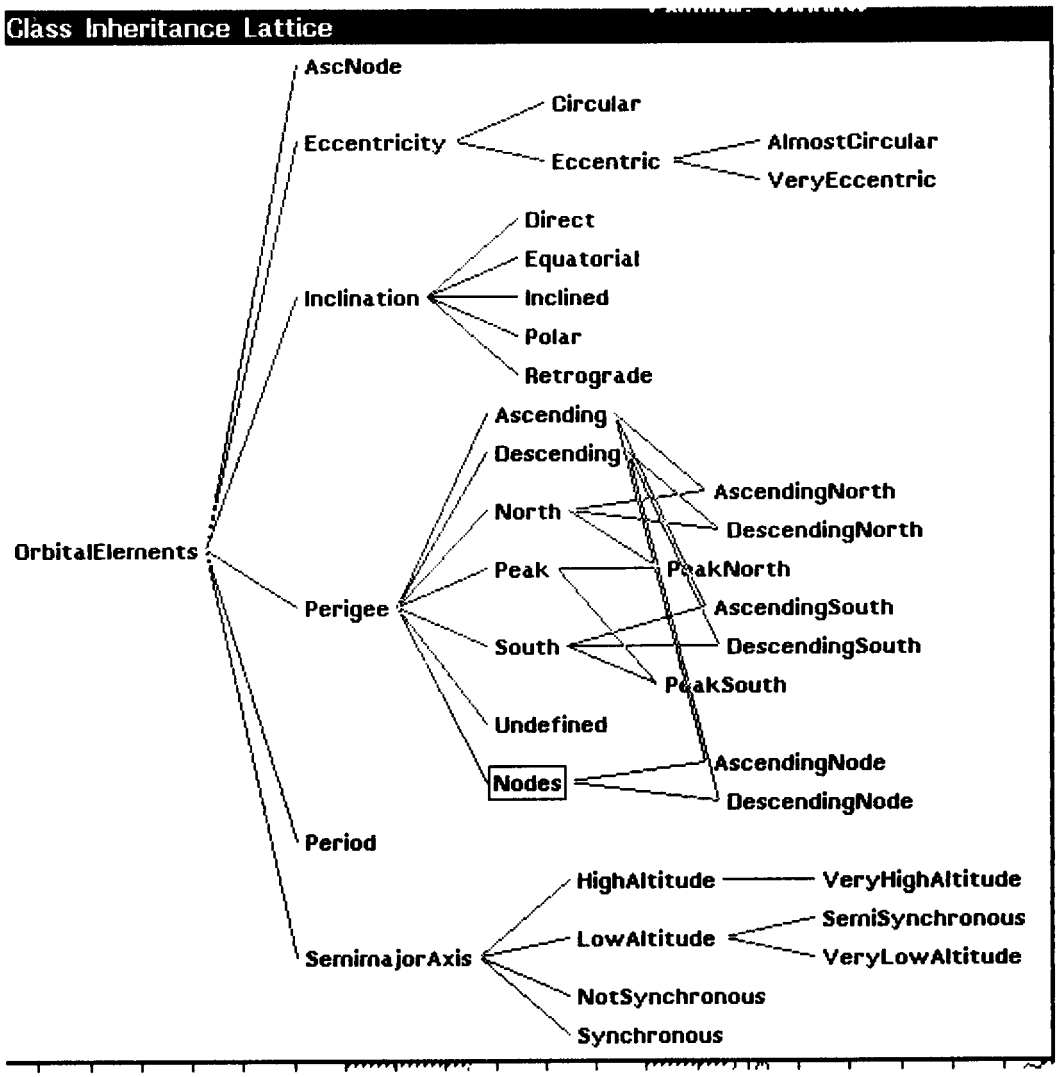

Figure 4: Orbit Objects