Artificial Intelligence Support for Scientific Model-building

Richard M. Keller
Artificial Intelligence Research Branch
MS 269-2
NASA Ames Research Center
Moffett Field, CA 94035-1000

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-92-29

August, 1992

To appear in Proc. 1992 AAAI Fall Symposium on Intelligent Scientific Computation

Artificial Intelligence Support for Scientific Model-building

Richard M. Keller

Sterling Software
Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 269-2, Moffett Field, CA 94035-1000
Keller@ptolemy.arc.nasa.gov

Abstract

Scientific model-building can be a time-intensive and painstaking process, often involving the development of large and complex computer programs. Despite the effort involved, scientific models cannot easily be distributed and shared with other scientists. In general, implemented scientific models are complex, idiosyncratic, and difficult for anyone but the original scientific development team to understand. We believe that artificial intelligence techniques can facilitate both the model-building and model-sharing process. In this paper, we overview our effort to build a scientific modeling software tool that aids the scientist in developing and using models. This tool includes an interactive intelligent graphical interface, a high-level domain-specific modeling language, a library of physics equations and experimental datasets, and a suite of data display facilities.

1. Introduction and Motivation: Software support for scientific model-building

Model-building is an integral part of the scientific enterprise. Scientists studying a particular phenomenon develop theories in order to account for novel observations and to make predictions about expected behavior. To validate their theories, scientists conduct *in situ* experiments whenever possible. Often, however, it is not possible to carry out direct experiments due to cost or other limiting factors. In these cases, scientists build models of the system under study and then test their theories against those models. Sometimes these models take the form of hardware (i.e., some sort of physical analog to the actual system), but often the models are expressed in software.

The construction of scientific software models can be a time-intensive and painstaking process. Many scientific models are written in terms of general-purpose numeric programming languages, such as FORTRAN, which have not been specially designed for the modeling task. Implementing a model can involve writing large and complex programs that access multiple datasets and utilize numerous different statistical and numeric processing packages. Development time for large scientific models may involve many months or years of effort.

For all the time and effort it takes to develop a model, the user community for most scientific models is limited to one -- the scientist who initially designed the model. This is not to say that one scientist has no use for another's models. On the contrary, model-sharing is highly desirable because it gives scientists the ability to "run experiments" and test their theories using different models without additional development overhead. The ability to easily inspect, use, and modify another scientist's models would be an extremely useful adjunct to current model-building practice, and an effective medium for communicating scientific ideas, as well. Just as scientists read technical papers describing theories, they should be able to inspect and exercise the software models that were used in validating those theories. Given the benefits to model sharing, why is it practiced so infrequently?

There are numerous technological barriers to scientific model sharing. Some of these barriers include:

- *Lack of comprehensibility*: Scientific software models are often sparsely commented and cryptic. Even a program that is initially well-written and commented will become fragmented over time.
- *Wrong level of abstraction:* A scientific model's structure is not obvious from the low-level program code. To understand an implemented scientific model, a scientist must examine the detailed code and attempt to infer high-level scientific content from low-level programming constructs.
- *Implicit assumptions:* Often important modeling assumptions are left implicit in the low-level code. These assumptions cannot be inspected or easily modified by new users.

Not surprisingly, some of these barriers are quite similar to those cited as discouraging conventional software sharing.

2. Objectives and Approach

Our primary research objective is to facilitate scientific model-construction and model-sharing by addressing the technological barriers described above. Although computer models play a crucial role in the conduct of science today, scientists lack adequate software engineering tools to facilitate the construction, maintenance, and reuse of modeling software. We are investigating the development of specialized software tools to ease the modeling process.

In particular, we believe that the following collection of advanced software techniques can substantially enhance the modeling process. We have begun to integrate these techniques in a *scientific modeling software tool* that serves as an aid to the scientist in developing and using models. The techniques include:

- *Interactive graphical interface*: To enhance comprehensibility and modifiability of models. Visual and iconic representations help the user to rapidly grasp the content of a model.
- *High-level modeling language*: To provide an appropriate level of abstraction for modeling and introduce natural domain concepts that are familiar to the scientist-user.
- *Analysis facilities*: To facilitate the interpretation of experimental results through the use of graphical plotting and statistical techniques.
- *Equation and dataset libraries*: To facilitate the sharing of standardized scientific equations and datasets.
- *Intelligent assistance*: To provide guidance and automate simple modeling steps. Artificial Intelligence-based techniques, such as constraint satisfaction, typed inheritance hierarchies, and backward-chaining control can reduce the amount of detail that the scientist-user needs to track.
- *Assumption maintenance facility*: To maintain explicit descriptions of modeling and data assumptions underlying a model and interdependencies among these assumptions.

At NASA Ames Research Center, we are building a knowledge-based software environment that employs the above techniques to make it easier for scientists to construct, modify, and share scientific models of physical systems. Examples of such models include planetary atmosphere models, ecosystem models, and biochemical process models. The SIGMA (Scientists' Intelligent Graphical Modeling Assistant) system functions as an intelligent assistant to the scientist. Rather than construct models using a conventional programming language, scientists can use SIGMA's graphical interface to "program" visually using a high-level data flow modeling language. The terms in this modeling language involve scientific concepts (e.g., physical quantities, scientific equations, and datasets) rather than general programming concepts (e.g., arrays, loops, counters). SIGMA assists the scientist during the model-building process and checks the model for consistency and coherency as it is being constructed. Then SIGMA automatically translates the conceptual model into an executable program, freeing the scientists from error-prone implementation details.

In order to provide this level of automation, the system must be given a significant amount of knowledge about the scientific domain, as well as general knowledge about programming. In our system, knowledge is represented and stored in a large knowledge base that contains information about scientific equations, physical quantities and constants, scientific units, numerical programming methods, and scientific domain concepts and relations. We have invested considerable time and energy into representing scientific knowledge in a form that is reusable across a variety of scientific disciplines.

In general, our approach to providing modeling assistance is extremely knowledge-intensive. We believe that our system must have extensive knowledge of the scientific problem under study in order to interact intelligently and synergistically with a scientist to create modeling software. Without this shared understanding, the system would have to rely on user guidance repeatedly during the model-building process. Tedious "hand holding" would likely increase user frustration and decrease the utility of the system to an unacceptable level. A beneficial side-effect of utilizing domain knowledge is SIGMA's ability to ensure the consistency of an evolving model and to catch errors during the model-building process.

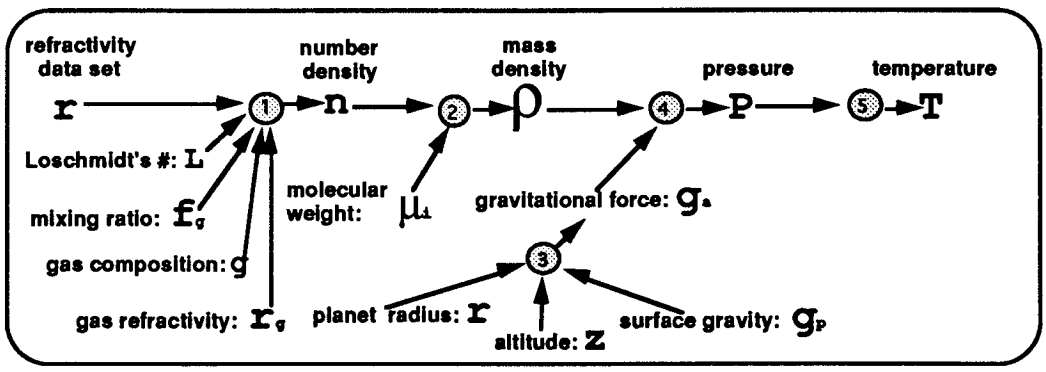
3. Example Modeling Task

To ground our work, we have been conducting a case study of a planetary atmosphere model developed (in FORTRAN) by planetary scientist and collaborator Christopher McKay. His model uses scientific data from Voyager I's flyby of Saturn's moon Titan to infer the thermal and radiative structure of Titan's atmosphere. Our methodology has been to "reverse-engineer" the Fortran model in order to reconstruct the knowledge and methods used by McKay to build such a model. Based on our analysis, we are developing a general tool that will ultimately enable atmospheric scientists to construct an entire class of atmospheric models -- including McKay's Titan model -- using our automated modeling assistant.

To avoid developing a system that is too narrowly scoped, we also have been studying models in other scientific domains. In this way, we increase confidence that our efforts are general and transferable to other scientific disciplines. For example, we are in the process of extending our knowledge base to support modeling activities associated with the Forest-BGC ecosystem model. This forest carbon-cycle and water-cycle model was developed by Drs. S. Running and J. Coughlin at the University of Montana, and is used by collaborator Jennifer Dungan and other researchers at NASA. Dungan's goal is to use SIGMA to build a generalized version of Forest-BGC that can be used to investigate regional-scale forest ecosystem phenomena.

As an illustration of the type of model that can be built with SIGMA, consider a fragment of McKay's Titan model shown as a data flow graph in Figure 1. The purpose of this fragment is to develop a profile of Titan's atmosphere that describes the pressure, temperature, and density of gases at various altitudes above its surface. The major source of relevant experimental data is the Voyager-I flyby of Titan back in November 1980. As Voyager-I reached the far side of Titan, it sent back radio signals that passed through Titan's atmosphere and then on to receiving stations on Earth. Due to the density of gases in the atmosphere, the radio waves were refracted slightly as they passed through the atmosphere, resulting in a diminished signal picked up on Earth. The amount of refraction was measured at different altitudes above the surface. This refractivity data serves as a starting point for inducing the desired atmospheric profile.

In Figure 1, the lettered nodes represent scientific quantities and the numbered nodes represent scientific equations. The model computes the temperature (T) at some altitude point above Titan's surface based on input refractivity data (r) from Voyager-I. In brief, the atmospheric profile is determined as follows (see Figure 1). First, for each atmospheric point profiled, the measured refractivity data (r) is used to compute the number-density (n) of the gases at that altitude. (The number-density of a mixture of gases is defined to be the number of molecules per volume of gas.) If the identity and relative percentages of gases in a



where:

① $n = \frac{r}{\Sigma_g f_g \frac{r_g}{L}}$

② $\rho = n \Sigma_i f_i \frac{\mu_i}{N0}$

③ $g_a = g_p \left(\frac{r}{r+z}\right)^2$

④ $\frac{dP}{dz} = -\rho g$

⑤ $T = \frac{P}{nk}$

Figure 1: Data flow graph representing fragment of a planetary atmospheric model. Letters represent physical quantities. Numbered circles correspond to equation application nodes.

mixture is known, the number-density can be computed as a function of refractivity using Equation 1. Next, using the molecular weight of the various gases in the mixture, the mass-density (ρ, or mass per volume of mixture) can be computed from the number-density using Equation 2. The hydrostatic law can then be used to determine the pressure (P) from the mass-density by numerically integrating the weight of the atmosphere above each profile point. Finally, the temperature (T) can be determined from the mass-density and the pressure by applying an equation of state, such as the ideal gas law (Equation 5).

This data flow graph represents a kind of high-level specification for a scientific model. The specification is expressed at the level of abstraction at which a physicist might describe the model, and is far more comprehensible and reusable than the corresponding FORTRAN code. The goal of the SIGMA project is to enable scientists to construct and manipulate models at this level of abstraction, thereby facilitating modification and reuse.

4. Model-building with SIGMA

In the current version of SIGMA, we conceptualize model-building as a kind of derivation process. Given some initial configuration state of the physical system being modelled, model-building can be viewed as a process of deriving a set of unknown physical variables from the known variables in that initial state. The initial state takes the form of a network structure of interrelated domain entities, each of which has an associated set of known and unknown quantities. The initial state reflects the scientist's understanding of the relevant entities and relations that are necessary to describe the physical system being modelled. The process of deriving unknown quantities is accomplished via a series of computational transformations that bridge the gap between the known and unknown variables. Conceptually, each transformation takes as input a set of variables and produces one or more variables as output. Two kinds of computational transformations currently supported in the system are equations (algebraic and ordinary differential) and Fortran subroutines.

The process of deriving unknown variables is accomplished via a simple backchaining procedure. In this backchaining process, the user first selects a target physical variable they wish to calculate. Then the system presents the user with a set of transformations that can be used to compute the desired variable. This set is determined based on the system's semantic understanding of the applicability of each transformation and its ability to match the applicability conditions to the current state. The user next selects one of the suggested transformations, and the system checks to see whether all the input variables required by this transformation are already known. If so, the desired output variable is computed using the selected transformation; if not, the backchaining process recurses for each of the unknown variables. During this process, the graphical interface displays a visualization of the current model as it is being built. The visualization takes the form of a data flow graph such as the one in Figure 1.

SIGMA is implemented in object-oriented CommonLisp on a Sun SPARCstation 2. We are currently implementing a Motif-based graphical user interface to support interaction with SIGMA. We intend to build up a complete visual programming environment in which scientific models can be constructed from equations graphically, just as components are wired together to build devices in a CAD/CAM tool.

Our longterm plan is to make SIGMA a viable tool to support scientists involved in future NASA planetary and earth-based activities, including those related to the upcoming Huygens probe of Titan's atmosphere, as well as the modeling activities associated with data collected by the Earth Observing System (EOS) network of satellites.

5. Acknowledgments

I would like to acknowledge the effort of the entire SIGMA team at NASA Ames, which includes Michal Rimon, Aseem Das, David Thompson, Michael Sims, Chris McKay, Jennifer Dungan, Caitlin Griffith, and Yaron Gold. Special thanks to Jennifer, who provided comments on an earlier draft of this paper.

This research is co-funded by NASA's Office of Space Science and Applications, and the Office of Aeronautics and Space Technology.

6. Bibliography

Richard M. Keller and Michal Rimon, "A Knowledge-based Software Development Environment for Scientific Model-building", submitted to *7th Knowledge-Based Software Engineering Conference (KBSE-92)*, Tysons Corner, VA, September 1992.

Richard M. Keller, "Knowledge-intensive Software Design: Can too much knowledge be a burden?", *Proc. AAAI-92 Workshop on Automating Software Design*, San Jose, CA, July 1992.

J. L. Dungan and R. Keller, "Development of an Advanced Software Tool for Ecosystem Simulation Modelling", Abstracts supplement of the *Bulletin of the Ecological Society of America* , 72(2) p.104, 1991.

Richard M Keller, "The Scientific Modeling Assistant: An Interactive Scientific Model-Building Tool", *Proc. AAAI-91 Workshop on Automating Software Design*, Anaheim, CA, July 1991.

R.M. Keller, M.H. Sims, E. Podolak, and C.P. McKay, "Constructing an Advanced Software Tool for Planetary Atmospheric Modeling", *Proc. i-SAIRAS'90* (International Symposium on Artificial Intelligence, Robotics and Automation in Space), Kobe, Japan, November 1990.