KNOWLEDGE ENGINEERING METHODS FOR CLIMATE MODELS

LADISLAV J. KOHOUT, ANDREAS STROTMANN, ROBERT A. VAN ENGELEN

Dept. of Computer Science & School of Computational Science and Information Technology
Florida State University, Tallahassee, Florida 32306-4530

1 Introduction

We are involved in the construction of problem solving environments for coupled climate modeling.

Numerical simulations of climate models that include atmospheric, oceanographic, geological, and other sub-models need to coordinate information along the boundaries between the sub-models in order to model correctly the interaction between the disparate sub-systems. Conventional high performance computing (HPC) programming paradigms make it difficult for users to integrate existing model components such as atmosphere, oceans, and chemistry because of the absence of uniform model development; "no mechanism for integrating research results exists."

This situation can be substantially remedied by the use of problem solving environments:

> "A Problem-Solving Environment (PSE) is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic and semi-automatic selection of solution methods, and ways to easily incorporate novel solution methods." (Houstis [1])

The application area that we considered above gives us an idea as to the components that a PSE of this kind is currently envisioned to have.

2 Some PSE Components

A central component of a Problem Solving Environment of this kind is the code generator, or model compiler. Its input is a rather complex high-level symbolic specification of the simulation, and it uses a database of symbolic representations of numerical algorithms and their properties to compositionally transform the input specification into high-level program code, e.g. High-Performance Fortran (HPF).

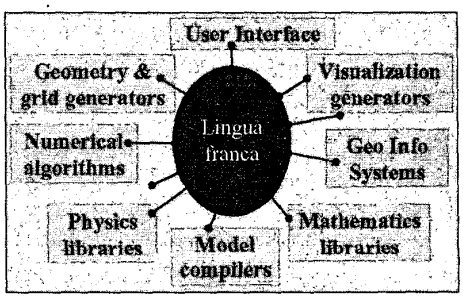

Fig. 1: Components of a Climate PSE

The code generator would typically submit its results (again, symbolically) to a set of special-purpose optimizers for parallel, vector, or super-scalar machines, whose Fortran output would go to a Fortran compiler to produce executable simulation codes [2].

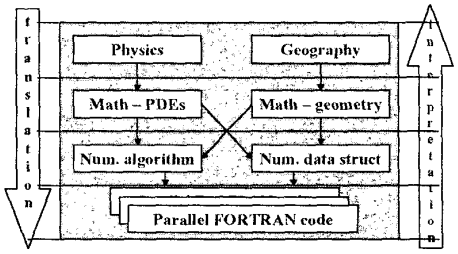

Fig. 2: Implementing a Climate Model

Another component envisaged for a PSE for climate studies is a set of geo-information systems (GISs) that offer information required for a coupled climate model (e.g., elevation data, vegetation data, ocean bottom data, seismic and volcanic activity distribution data, ice cap data, river systems data, or meteorological data). GISs would be queried to provide input to model

0-7803-7087-2/01/$10.00 © 2001 IEEE

simulations that depend on discretization algorithms (e.g. triangulation or voxel domain decomposition) or for grid generation, or they might be involved in analyzing or visualizing the outcomes of model runs.

The PSE would typically provide access to a Computer Algebra (CA) back-end. When a scientist wants to modify the mathematical model used in the simulation, she would have to go and change some equations, and could use a CA system to transform the altered specification into a form recognized by the code generator and required by the algorithm.

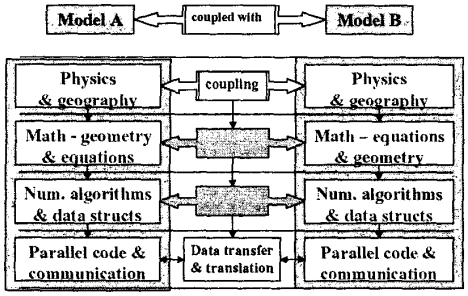

Fig. 3: Implementing a coupled climate model

The numerics people in this group are looking at re-designing numerical algorithms for different components of the coordinated climate models in such a way that the boundary interactions between the models is handled properly. Indeed, the method of choice for getting from a time-line of inherently incomplete and fuzzy observation data to a quick, robust, good climate model may be completely different from the approach taken by current climatologists. Fuzzy techniques could be useful, too, in doing quality control or assurance for the communication channels between the components of the PSE [5].

All this puts a high demand on the forms of communication among the PSE components. Clearly, knowledge, information, and data from the following fields are involved in the communication patterns of a cooperative PSE of this kind:

- mathematics (calculus, functional analysis, complex variables, n-dimensional analysis, numerical analysis, discrete math, discretization),
- algorithms (parallel/distributed cooperating processes),
- computer architecture and networks (communication patterns, cost analyses, optimization),
- symbolic transformations (code differentiation, integration, PDE analyses such as Lie symmetry recognition),
- geometric modeling (finite elements),
- geo-sciences (geography, oceanography, geology, meteorology (GIS)),
- biology, biophysics, chemistry (chemical transformations and interactions), physics,
- interdisciplinary information (e.g. heat absorption by chemical molecules, which combines physical and chemical concepts with implications in meteorology and/or oceanography and bio-systems),
- bibliographical material (which experiments provide input data, where do you find the theoretical background of the model).

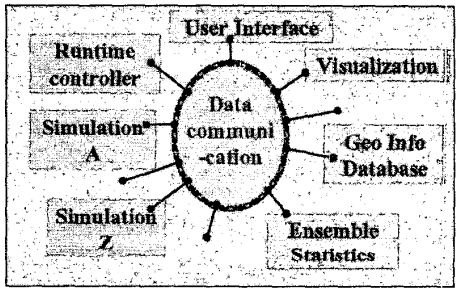

Fig. 4: Runtime data communication in a simulation generated by a Climate PSE

In addition, numerical data will flow between the visualization tools and the simulation codes generated by the model compiler.

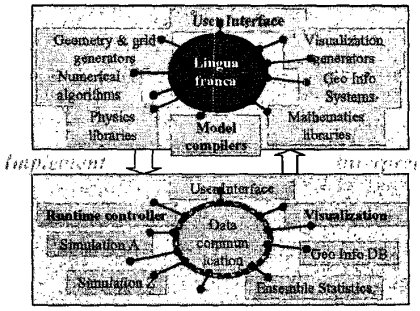

Fig. 5: Compile time and runtime components of a Climate Modeling PSE

This data should, however, be accompanied by generous amounts of symbolic meta-data determining the concrete meaning of each numerical datum. Furthermore, conversion between field measurements, scientific data format standards, and semantically oriented representations will need to be designed and implemented.

3 The structure of climate modeling

As we have seen, the coupling of climate models poses interesting new problems to the working climatologist. Typically, discretizations (grids and time steps) do not match across a boundary, and require translation via interpolation or integration. Frequently, to make things worse, computations actually do not take place in a simple time/space grid but rather in a model-specific problem space, with depth in ocean models, for example, represented as constant adjusted pressure surfaces rather than distance from the surface, say.

Moreover, the translation has to be chosen in such a way that it will introduce as little numerical error as possible into either of the two sub-models that it connects. Worse, even an optimally faithful translation in this sense may cause numerical instability of the coupled model through "resonance" along the common boundary.

In a nutshell: writing a numerical model of a single aspect of world climate is already a difficult task even for those who are experienced researchers in this field, but devising a climate model that couples multiple sub-models is a particularly non-trivial task in the traditional HPC setting.

Thus, the illustration in Fig. 4 is actually quite unrealistic in the traditional HPC setting. The data communication backbone of such a structure poses a daunting obstacle.

If we want to overcome this obstacle, we first need to understand why that is so. And for this, we need to interview the working climatologist how he actually writes a numerical model of aspects of the climate.

The climatologist will then likely go and give a seminar[1] on developing a climate model "*ab initio*." Starting with the physics of the situation he wants to model, he will explain how certain differential equations need to be transformed in a long series of steps until one arrives at a dimensionless mathematical set of differential equations that provide a mathematical model that describes in sufficient detail the physics of the problem at hand. Along the way, analysis of scales will help "ignore" lower-order terms, often replacing them by "parameters." The mathematical model is then transformed into a numerical one, based on decades of experience gained in the trade on such things as numerical stability, ease of coding, and overall speed of different types of solution methods.

Fig. 2 summarizes in a highly schematic way this highly complex cognitive process for the development of a single coherent climate model: the climatologist intuitively works at the top-most levels of semantics and pragmatics, while the number-crunching HPC machine works at the lowest level of data and, perhaps, information.

One of us (van Engelen) has taken such an approach and shown that it is possible to take a formal mathematical specification of a climate model along with grid pattern information and estimates of machine performance parameters, and to automatically transform these into highly efficient code [2, 3].

Now, the crux of the problem of coupled climate models is that each sub-model is practically certain to be developed along a very different path down from the levels of world-knowledge and the natural sciences to the levels of information and data (see Fig. 6). Any attempt at taking the two resulting numeric codes and coupling the two models at their own level via simple data transfer between the component sub-model codes is therefore doomed to fail.

Instead, the development of a coupled model needs to follow the same route as that of a single model, as sketched in Fig. 3. The coupling between the sub-models needs to be understood as a physical coupling between physical sub-spaces, and the numerical coupling as an implementation of this coupling that is developed via yet another parallel path down through all the levels of abstraction.

4 Knowledge engineering issues

The process of synthesizing a program that is to be run on a high-performance computer is a complicated one. In coupled models, as we have seen, knowledge domains of several disciplines are intricately interwoven. In the case of climate models, for example, atmospheric, ocean, and

[1] In our case, W.K. Dewar kindly gave one to the authors.

land models interact as a minimum, with ocean ice, glacial, or biosphere models adding verisimilitude and complexity. Each discipline contributes some concepts that are cast into a mathematical form, usually and most commonly that of mathematical equations, but frequently also into the form of a heuristic algorithm.

Such equations in their full complexity rarely have exact analytical solutions. This necessitates the use of algorithms for numerical solutions that are necessarily just approximations. As we shall see later, these, however, are not the only kinds of approximations that enter physical models. The numerical methods employed determine to a large extent the final form of mathematical formulas (algorithms) that are converted into executable code.

Performance issues are crucial for getting simulation results within a timeframe that is constrained by the available computational resources (in case of predictions that have to be available at certain time, this issue become even more important). For that reason, the know-how about the numerics of solving equations has to be supplemented by information on computer architectures (more specifically their possible configurations) on which the final (completed) program will be run.

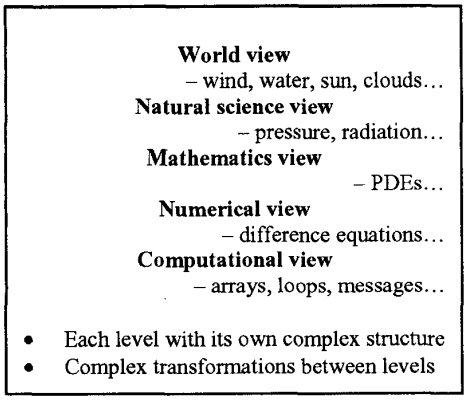

Fig. 6: Model levels of abstraction

Each mathematical variable has some conceptual representation; furthermore, variables of a particular mathematical type can have different conceptual meanings (interpretations) attached to them, i.e. different knowledge domains. In terms of knowledge engineering and computational intelligence this involves several kinds of ontologies. In OpenMath terms, specific content dictionaries for each knowledge domain are required.

Complete, exact and exhaustive combination of all entities of all ontologies involved in a particular model would be extremely complex. Such excessive complexity would preclude any practical use of such a model in actual computations. The problem of combining different conceptual systems and ontologies of different knowledge domains, however, goes much deeper. It is not just the issue of complexity that is involved. When combining concepts and ontologies of different domains one cannot rely on their complete logical compatibility. There may be inherent incompatibilities and conflicts present. What one can hope for at best is *compatibility* of restricted conceptual domains and combination of ontologies with further constraints imposed.

In addition to this logical question of compatibility of meaning, the compatibility of different physical scales is involved as well. One mathematical type of equation may be used to model many different phenomena. A particular equation of fluid dynamics may for example be used for meteorological atmospheric effects, using a certain specific range of parameters and a particular conceptual interpretation of all the variables. In a model of an ocean, a different conceptual interpretation and different range of parameters are imposed on the same kind of fluid dynamics equation; to this the structure of couplings has to be added, to form a unified coupled model of atmosphere and ocean in interaction.

When studying different aspects of the physics of the atmosphere and of the oceans, different scales may be involved. Each measurement scale represents a certain metric imposed at a different, what system scientists would call, resolution level of a systemic variable. Because different problems involve different scales and some variables and parameters may have values negligible at certain scales, approximation is involved. This kind of approximation is epistemologically different from approximations necessitated by numerical methods, yet these different kinds of approximations may or may not be logically independent.

The steps of modeling described above are usually done intuitively by geo-scientists. They often dismiss any systematic description of what is involved as philosophy, when in fact this belongs to applications of general systems

methodology when done rigorously, by exploring systems concepts, appropriate mathematics and logics, often taking algebraic form. When, however, a computer is used as a tool that helps to automate some of the activities of producing highly optimized code for HPC, such framework has to be provided explicitly, so that correct code can be produced.

5 Knowledge elicitation

The algebraic specification of physical models and theories at higher levels of abstraction is well established in quantum physics. By contrast, rank and file members of the fluid dynamics community are more concerned with the mathematical specialization of fluid dynamics equations, fitting them to specific problems to be solved, and with the mathematical formulation of adequate numerical methods for correct and efficient approximate numerical solutions of such equations. The conceptual stages that precede the production of a mathematical and numerical formulation of a geophysical problem are done purely intuitively by practicing scientists.

So the question arises as to how to capture in a computer-representable form at least a part of this intuitive process. Fortunately, knowledge engineering techniques of exploratory knowledge elicitation can help. There are also available various methods of computational intelligence that can help with precise formulation of ontologies. In order to explore these in a practical way in distributed computations, mathematical knowledge *communication* schemes such as MathML with its extension mechanism OpenMath, as well as mathematical knowledge *manipulation* systems such as computer algebra, formal specification, or automated theorem proving systems have to be used.

We performed a preliminary analysis of the cognitive structure of a recent paper [4] on the development of a simple coupled ocean-atmosphere climate model, with the goal of estimating the amount of effort that would need to be expended on developing a computer-understandable top-level ontology for the field. Both the vocabulary used in the text and the notations used in formulas were extracted and classified.

Not surprisingly, both exhibited a highly complex structure. In the paper's text, modifiers were used extensively to express the several dimensions, layers of abstraction, models, components, and other ways of structuring the knowledge domain(s). In the formulas, some of the particularly interesting higher-order aspects of the representation of mathematical knowledge in content communication language were illustrated in the paper, such as the use of ad-hoc generalized quantifiers or of an ad-hoc notation for currying a special function.

When we discussed our analysis with the author of that paper, he pointed out that the paper only contains a fraction of the conceptual domains that are actually involved in the development of a full coupled climate model, namely those required for the final steps in the model development. Entire knowledge domains are hidden behind some terminology used in passing as a linguistic reference for the fellow expert.

6 Knowledge communication

This experiment points out two features. First, knowledge communication for PSE components requires a sophisticated lingua franca; higher-order concepts abound in this field of applications, and are likely to be just as frequent in any other scientific or engineering field.

Second, the knowledge field is too large for it to be possible to define a vocabulary that covers the whole of the field from the start. Therefore, realistically, such a vocabulary would need to grow over time as the tools in the PSE are added to cover more and more sub-models and their conceptual spaces.

7 Conclusions

For both these reasons, we conclude that a flexible and extensible language for knowledge communication would be a necessary prerequisite for building a problem solving environment for building coupled climate models. Especially in the case of higher-order concepts that need to be expressible in such a language, proper design criteria such as the compositionality of such a language play a crucial role. Currently, as we have argued in [6], OpenMath is to our knowledge the only such language that meets the compositionality requirement, and that is thus qualified to be used as the base on which to build the lingua franca for knowledge communication in a climatology PSE.

We also conclude that knowledge engineering techniques, and in particular exploratory knowledge elicitation techniques [7] combined with higher-order concept analysis, are necessary tools in the building of such a language, and thus ultimately in the building of a coupled climate model problem solving environment. Fuzzy relational knowledge representation plays an important role in higher-order concept analysis [8],[9],[10].

Acknowledgement.

This is a contribution of the Climate Institute, a Center of Excellence supported by the Florida State University Foundation.

References

1. E. Gallopoulos, E. Houstis, and J.R. Rice, Computer as Thinker/Doer: Problem-Solving Environments for Computational Science, IEEE Computational Science and Engineering, Vol. 1, pp. 11-23, 1994.
2. R.A. van Engelen, L. Wolters, and G. Cats, Tomorrow's Weather Forecast: Automatic Code Generation for Atmospheric Modeling, IEEE Computational Science & Engineering, Vol.4, No. 3, pp. 22-31, 1997.
3. R.A. van Engelen, L. Wolters, G. Cats, Ctadel: A Generator of Multi-Platform High-Performance Codes for PDE-Based Scientific Applications, in proceedings of the 10th ACM International Conference on Supercomputing, pp. 86-93, 1996.
4. W.K. Dewar: Quasigeostrophic Climate Dynamics. Submitted to the Journal of Marine Research, 1999.
5. Kohout, L.J. and Bandler, W. The use of fuzzy information retrieval techniques in construction of multi-centre knowledge-based systems. In: Uncertainty in Knowledge-Based Systems (Lecture Notes in Computer Science vol. 286) Bouchon, B. and Yager, R.R. (eds.), Springer Verlag, Berlin, pp.257-264, 1987.
6. Strotmann, A. and Kohout, L.J., OpenMath: Compositionality achieved at last. ACM SIGSAM Bulletin, Special issue on OpenMath, Mike Dewar, ed., December 2000.
7. Kohout, L.J. and Anderson, J. and Bandler, W. et al., Knowledge-Based Systems for Multiple Environments. Ashgate Publ. (Gower), Aldershot, U.K., 1990.
8. Kohout, L.J. and Bandler, W., Use of fuzzy relations in knowledge representation, acquisition and processing. In: Fuzzy Logic for the Management of Uncertainty, Zadeh, L.A. and Kacprzyk, J. (eds.), John Wiley, New York, pp. 415-435, 1992.
9. Kohout, L.J., and Kim, E., Reasoning with cognitive structures of agents I: Acquisition of rules for computational theory of perceptions by fuzzy relational products. In: Fuzzy IF-THEN Rules in Computational Intelligence, DaRuan and Kerre, E. (eds.), Kluwer, Boston, pp. 161-188, 2000.
10. Kohout, L.J., Boolean and fuzzy relations. In: The Encyclopedia of Optimization, Pardalos, P.M. and Floudas, C.A. (eds.), Kluwer, Boston, in press.