ELSEVIER

The Science of the Total Environment 183 (1996) 25–31

the Science of the Total Environment
An International Journal for Scientific Research into the Environment and its Relationship with Man

Model transformation rules and model disaggregation

Robert I. Muetzelfeldt*[a], Ruth D. Yanai[b]

[a]*The University of Edinburgh, Institute of Ecology and Resource Management, Darwin Building, King's Buildings, Edinburgh, EH9 3JU, Scotland, UK*

[b]*SUNY College of Environmental Science and Forestry, Syracuse, NY 13210, USA*

Abstract

Ecosystem models vary considerably in their degree of disaggregation: the extent to which they 'lump' or 'split' model components. However, the criteria for the degree of disaggregation in a model are frequently not clear. The development of such criteria would be enhanced if the degree of disaggregation in a model could be easily changed, permitting a ready comparison of the alternative versions. In this paper, a framework is proposed for representing model transformation rules. Each rule indicates how a particular model component or set of components can be replaced by more- or less-disaggregated components. These rules have the potential to automate the process of generating alternative versions of a model differing in degree of disaggregation, and provide a framework within which modellers can express their expert opinion on the legality, costs and benefits of particular lumping or splitting decisions. The approach is wholly dependent on the symbolic representation of model structure, since each transformation rule is in effect a symbolic re-write rule, expressing how the set of symbols defining one model can be replaced by another set defining an alternative model. It is proposed that the logic programming language Prolog is suitable both for representing model structure in symbolic form, and for representing the model transformation rules.

Keywords: Ecosystem modelling; Model disaggregation; Prolog

1. Introduction

Modelling ecosystem behaviour is essential for predicting the response of ecosystems to previously unobserved changes in environmental conditions or management regimes.

Ecosystem models cover a broad range in the extent to which they 'lump' or 'split' ecosystem parts and processes. At one extreme are lumped empirical models such as yield tables and unit hydrographs. These models succeed in describing desired ecosystem responses but, because they are empirical, have no grounds for extrapolating predictions beyond the conditions under which they were derived. At the other extreme are models that attempt to include all the processes internal to the ecosystem that contribute to the modelled behaviour. These models tend to contain large numbers of parameters that are difficult to obtain, making model predictions difficult to evaluate. Note that, although there is a tendency for lumped models to be empirical and have a small number of parameters, and for split models to be process-based with many parameters, this is not a logical necessity. There are lumped, process-based models, and lumped models with many parameters; and there are split empirical models, and split models with few parameters.

* Corresponding author.

SSDI 0048-9697(95)04975-7

One purpose of the workshop for which this paper was prepared was to question the assumption that it is necessary to increase the level of detail in a model to improve its predictive success. Although some modellers tend to be 'splitters' and some to be 'lumpers', there do not appear to be any generally accepted criteria for deciding on the level of detail to be included in a particular model developed to address a specific problem.

The aim of this paper is to propose a framework within which such criteria can be expressed, and which will permit the automatic generation of a model which is more or less aggregated than an alternative model. In this approach, the process of lumping or splitting a model component is viewed as a transformation of one model into the other, governed by a set of transformation rules. These rules specify a set of legal transformations in model structure. The concept of transformation rules has been applied in architecture, in which a building is designed by taking some initial design and applying rules that specify legal additions, changes and deletions of rooms to produce new designs (Coyne, 1988). This approach offers the potential of developing computer-based tools for automatically exploring degrees of disaggregation in models. It could also encourage the modelling community to develop a consensus on the disaggregation process.

Initially other approaches to the problem of model aggregation are considered. Then, a classification of disaggregation methods is proposed. This is followed by an introduction to the declarative, symbolic representation of model structure, because the model transformation approach requires such a representation. The general case of model transformation is then given, followed by a simple example. Finally, the way in which particular model transformation rules can be associated with the implications of applying the transformation, in relation to model objectives, is considered.

2. Existing approaches to exploring model aggregation

Studies of the desired level of aggregation/disaggregation in ecosystem models fall into three main categories. First, some studies compare a set of models, which happen to differ in degree of disaggregation as well as in other respects. Second, there have been studies into the effect of aggregating one part of a disaggregated model. The third approach involves the theoretical analysis of certain types of mathematically-tractable models.

There are specific case studies comparing models that differ in degree of disaggregation. For example, Rose et al. (1991a) compare the predictions of three catchment acidification models. To compare the different models systematically required specifying a mapping procedure for the hundreds of input variables used by the models (Rose et al., 1991b). In a less controlled comparison of models (van Heerden and Yanai, in press), the authors of 16 forest-soil-atmosphere models were given input files containing site-specific soil and climatic data for the Solling test site. These models were used to predict a number of key variables, such as forest growth, water use, and soil nutrient concentrations. The conclusion was that complex models did not perform better than simple models, and that differences in model performance were probably more related to parameterisation than to model structure. Case studies such as these, based as they are on a comparison of existing models, fail to isolate the question of model disaggregation, because the models to be compared differ in many respects, not just in the degree of disaggregation of well-defined components.

A better method for isolating the question of model disaggregation is to lump or split one component. For example, Gardner et al. (1982) generated 40 alternative population models, exemplifying four types of aggregation and a variety of process equations. They demonstrated the degree of error introduced by aggregating components with different turnover rates. Bartell et al. (1988) compared the behaviour of a model of daily production by 100 algal populations with three aggregated versions of the same system, each one aggregating the populations into 10 population groups using different criteria. They found that the way the aggregation was undertaken had a significant impact on the performance of the aggregated model compared to the original. Hutson and

Wagenet (1993) took a complex model of pesticide leaching and compared it with a simplified version, differing in the representation of water flow. The simpler model was found to be sufficiently accurate for field-scale simulations, where data are limited and variability is high. Studies of this type, comparing models that differ in defined aspects, are rarely undertaken, possibly due in part to the programming effort required to generate strictly comparable models. Further, disaggregation requires a number of decisions to be made, such as which parameters should be split, and the correspondence between the value of the lumped parameter and those of the split parameters. This makes it difficult to attribute the resulting differences in model performance to the specific disaggregation process. Finally, it would be difficult to generalise the results beyond the context of the specific model.

Theoretical analyses have been undertaken of the aggregation problem (Iwasa et al., 1987), and can identify the conditions that a model must satisfy for aggregation to be appropriate. However, these analyses are restricted to certain classes of model, and do not provide a general solution to the problem of deciding how aggregation or disaggregation should be undertaken.

3. Types of disaggregation

The following classification of types of disaggregation is proposed, according to two criteria (Muetzelfeldt and Sinclair, 1993).

The first describes whether the number of elements is fixed during the course of a simulation.

(1) In static disaggregation, the number of elements used to represent a component is fixed during the course of the simulation. Examples include a grid-based representation of space; a fixed number of soil layers; animal age-classes; and a population represented as a fixed number of individual animals.

(2) In dynamic disaggregation, the number of elements varies during the course of a simulation run. Examples include individual-based models of population dynamics, and dynamic models of tree branching patterns or root growth.

The second criterion describes the relation of the disaggregated parts to the original part (Uschold, 1990).

(1) Component-part disaggregation involves the representation of a model object by the (usually dissimilar) parts that make it up. Thus, the object 'forest' in a lumped model might be represented as the objects 'foliage', 'stem', 'branch' and 'root' in a split version of the same model.

(2) Sub-class disaggregation involves splitting a model object into a set of miniature versions of itself. Thus, a population can be split into a number of sub-populations (on the basis of sex, age, or location); or 'soil' can be split into a number of soil layers.

(3) Individual-based disaggregation involves representing a population by the set of individuals that constitute it.

4. Declarative representation of model structure

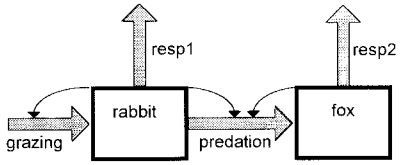
The usual method for representing the structure of a model is though the use of text, equations, diagrams and program listings on paper, and executable programs in a procedural language on the computer. The former is used for communication of model structure, the latter for solution of the model. This means that the only operation the computer can do with the model is to run it: other operations on the model are not possible, and in particular one cannot apply the model transformation approach proposed here.

An alternative is to represent the model declaratively rather than procedurally: as a symbolic specification of model structure, rather than as an executable computer program. Muetzelfeldt et al. (1989) illustrate this approach in the context of System Dynamics modelling, and suggest that the logic programming language Prolog is particularly well-suited to this task. This method provides a single representation that can be used both to convey information on the model (by interrogating the database or by producing model summaries) and to produce executable versions of the model. It also opens up many possibilities for model analysis: for example, to detect if the model violates rules of model construction.

Once a model is represented in this way, it can be transformed in various ways, by replacing some

of the symbols defining model structure with others. For example, one type of transformation could involve replacing one function with another. Another type of transformation is to vary the degree of disaggregation in the model: it is this approach that is proposed in this paper.

The approach is illustrated here with models that fit the System Dynamics modelling paradigm. This paradigm is chosen for illustrative purposes because it is widely known and understood, and it involves a small number of simple model elements. Fig. 1 shows a declarative representation (in Prolog) of the compartment-flow-influence structure of a simple System Dynamics model of a predator-prey system. In the Prolog representation, three predicates are used to represent the three model elements on the diagram: the predicate 'compartment' to define a compartment; 'flow' to define a flow between two compartments; and 'influence' to indicate that one element in the model is used in calculating a value for another. For simplicity, the Prolog representation has been restricted to the elements shown in the model diagram: however, further details of the model — for example, the equations used to calculate individual flows — can be included in the same way. The interested reader will find a more detailed description of this approach in Muetzelfeldt et al. (1989).




```
compartment(rabbit, biomass).
compartment(fox, biomass).
flow(grazing, outside, rabbit, biomass).
flow(predation, rabbit, fox, biomass).
flow(resp1, rabbit, outside, biomass).
flow(resp2, fox, outside, biomass).
influence(rabbit:biomass, grazing).
influence(rabbit:biomass, predation).
influence(fox:biomass, predation).
```

Fig. 1. Declarative representation in Prolog of a simple, two-compartment predator-prey model.

5. Using transformation rules to effect changes in model disaggregation

5.1. The general case

Each transformation rule is represented as a Prolog clause with the following form:

```
rule([a4,a7], [b5,b9,b11]).
```

which we read as: 'If you can find the terms a4 and a7 in the declarative description of model structure, then remove them and add the terms b5, b9 and b11'.

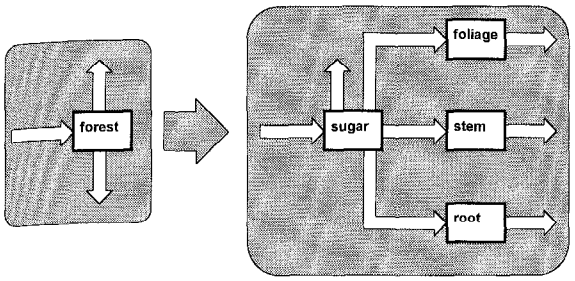
The word 'rule' is the name chosen for the predicate that is used to represent the transformation rules. (This predicate name has no built-in meaning in Prolog: it is chosen merely for its readability.) This predicate has two arguments. The first is a list of the model elements that must be in the existing model specification, and will be deleted from the model specification by application of the rule. There can be any number of terms represented here by a4 and a7 and they can occur anywhere, in any order, in the model specification. The second argument is another list, giving the model elements that will be added to the model specification by application of the rule. There can also be any number of these.

This general approach is illustrated with an example.

5.2. Example: component-part disaggregation

In this example, the disaggregation is represented by an increase in the number of compartments in the model. A simple model of forest carbon dynamics might represent the total carbon contents of the forest with a single compartment. An alternative, less aggregated view of the forest is to represent the carbon content of the major carbon stores: sugar, foliage, stem and root, perhaps. This transformation can be represented both diagrammatically and in Prolog (Fig. 2).

The diagram is interpreted as a rule that shows how a System Dynamics diagram is redrawn to show the change in model structure. It is important to realise that this diagrammatic representation of the transformation rule does not show the whole model, but just the part affected by the transformation. Thus, this rule could be applied to a much bigger System Dynamics model, most of




```
rule( [compartment(forest,carbon),
       flow(phsyn, outside, forest, carbon),
       flow(respiration, forest, outside, carbon),
       flow(litter, forest, outside, carbon)],

      [compartment([sugar,foliage,stem,root], carbon),
       flow(phsyn, outside, foliage, carbon),
       flow(respiration, sugar, outside, carbon),
       flow(allocation, sugar, [foliage,stem,root], carbon),
       flow(litter, [foliage,stem,root], outside, carbon)]).
```

Fig. 2. Diagrammatic and Prolog representations of a simple rule for disaggregating a one-compartment forest component into four compartments.

which would remain unchanged by the application of this transformation.

The corresponding Prolog rule shows that the rule will be applied when the model specification contains the specified compartment ('forest') containing carbon, and the three flows. On application of the rule, these model elements will be removed, and replaced by the four compartments and eight flows shown in the second argument of the rule.

5.3. Extensions of the basic transformation rule

The power of this approach can be readily extended. For example, a model element in the left-hand side of the rule can also appear in the right-hand side: i.e. it is removed then replaced. This is one way of specifying that the transformation rule should apply only to those models that contain certain elements.

The same effect can be achieved by making the transformation rule itself conditional on some aspect of the model specification. This is easy to represent in Prolog, because a Prolog rule (which is different from the term 'transformation rule' used in this paper) would simply be used to express the condition. Thus, if the transformation rule given in Fig. 2 were to be applied if the model contained a water compartment, the rule could be modified to read:

```
rule([...], [...]):-
  model(M),
  member(compartment(soil,water),M).
```

The... represent the model elements given in Fig. 2. The :- symbol is read as the word 'if' in English. The letter M stands for the list of all the elements in the model description; and the 'member' predicate checks to see if the element 'compartment(soil,water)' is in the list M of model elements.

A third type of extension is to use variables instead of actual names of model elements in the transformation rules. Thus, a transformation rule that lumped two predator species into one predator species group, regardless of the actual names of the predators, might be required. This could be implemented by using Prolog variables in the transformation rule, and using the Prolog rule mechanism to check that the species referred to are actually predators.

6. The implications of lumping/splitting decisions

The decision 'to lump' or 'to split' is taken in order to obtain a model which is in some sense better than the alternative. The criteria used for evaluating a model are often implicit rather than explicit, but include:

- required accuracy;
- cost of model implementation;
- execution speed;
- data availability;
- generalisability;
- clarity;
- use of existing knowledge;
- cost of determining relationships.

Any model transformation has cost/benefit implications in terms of these criteria. The benefits point to positive reasons for applying the transfor-

mation: for example, increased accuracy, reduced run time. The costs suggests reasons for not applying the transformation: for example, decreased accuracy, increased run time. The modeller needs to make a decision, based on changes across all these criteria, as to whether a particular transformation is desirable. The decision will depend on the relative weighting for these criteria that applies in a particular context: for some, computing resources may be poor, so that execution speed is the limiting factor counting against a particular transformation; for others, accuracy may be of overriding importance.

The 'transformation rule' approach permits the implications of a particular transformation to be associated intimately with the transformation. This can be achieved by adding an extra term to the rule, being a list of all the criteria affected by the application of that transformation, and the change in each criterion resulting from the application of the transformation. For example, the application of the disaggregation transformation in the forest example could be considered to: potentially increase model accuracy; increase model implementation time; increase model execution time; and increase the data requirements.

Problems remain with the realisation of this approach. First, the implications of applying a disaggregation transformation do not just depend on whether that transformation is applied, but also on the extent of the disaggregation. For example, the implications for execution speed of applying a spatial disaggregation depend on the number of spatial units specified, not just on the fact that there will be spatial units. Second, model refinement through the application of transformation rules could involve the use of many such rules, not just one. It is therefore necessary to devise methods for propagating their implications through multiple transformations. For example, one transformation may decrease data requirements while another may increase them: what is the net effect? Treating model design as the application of a series of transformation rules makes it possible to state such questions in a general form; solving them will permit a balanced evaluation of alternative model designs.

7. Conclusions

1. Model transformation rules can provide a formal and rigorous basis for investigating the appropriate level of lumping or splitting in ecological models, for two reasons:

— They force the modelling community to express formally the precise ramifications of a particular lumping or splitting operation on model structure.

— They enable the modelling community to discuss, in a focused way, the costs and benefits of lumping or splitting operations.

2. A library of model transformation rules would greatly simplify the task of generating lumped and split models of the same system. This would make 'structural sensitivity analysis' (studying the effect of changing model disaggregation on model behaviour) the norm rather than the exception.

3. It is a necessary precondition of this approach that a language is developed for the declarative representation of model structure. This language must be capable of handling the great diversity of modelling approaches used in the ecological and environmental sciences.

References

Bartell, S.M., W.G. Cale, R.V. O'Neill and R.H. Gardner, 1988. Aggregation error: research objectives and relevant model structure. Ecol. Modelling, 41: 157–168.

Coyne, R., 1988. Logic Models of Design. Pitman, London, 317 pp.

Gardner, R.H., W.G. Cale and R.V. O'Neill, 1982. Robust analysis of aggregation error. Ecology, 63: 1771–1779.

Hutson, J.L. and R.J. Wagenet, 1993. A pragmatic field-scale approach for modelling pesticides. J. Environ. Qual., 22: 494.

Iwasa, Y., V. Andreasen and S. Levin, 1987. Aggregation in model ecosystems. I. Perfect aggregation. Ecol. Modelling, 37: 287–302.

Muetzelfeldt, R.I. and F.L. Sinclair, 1993. Ecological modelling of agroforestry systems. Agrofor. Abstr., 6: 207–247.

Muetzelfeldt, R.I., D. Robertson, A. Bundy and M. Uschold, 1989. The use of Prolog for improving the rigour and accessibility of ecological modelling. Ecol. Modelling, 46: 9–34.

Rose, K.A., A.L. Brenkert, R.B. Cook and R.H. Gardner, 1991a. Systematic comparison of ILWAS, MAGIC and

ETD watershed acidification models. 2. Monte Carlo analysis under regional variability. Water Resour. Res., 27: 2591–2603.

Rose, K.A., R.B. Cook, A.L. Brenkert and R.H. Gardner, 1991b. Systematic comparison of ILWAS, MAGIC and ETD watershed acidification models. 1. Mapping among model inputs and deterministic results. Water Resour. Res., 27: 2477–2589.

Uschold, M., 1990. The Use of Typed Lambda Calculus for Comprehension and Construction of Simulation Models in the Domain of Ecology. Unpublished Ph.D. thesis, Department of Artificial Intelligence, The University of Edinburgh, UK.

van Heerden, C. and R.D. Yanai, (in press). Effects of stresses on forest growth in models applied to the Solling spruce site. Ecol. Modelling.