Multi–Level Cooperative Dialogue in Intelligent Front Ends

Dave Robertson
Department of Artificial Intelligence,
University of Edinburgh.
6^{th} May 1990

Abstract

The control of multi–level cooperative dialogue is considered a difficult problem in knowledge based systems but may be easier to tackle in the more restricted domain of Intelligent Front Ends. This paper discusses the requirement for multi–level cooperative dialogue in IFE programs and describes several techniques for supporting this form of interaction, using examples from implemented systems.

Acknowledgements

The work described in this paper was part of the ECO project, funded by SERC Grant GR/E/00730. I would like to thank the other participants in this project (Alan Bundy, Bob Muetzelfeldt, Mike Uschold and Mandy Haggith) for their contributions to this research.

1 Introduction

The nature of the dialogue conducted by an Intelligent Front End (IFE) is a crucial factor in determining whether the IFE will be accepted by the users for whom it has been designed. A common feature of most IFE dialogues is that they are cooperative, in the sense that both the IFE program and its users contribute during the problem solving process. Ensuring that cooperative dialogue can proceed effectively is known to be a hard problem for knowledge based systems in general. A major cause of this problem is the requirement to link the system's dialogue to the basic properties of the problem which it is meant to solve. Alison Kidd sums up this problem in the context of diagnostic systems ([Kidd 85]):

> ....our study of naturally occurring expert consultations in the range of diagnostic task domains has demonstrated the importance of the expert and user negotiating over the suitability of a remedy to clear a particular fault. However, the sophisticated

> dialogue and goal inference processes necessary to support adequately such a complex negotiation task will not be achieved for some time yet....In the meantime, I propose that some of the users' advice requirements might be met by an expert system which represents and makes use of detailed knowledge about functional classes of remedies and about the pros and cons of alternative remedies within these classes.

The requirement which Kidd and many others perceive is for an explicit representation of the fundamental principles on which a solution must be based. This, it is argued, provides a sound basis for cooperative dialogue because it allows users to interact with the system at different levels of generality. At a high level, they may talk about general aspects of the problem which is to be solved. However, if the high level diagnosis is inadequate (perhaps because of lack of information on the part of the system) it is possible to resort to lower levels of dialogue at which the fundamental components of the problem are teased out. A key reason why this form of dialogue is difficult to provide for most applications of knowledge based systems is that the fundamental principles governing the solution to a problem are usually difficult to supply. For example, in a medical diagnostic system it in difficult to decide with certainty upon the fundamental mechanisms which operate within the domain. It is upon this point that IFE systems have a distinct advantage over knowledge based systems in general. The purpose of an IFE system is normally to supply coherent input for some existing software package. This means that the basic requirements for the solution provided by the IFE system are known with a high degree of precision. Thus, while the problem of multi–level cooperative dialogue resists attack in general, it should be more easy to tackle in IFE research.

In this paper, I shall describe several general techniques which have been used to conduct cooperative dialogue in IFE systems. These techniques have been used during the ECO project in the Department of Artificial Intelligence at the University of Edinburgh. The ECO project was directed at a single problem – that of enabling users with little experience of ecological modelling to construct simulation models of ecological systems – and examples from this domain will be used to illustrate the discussion. Those readers who wish to find out more about the ECO project and related work are referred to [Robertson *et al.* 91]. As a preliminary to the main discussion I shall first consider the conditions under which multi-level dialogue becomes necessary in IFE systems.

2 The Conditions Under which Multi–Level Dialogue is Required

An IFE system exists in order to supply appropriate instructions to some application package. These instructions, plus the constraints imposed on them by the mechanisms of the application, constitute the basic operations towards which all dialogue is ultimately directed. It is normal for the IFE, instead of attempting to obtain directly the instructions for the application, to construct a problem description from which the appropriate instructions may be extracted. The crucial points about this scheme are:

1. The problem description allows users to describe formally the task which they want to perform without having to input this description directly to the application package. This allows the dialogue necessary to describe the problem to be separated from constraints imposed by the application package.

2. The knowledge representation used for the problem description must be in a form sufficiently close to the those concepts with which an intended user is familiar in order to allow them to be presented to the user in a form which he/she can understand readily. Ideally, we would like a 1–to–1 mapping between the concepts with which users naturally describe their problem and the elements of the problem description.

3. The knowledge representation used for the problem description must be sufficiently close to that of the application package to allow the problem description to be translated into expressions required by the application. Ideally, this translation should be automatic because the problem description is intended to provide all the information required to run the application.

4. From 2 and 3 we can see that there is an inherent tension between the demands of the user on the problem description (which call for natural dialogue) and the demands of the application package (which requires technical instructions). One way of reducing this tension is to subdivide the problem description into a number of levels, with "lower" levels being close to the requirements of the application package and "higher" levels containing information with which the user is familiar.

For some applications it is possible to provide a neat analogy between the forms of instruction required by the application package and concepts with which the intended users are familiar. In these circumstances it is possible to dispense with multi–layered dialogue and simply provide users with a set of primitive "building blocks" with which to construct a problem description. These primitives are often displayed graphically, allowing them to be connected together in prescribed ways. We shall refer to these IFE's as direct manipulation systems.

A good example of a direct manipulation system is STELLA ([Lewis 86]) which allows users to construct simulation models in a System Dynamics notation ([Forrester 61]). The computational framework of System Dynamics can be thought of in terms of tanks (representing state variables) with flows between them (providing the dynamic behaviour of state variables). The rates of flow are determined by equations which, in turn, receive as input parameters and state variables. The problem description consists of a diagram in which tanks, flows, equations and parameters are represented using graphics objects. The user is allowed to connect these objects in ways which are consistent with the System Dynamics notation - for example, graphics operations are provided for connecting two tanks with a flow. Users of STELLA are familiar with the idea of flows of material between tanks and this analogy, between flows of material and changes to state variables, is used to help users understand what they are doing when they construct a description of a simulation model.

The problem with this sort of direct manipulation system is that the users of the system must have the sort of problems which suit the analogy on which the system is founded. If not, they will be forced to adapt their views of the problem to fit the requirements of the problem description. This adaptation may not be easy. For example, STELLA is not a good system in which to describe simulations involving movement of objects over some surface because it is difficult (though not impossible) to envisage spatial movement in terms of tanks and flows. The need for access to higher levels of problem description (thus creating a multi–layered dialogue) occurs when the instructions required by the application package become too complex to be manipulated directly by the intended users. In these cases the tension between requirements 2 and 3 (above) becomes a real problem. This problem has been discussed at some length in [Robertson *et al.* 88c]. In this paper I shall describe some of the solutions which were attempted during the course of the ECO project.

3 Dividing High and Low Level Dialogue into Phases

An obvious way of tackling the problem of integrating high and low level dialogue is to separate the interaction with the user into two distinct phases: the first phase covering the high level dialogue and the second phase taking care of the residual low level details. In this scheme, the information obtained

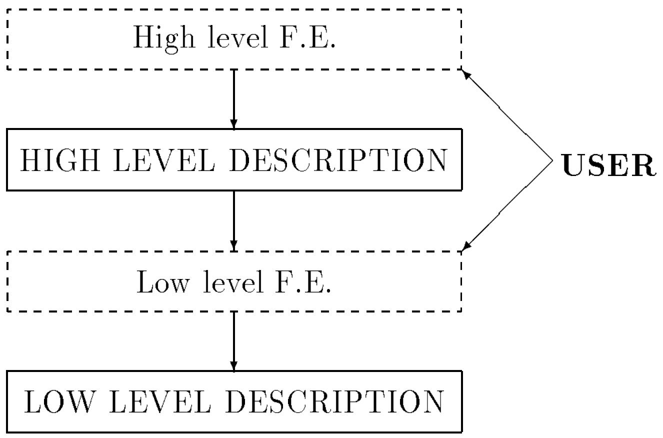

Figure 1: Phased high level and low level dialogue

in the high level phase provides a way of constraining the scope of the dialogue in the low level phase, thus making it possible for users to cope with the (otherwise overwhelming) low level details. We really have two IFE systems combined within this framework: the first controlling the high level dialogue and the second handling low level dialogue, as shown in Figure 1.

This plan formed the basis of the EL system ([Robertson *et al.* 88a]). In EL, the language used for the high level problem description is an order sorted logic (see [Cohn 89] for an introduction to this variety of logic). The basic building blocks of the high level description are sorted logic statements with which (after they have been rendered into English phrases using a Definite Clause Grammar) a user is expected to be familiar. For example, the statement:

$$\exists A \in attribute\ \forall X \in animal\ varies\_with(A, X, location, X) \qquad (1)$$

is used to denote that there is some attribute, A, of every animal, X, which varies with the location of X. Notice that this statement does not have the status of something which must be true in all ecological problems. It is intended as a *template*, to be selected by a user and restricted to describe his/her specific problem. For instance Statement 1 (above) might be restricted to produce:

$$\forall X \in deer\ varies\_with(mass, X, location, X) \qquad (2)$$

which denotes that the mass of all deer varies with their location. Statement 2 was obtained by restricting the values for X to range over sorts of deer and instantiating the variable, A, in Statement 1 to the specific attribute of mass.

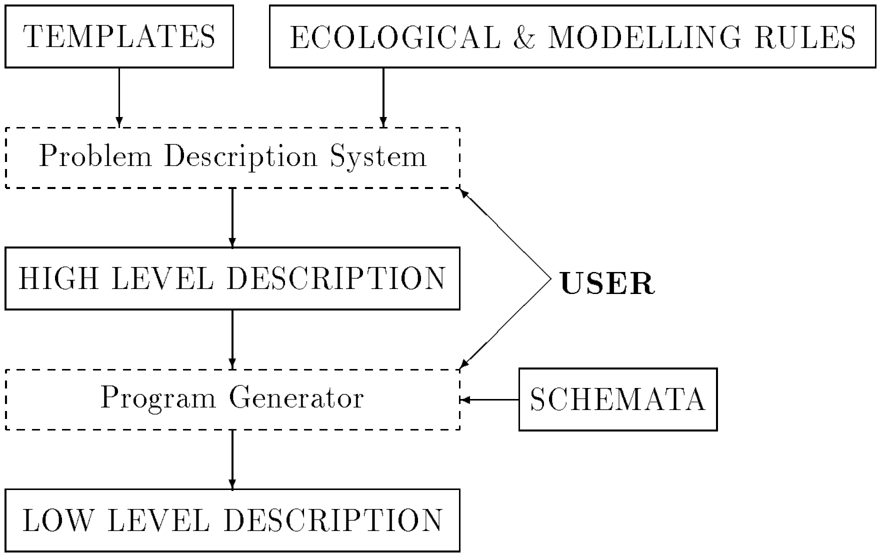

Figure 2: A plan of the EL system

The system is provided with a hierarchy of commonly used sorts of ecological object, which the user can adapt if necessary. Knowledge of these classes and subsort relationships between them allows EL to check that the restrictions imposed by the user on any statement are consistent with the existing sort restrictions on the statement. For example, it would be an error to attempt to restrict the values of the variable, A, in Statement 1 to include animals because attributes and animals are disjoint sorts in the sort hierarchy. In addition, a collection of ecological and modelling rules is used to prompt the user for further high level information (see Section 4 for more on this process).

When they are satisfied with their high level problem descriptions users then move on to the phase of low level dialogue. The purpose of this dialogue is to construct a Prolog program by applying sequences of standard program structures, which we refer to as *schemata*. The mechanism for schema application is described in [Robertson *et al.* 88b] and [Robertson *et al.* 91] so there is no need to describe it in detail. The key point is that a very large number of sequences of schema application are possible and the information contained in the high level description is used to restrict a user's attention to only those schemata which are appropriate to the problem which he/she is trying to solve. A diagram depicting the main components of EL appears in Figure 2.

Although EL does permit high level and low level dialogue to be meshed together it has several deficiencies:

- The phases of high and low level dialogue are kept strictly separate. Therefore, users cannot interpose sections of low level information within the high level dialogue.

- It is not always clear when enough high level information has been obtained to allow the user to complete the dialogue entirely in low level mode.

- The course of the high level dialogue is determined purely with reference to the high level information (*e.g.* ecological rules) which is available to the system. There is no mechanism for allowing low level information (about the developing simulation model) to influence the collection of high level information.

In Sections 4 and 5 we shall consider ways in which these problems have been addressed in EL and related systems.

4 Incorporating Low Level Information into High Level Dialogue

A partial solution to the problem of phased high and low level dialogue is to transplant some of the low level information to higher levels. This leaves the general architecture (shown in Figure 2) unchanged but requires some extra sophistication of the high level dialogue mechanism because the objects and relationships being described during the high level dialogue are often different from those at the low level. This is the case in the ecological modelling domain, where the high level description is primarily concerned with ecological relationships while the low level description contains an idealised version of these relationships which may be run as a simulation program. As part of the idealisation process the objects referred to in the high level description may re-appear as different objects (with different properties) in the low level description. For example, we might have a simple high level description consisting of the statement that the deer feed on all the trees – expressed formally in the sorted logic as:

$$\forall A \in deer \; \forall B \in trees \; feeds\_on(A, B) \tag{3}$$

This might be converted (after the application of appropriate schemata) into the low level equation[1]:

$$mass(deer, T) = mass(deer, T-1) + \mathcal{F}(mass(trees, T-1)) \tag{4}$$

[1]This would actually be expressed in Prolog but we have spared the reader the extra notation.

where the mass of deer is calculated from its previous mass plus some function, $\mathcal{F}$, of the previous mass of trees. The important point made by this toy example is that the nature of the names *deer* and *trees* are fundamentally different in the high and low level descriptions. At the high level they are both sorts, referring to some sets of deer and trees, while at the low level they have become objects referring to particular constants in the simulation model. This sort of aggregation is common practice among ecological modellers who must simplify parts of an ecological problem in order to make it computationally tractable.

The problem which arises when moving information from the low level to the high level is that of preventing the low and high level dialogue becoming confused. In the EL system this problem was ameliorated by distinguishing high level ecological rules form low level modelling rules. However, it was also necessary to adapt the inference mechanisms which were used at the high level. To demonstrate why this was so, consider the following example.

It was found useful to provide, as part of the high level dialogue mechanism, the facility to provide users with suggestions for statements they might want to add to the problem description, based on what they had already said. This was done by employing an abduction mechanism ([Cox & Pietrzykowski 86]). For instance, the ecological rule base might contain the implication:

$$\begin{aligned} &\forall A, B \in attribute\ \forall X \in animal\ varies\_with(A, X, B, X) \rightarrow \\ &\qquad has\_attribute(A, X)\ \&\ has\_attribute(B, X)\ \&\ \neg A = B \end{aligned} \tag{5}$$

which denotes that if any attribute an object varies with some other attribute of that object then the object possesses both attributes. This rule is useful because it allows the system to suggest the possibility that one attribute of some object might vary with another of its attributes, given that the user has told the system that the object possesses each attribute. Thus if EL had in its knowledge base the statements:

$$\forall X \in deer\ has\_attribute(mass, X) \tag{6}$$

$$\forall X \in deer\ has\_attribute(location, X) \tag{7}$$

it would suggest to the user the more useful statement:

$$\forall X \in deer\ varies\_with(mass, X, location, X) \tag{8}$$

Suppose that the user accepts statement 8 and adds it to the problem description. It would be useful to be able to use this new information to enable the system to suggest preliminary low level details of model structure. In particular, it would be useful to be able to suggest various forms of spatial representation for the model because if some attribute varies with location it is likely that the model will contain some spatial component. An example of a general form of spatial representation in a simulation model is to use grid squares – where the spatial area is divided into a grid and each object in the model is allocated to a cell in that grid. Therefore, we would like to be able to suggest to the user at this point that the model might contain a grid square spatial representation. The question is how to provide rules which will allow this suggestion to be made. The most obvious option would be to include the rule:

$$\otimes \quad \forall A \in attribute\ \forall X \in animal\ varies\_with(A, X, location, X) \rightarrow \\ spatial\_representation(X, grid\_squares) \tag{9}$$

Unfortunately, this rule is incorrect[2] because (if we take the standard logical interpretation of the implication symbol) it allows us to infer a grid square representation *whenever* the attribute of some animal varies with location. This isn't always true because it would be quite reasonable to have some other form of spatial representation (e.g. coordinate points). If we switch the direction of the implication, with a view to applying abduction, then we obtain the statement:

$$\otimes \quad \forall A \in attribute\ \forall X \in animal\ spatial\_representation(X, grid\_squares) \rightarrow \\ varies\_with(A, X, location, X) \tag{10}$$

This is also logically incorrect because it isn't true that a model with a grid square representation for some object, X, would always imply that X had some attribute which varied with location. It could be that some other object, Y, had an attribute which varied with location and the grid representation for X was merely a side–effect of the decisions made for Y.

Clearly, it is not possible to obtain the suggestion of grid squares by a simple "one step" inference. Therefore, EL must use a combination of deductive and abductive inference and this requires the following two rules:

$$\forall A \in attribute\ \forall X \in animal\ varies\_with(A, X, location, X) \rightarrow \\ spatially\_represented(X) \tag{11}$$

$$\forall X \in animal\ spatial\_representation(X, grid\_squares) \rightarrow \\ spatially\_represented(X) \tag{12}$$

This allows EL to deduce, from statements 8 and 11 that deer are spatially represented:

$$\forall X \in deer\ spatially\_represented(X) \tag{13}$$

[2]I have used the $\otimes$ symbol to flag invalid statements.

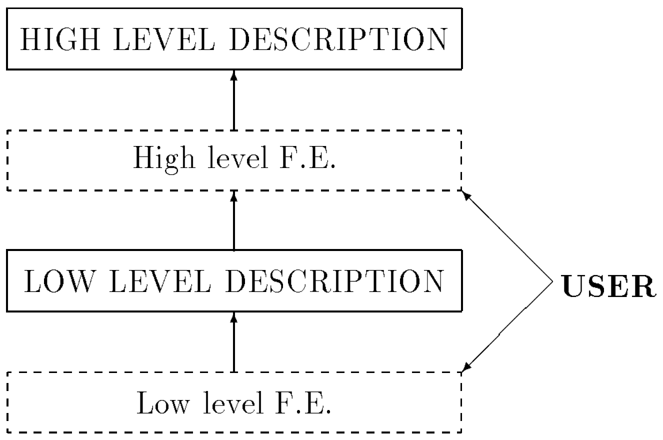

Figure 3: Controlling high level dialogue from low level requirements

By applying the abduction mechanism, EL can then use statements 12 and 13 to suggest to the user that deer have a grid square spatial representation:

$$\forall X \in deer\ spatial\_representation(X, grid\_squares) \tag{14}$$

The moral of this story is that low level information cannot simply be "shifted" into higher levels of dialogue without carefully considering the inference mechanisms necessary to bridge between the two different descriptions. However, there is a more radical approach to merging low and high level dialogue which we consider in the next section.

5 Driving High Level Dialogue from Low Level Information

In general it is unsatisfactory to have separate phases of high and low level dialogue. The main reason for this is that it is easy for the high level dialogue to get "out of step" with the requirements at the lower level (see Section 3). One way of ensuring that the high level dialogue is always pertinent to the requirements of the application is to drive all dialogue (including high level) from the low level front end. This is the reverse of the approach taken in Sections 3 and 4 and is illustrated in Figure 3.

The NIPPIE program ([Haggith 90]) used the basic low level schema application mechanism from EL to control dialogue at both high and low levels. This basic mechanism did not, itself, require much alteration. The key difference between EL's schema application mechanism and NIPPIE's version is that in EL the information needed to apply each schema was assumed to have already been supplied from earlier high level dialogue. In NIPPIE this assumption was relaxed and techniques were provided for establishing information needed at the low level by resuming dialogue at the high level. Therefore, in NIPPIE high level dialogue was constrained by the requirements of the application and so the mismatch problem with which we began this section was solved. Unfortunately the NIPPIE solution also has weaknesses:

- In order to ask "user–friendly" questions at the high level which will satisfy requests for information at the low level, NIPPIE must be able decide on which high level statements each low level request will depend. Recall that in Section 4 the connection between ecological and modelling knowledge was shown to be complex, involving the combined application of both deductive and abductive inference. This makes it difficult to decide, for a given piece of low level information, which will be the correct statements to look for at the high level in order to confirm its appropriateness for the simulation model. In NIPPIE this problem is finnessed by maintaining "dependency" relations between high and low level information. This allows fairly standard goal–directed search techniques to be used to find the high level questions appropriate for particular low level requests for information.

- The schema application algorithm relies upon a standard depth-first search strategy. Thus, the order in which pieces of information become necessary at the low level is unlikely always to match the sequence in which users consider it natural to provide information. To tackle this problem, NIPPIE has the ability to delay requests to the user for information until a winning request for information could be obtained from a set of options. A number of dialogue ordering heuristics are used to order the set of options and select the winner. These heuristics include:

  - Asking general questions before specific ones.
  - Asking related questions together.
  - Following up general questions with more detailed questions.

  This process is described in greater detail in [Haggith 90].

- The NIPPIE system is strictly monotonic. Once a piece of information has been added to the low level description there is no way of revising this information in the light of new information from the user. This problem is less apparent in the (also monotonic) EL system because the low level details aren't worked out until the high level description has been completed. If NIPPIE were to be given the facility to revise its low level description during a session this would raise all of the "housekeeping" problems associated with truth maintenance systems. Added to these complications would be the need to revise the high level description in order to keep it in phase with the low level revisions. In other IFE systems where the application package is run during the construction of low level description this problem may be insuperable because there would be no easy way of revising the changes which had taken place deep within the application.

6 Conclusion

This paper has described some programs which have been used to conduct multi–level dialogue in the domain of ecological modelling. More than one level of dialogue is required in this domain because the final application program is an interpreter for running completed simulation programs, while the users of the system are conversant with ecology but know little of simulation modelling. Thus, there is a large conceptual gap between the users of the system and the application package. To help bridge this gap it is useful to have formal representations of both the users' (high level) descriptions and the application's (low level) requirements. The problem addressed by this paper is how to manage the dialogue at both high and low levels in a way which users can understand.

One possibility is to separate the dialogue into high and low level phases (Section 3). This has the advantage of keeping high and low levels distinct, thus reducing non–monotonicity problems, but raises the problem of ensuring that high level dialogue doesn't stray beyond the confines of the low level requirements of the simulation model. This problem can be patched by transplanting some of the low level information to the high level (Section 4), provided that the inference mechanisms used in high level dialogue can be adapted to suit the new information. However, this is not a complete solution because some of the information is still relegated to the second phase of low level dialogue. A complete solution to this problem demands that high level dialogue should be based on the modelling requirements at the low level (Section 5). Unfortunately, this solution creates new problems because the structure of dialogue at a low level may require considerable transformation in order to convert it into a form suitable for consumption by users at the high level. These new problems have been tackled in the ECO project but the solutions which we have found are not general. More work is required in order to understand the nature of multi–level dialogue and to provide computational techniques with which it may be controlled.

References

[Cohn 89] A.G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3(2):89–129, 1989.

[Cox & Pietrzykowski 86] P.T. Cox and T. Pietrzykowski. Causes for events: Their computation and applications. In J. Siekmann, editor, *Lecture Notes in Computer Science: Proceedings of the 8th International Conference on Automated Deduction*, pages 608–621. Springer-Verlag, 1986.

[Forrester 61] J. W. Forrester. *Industrial Dynamics*. MIT Press, 1961.

[Haggith 90] M. Haggith. Interactive program construction. In S.L.H. Clarke, editor, *Proceedings of UK IT 90*, pages 227–33, Brighton, England, 1990.

[Kidd 85] A.L. Kidd. What do users ask ? some thoughts on diagnostic advice. In *Expert Systems 85*, pages 9–19, Coventry, England, 1985. British Computer Society Specialist Group on Expert Systems.

[Lewis 86] J. Lewis. Stella: A model of its kind. *Practical Computing*, pages 66–67, September 1986.

[Robertson *et al.* 88a] D. Robertson, A. Bundy, R. Muetzelfeldt, M. Haggith, and M Uschold. Using ecological descriptions to guide the construction of simulation programs. In *Proceedings of Alvey Annual Conference/UK IT 88*, Swansea, Wales, 1988. Also available as D.A.I. Research paper 381.

[Robertson *et al.* 88b] D. Robertson, A. Bundy, M. Uschold, and R. Muetzelfeldt. The ecologic system. Technical Report 1, Department of Artificial Intelligence, University of Edinburgh, 1988.

[Robertson *et al.* 88c] D. Robertson, M. Uschold, A. Bundy, and R. Muetzelfeldt. The ECO program construction system: Ways of increasing its representational power and their effects on the user interface. *International Journal of Man Machine Studies*, 31:1–26, 1988.

[Robertson *et al.* 91] D. Robertson, A. Bundy, R. Muetzelfeldt, M. Haggith, and M Uschold. *Eco-Logic: Logic-Based Approaches to Ecological Modelling.* MIT Press (Logic Programming Series), 1991. ISBN 0-262-18143-6.