Supporting Conceptual Knowledge Capture Through Automatic Modelling

Jochem Liem, Hylke Buisman, and Bert Bredeweg
Email: {J.Liem,B.Bredeweg}@uva.nl, hbuisman@gmail.com

Human Computer Studies Laboratory, University of Amsterdam, The Netherlands.

Abstract. This paper proposes an automated modelling algorithm that, based on a observations, creates a qualitative conceptual model that provides a causal explanation of system behaviour. The algorithm is envisioned to become a integral part of our model building methodology, and is meant to supports model builders.

1 Introduction

Within the NaturNet-Redime EU project, ecologists made qualitative models about river restoration ecology issues in several regions. A modelling methodology was developed to support these domain experts [1]. This structured approach describes how the expert's conceptual ideas can be formalised, and gradually refined into a complete model implementation. Modellers were able to follow the methodology, however frequent support from the modelling experts was required to overcome modelling issues.

This paper proposes an automatic modelling algorithm that learns qualitative conceptual models based on (a qualitative description of) observations of the system's behaviour. The algorithm is envisioned to be an integral part of the modelling methodology. As such, it takes some of the intermediate modelling results created in the modelling framework as input. The result is a qualitative conceptual model that provides a *causal explanation* of the system's behaviour.

Besides supporting modellers in articulating their conceptual ideas, there is the question whether it is possible to automatically discover causal explanations of system behaviour. Several researchers are trying to learn process models based on the behaviour of the system [2, 3], however their approaches typically learn differential equations and focus more on *numerical accuracy*. In contrast, our approach attempts to learn *causal explanations*, such as those found in e.g. Qualitative Process Theory (QPT) [4].

2 Garp3

The qualitative modelling and simulation workbench Garp3 (http://www.garp3.org) allows modellers to build conceptual models about systems. These models particularly focus on the cause-effect relationships (*influences* and *proportionalities*). Figure 1 shows a representation of a population with the quantities *Number of* and *Birth* and their quantity spaces. Quantity spaces indicate the possible magnitude (e.g. $M_v(\textit{Number of}) \in \{\textit{Zero}, \textit{Small}, \textit{Medium}, \textit{Large}\}$) and derivative values ($D_v(Q_1) \in \{-, 0, +\}$) of each quantity. We use $M_s(Q_1)$ and $D_s(Q_1)$ to refer to magnitude and derivative signs.

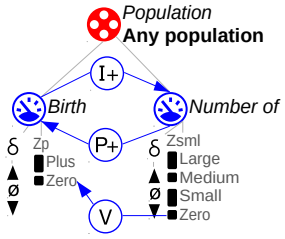


Fig. 1. Model fragment modelling the birth process.

The figure shows a positive influence (*Birth* $\overset{I+}{\rightarrow}$ *Number of*). This influence will increase D_v(*Number of*) when M_s(*Birth*) $= +$, decrease it when $M_s(Q1) = -$ (which is impossible), and have no effect when $M_s(Q1) = 0$ (i.e. the size of the population will increase if there is a birth rate). For a negative influence it would be vice versa. In effect, influences determine derivatives based on magnitudes.

The positive proportionality (*Number of* $\overset{P+}{\rightarrow}$ *Birth*) will increase D_v(*Birth*) when D_s(*Number of*) $= +$, have no effect when D_s(*Number of*) $= 0$, and decrease when D_s(*Number of*) < 0 (i.e. the derivative of population size is determined by the derivative of the birth rate). For a negative proportionality, it is vice versa. Since proportionalities determine derivatives based on derivatives, they can be said to propagate changes.

Other ingredients in Garp3 are operators, inequalities, value assignments and correspondences. Operators (+ and -) calculate the magnitude value of quantities (e.g. $Q_1 - Q_2 = Q_3$, to indicate $M_v(Q_1) - M_v(Q_2) = M_v(Q_3)$). Inequalities have multiple uses, e.g. $M_v(Q_1) > M_v(Q_2)$. Value assignments indicate that a quantity has a certain value ($M_v(Q_1) = Q_1(Plus)$). Finally, correspondences indicate that the current value of one quantity can be inferred from the current value of another quantity. There are quantity correspondences ($Q_1 \overset{Q_{qs}}{\leftrightarrow} Q_2$) and value correspondences ($Q_1(Plus) \overset{Q_v}{\rightarrow} Q_2(Plus)$), which can both be either directed or undirected. The value correspondence indicates that if $M_v(Q_1) = Q_1(Plus)$, $M_v(Q_2) = Q_2(Plus)$ (see Figure 1). If the value correspondence is bidirectional, the reverse inference is also possible. Quantity correspondences can be considered a set of value correspondences between each consecutive pair of the values of both quantities. There are also inverse quantity space correspondences ($Q_1 \overset{Q_{qs}^{-1}}{\leftrightarrow} Q_2$) that indicate that the first value in Q_1 corresponds to the last value in Q_2, the second to the one before last, etc.

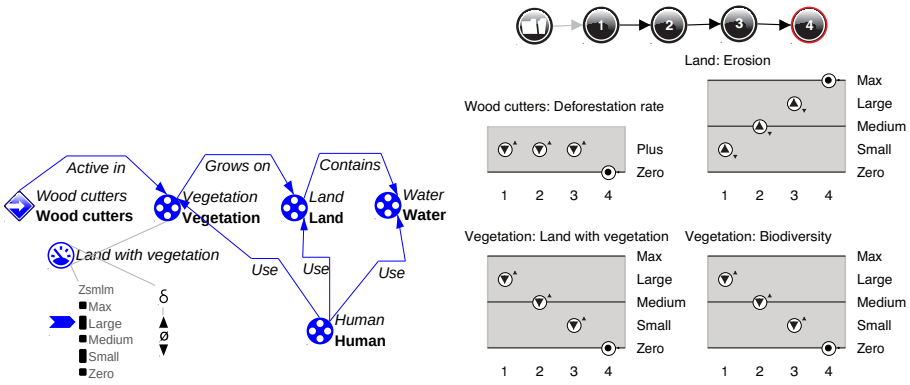
Garp3 models generate non-numerical simulations. These result in a state graph that represents how the initial situation changes (via the transitions) into other situations due to changes in quantity values (and inequalities between quantities). There are often multiple solutions, i.e. there can be branching in the state graph due to ambiguity. For example when there are multiple possible derivative values for a quantity.

3 Algorithm Design

3.1 Algorithm Input

The algorithm takes a qualitative description of a set of observations as input (Figure 2). The first input is a *scenario* (Figure 2(a)); the observed initial state of the system. The

second input describes the observed possible behaviours the system can exhibit given the scenario (Figure 2(b)). This second input is represented as a state graph, in which each state represents a system state (with particular magnitude and derivative values for each quantity), and the transitions represent the possible ways states can change into other states.



(a) Scenario (b) Simulation state graph and value history.

Fig. 2. The input of the automatic model building algorithm.

3.2 Desired Algorithm Output

The desired output of the algorithm is a qualitative causal model that explains the input observations. That is, the contents of the model should be such that simulating the model in Garp3 should reconstruct the behaviour (i.e. the state graph) that was provided as input. Particularly important are the causal dependencies represented in the model (e.g. as in Figure 1).

3.3 Key Algorithm Ideas

Covering and Pruning One key idea of the algorithm is to first generate those causal relations (proportionalities and influences) that 'cover' the desired output behaviour. That is, simulating the model with those should generate a simulation state graph that is equal or a superset of the desired state graph. The state graph is then pruned (i.e. the spurious states are removed) using correspondences and inequalities.

Causal Abstraction In many cases the precise causal ordering of quantities cannot be determined as only a correlation between quantities is known. As a result, the number of possible models grows exponentially in the number of quantities. To reduce the number of possible models, the possible causal orderings are represented as abstracted representations.

Actuator Patterns The causes of change within qualitative models are almost always influences. These influences model processes that affect the system. We have investigated several well-established models to search for patterns that include influences. We call these actuator patterns, since they serve as actuators of the system.

Representation Guided Learning The goal of the algorithm is to create a qualitative model. The algorithm uses the semantics of the qualitative model language to learn the representation.

4 Automated Model Building Algorithm

An overview of our automated model building algorithm is shown in Figure 3. The rounded squares represent functions, while the squares represent results. The algorithm starts with providing the input to the 'Find Naive Dependencies' function. The flow of the algorithm is clockwise from this function. Some of the intermediate results are used in other functions. The 'Check Concistency' function is used in two other functions, but is not one of the steps in the algorithm.

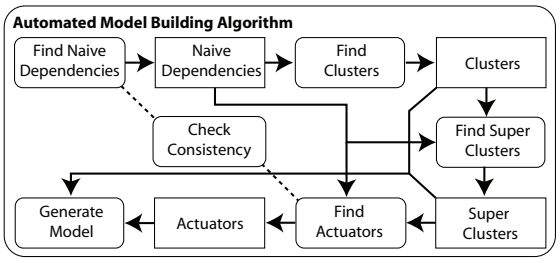


Fig. 3. Overview of the automatic model building algorithm.

4.1 Check Consistency

The check consistency function determines whether relations (generated by the algorithm) are consistent with the input observations. That is, whether the relations are consistent with all magnitude and derivative values of all quantities in each state of the input state graph.

The input of the function is a set of states consisting of the system structure, the quantities, the magnitude and derivative values in those states, and a set of (causal) relations. The function verifies the magnitudes and derivatives of the quantities based on the given values and the relations. This is done by two types of reasoning. Firstly, influence resolution which determines the derivatives of quantities based on the proportionalities and influences. And secondly, inequality reasoning to check for consistency.

A set of relations is considered consistent if they are consistent with all the states in the input state graph. That is, the derived values are equal to the given values in the state. Otherwise, the relations are considered inconsistent.

4.2 Find Naive Dependencies

The 'Find Naive Dependencies' function determines each possible single causal relation that is consistent with all input observations. This set of naive dependencies is used to

determine the plausibility of patterns in the next steps in the algorithm. The function uses 'Check Consistency' to check the consistency of single proportionality, influence and correspondence relations between each possible quantity pair.

The result a set of (causal) relations of which some are true (e.g. Land with vegatation $\overset{P+}{\rightarrow}$ Biodiversity), and some are false (Biodiversity $\overset{I-}{\rightarrow}$ Land with vegetation). This last false relation is consistent because Land with vegetation always decreases in the simulation result when Biodiversity has a value above zero, however it does not represent a valid explanation.

4.3 Find Clusters

The 'Find Clusters' function finds compartments of causality, called clusters, to reduce the amount of possible models. These clusters consist of quantities exhibiting equivalent (or inverse) behaviour, i.e. they have the same magnitude and derivative values in each state. Therefore, there should be (negative) proportionalities and (inverse) correspondences between each pair of quantities in the set of naive dependencies.

The causal ordering (i.e. which quantity affects which other quantity) in clusters cannot be determined, since there are possible causal relations between each pair of quantities (in both directions). This causes an explosion of the number of possible models, as each possible causal ordering can become a model. To deal with this, a single representation is used to store all the possible causal orderings of the cluster. For simulation purposes an arbitrary causal ordering is used (alphabetical so that clusters with the same type of quantities get the same ordering). Note that for reasoning purposes we assume that each cluster has a linear ordering, and that as a result each cluster has a single input (Q_{C1}^{In}) and a single output quantity (Q_{C1}^{Out}).

A second constraint is that quantities in a cluster should belong to the same structural entity. This idea (following the 'No function in structure' principle [5]) indicates that causality cannot affect other components in the system other than through its input and output. Our interpretation is that causal relations are mostly contained in compartments of causality (clusters), and only affect other clusters through their output quantities. After the 'Find Clusters' step each quantity is part of a cluster. An example of a cluster in our example model is: {Biodiversity, Land no vegetation, Land with vegetation} (belonging to the Vegetation entity).

4.4 Find Super Clusters

The 'Find Super Clusters' function searches for larger compartments of causality (clusters of clusters) to reduce the amount of possible output models. The function takes clusters and the naive dependencies set as input. A super cluster consists of clusters that exhibit equivalent behaviour. Therefore there are proportionality and correspondence relations (in the naive dependencies set) between each quantity pair in the super cluster.

The causal ordering between each of the clusters in unknown due to the fact that all the quantities in the super cluster exhibit equivalent behaviour. Each ordering would result in a possible output model. Therefore, a single representation to represent all the possible causal orderings. We assume a linear ordering of the clusters in the super

cluster. The first quantity of the first cluster is considered to be the input quantity of the super cluster, while the last quantity of the last cluster is the output quantity. After this step each cluster is part of a super cluster. In the example model (Figure 2), the clusters belonging the entities 'Human', 'Vegetation', 'Land' and 'Water' form a super cluster.

4.5 Find Actuators

Each of the super clusters has the capability to propagate the magnitude values (through its correspondences) and derivative values (through its proportionalities). Therefore, all the quantities in the system will get a magnitude and derivative value if the input quantity of each super cluster is actuated (given a magnitude and derivative value).

The 'Find Actuators' function searches for patterns that frequently occur in qualitative models. These patterns include influences that initiate change in the system (the derivative), and different ways to set magnitudes. Thus far we found the need for three re-occurring patterns that cause change. The patterns involve quantities from two or three super clusters. Each found pattern is validated using the consistency checking function.

External Actuator Pattern The external actuator pattern models an influence from outside the system (usually an agent). This pattern consists of an interaction between two super clusters. The output quantity of the first super cluster causes the actuation of the input quantity in the second super cluster ($Q^{Out}_{SC1} \overset{I+}{\rightarrow} Q^{In}_{SC2}$), and there is a feedback relation back to the first super cluster ($Q^{Out}_{SC2} \overset{P+}{\rightarrow} Q^{In}_{SC1}$). The influence causes the derivative of the quantities in the second super cluster to be set, while the feedback determines the derivative of the first super cluster. The magnitude value of the input quantity of the first super cluster is set using a value assignment ($Q^{In}_{SC1} = Q^{In}_{SC1}(Plus)$). The input quantity of the second super cluster has to be set in the scenario. In the example model the external actuator pattern is found between the 'Deforestation rate' of the Woodcutters and the large super cluster mentioned in the Find Super Cluster subsection.

Equilibrium Seeking Mechanism Pattern The equilibrium seeking mechanism pattern models equalizing flows due to a potential difference. For example, energy exchange between two objects with different temperatures. In this pattern three super clusters interact. The first two model the objects involved, while the third models the flow. Subtracting the output quantities of the two super clusters (e.g. the temperatures of the two objects) results in a flow (e.g. an energy flow between the two objects) ($Q^{In}_{SC3} = Q^{Out}_{SC1} - Q^{Out}_{SC2}$). This flow negatively influences the input quantity of the first super cluster, and positively influences the input quantity of the second super cluster (e.g. the energy quantities in each cluster) ($Q^{Out}_{SC3} \overset{I-}{\rightarrow} Q^{In}_{SC1}$, $Q^{Out}_{SC3} \overset{I+}{\rightarrow} Q^{In}_{SC2}$). The output quantity of the first super cluster is positively proportional to the flow, while the output quantity of the second super cluster is negatively proportional to the flow ($Q^{Out}_{SC1} \overset{P+}{\rightarrow} Q^{In}_{SC3}$, $Q^{Out}_{SC2} \overset{P-}{\rightarrow} Q^{In}_{SC3}$). The input quantities of the first two super clusters should be set in the scenario, while the flow is determined by the subtraction. The derivatives are determined by the proportionalities. This pattern is found in the communicating vessels model.

Interacting Processes Pattern The 'Interacting Processes' pattern consists of multiple interacting influences that model competing processes. This pattern consists of at least three super clusters. Two of these represent the competing processes ($Q^{Out}_{SC1} \xrightarrow{I+} Q^{In}_{SC3}$, $Q^{Out}_{SC2} \xrightarrow{I-} Q^{In}_{SC3}$), while the third represents the affected quantities. There are feedbacks from the affected quantities back to the influencing quantities ($Q^{Out}_{SC3} \xrightarrow{P+} Q^{In}_{SC1}$, $Q^{Out}_{SC3} \xrightarrow{P+} Q^{In}_{SC2}$).

The single influences in this pattern are not universally true (i.e. are not in the set of naive dependencies). The reason is that if there is no inequality information about the magnitudes of the influencing quantities, there are three options. Firstly, $Q^{Out}_{SC1} > Q^{Out}_{SC2}$, so $D_s(Q^{In}_{SC3}) = +$. Secondly, $Q^{Out}_{SC1} < Q^{Out}_{SC2}$, so $D_s(Q^{In}_{SC3}) = -$. Or finally, $Q^{Out}_{SC1} = Q^{Out}_{SC2}$, so $D_s(Q^{In}_{SC3}) = 0$. If no inequality information is known, all three behaviours will appear in the state graph (in different branches). Consequently, none of the single influences can be true in all these states. The interacting processes pattern is found in the population dynamics model.

4.6 Generate Model

The 'Generate Model' function generates the complete model. Firstly, it creates a model fragment in which it recreates the structure mentioned in the scenario and adds the quantities to the correct entities. Secondly, it adds the relations based on the established clusters. Thirdly, it adds the relations based on the super clusters. Finally, it adds the applicable actuator patterns to the model fragment to finish the model. When the generated model is simulated (with the input scenario) it generates the input behaviour.

5 Results and Conclusions

Our automatic model building algorithm successfully recreates four well established models: the tree and shade model, the communicating vessels model, the deforestation model (Figure 4) and the population dynamics model [6]. It is not yet able to deal with the Ants' Garden model [7] and the R-Star Plant-Resource model [8] due to their use of conditional inequalities to indicate when processes are active or not.

The results suggest that generating adequate qualitative causal models based on the behaviour of a system is a feasible approach to support researchers in articulating and discovering their conceptual understanding. Automatically building and improving the model should be significantly faster than building a model from scratch. As such it will help to establish theories that explain the phenomena provided as input data.

6 Future Work

We envision the algorithm as part of our modelling methodology [1]. The algorithm should be adapted to exploit the intermediate modelling results (created in Garp3) that result from the framework. For example, the structural model can be used to provide hints to the causal ordering in super clusters, since the causal relations likely follow the

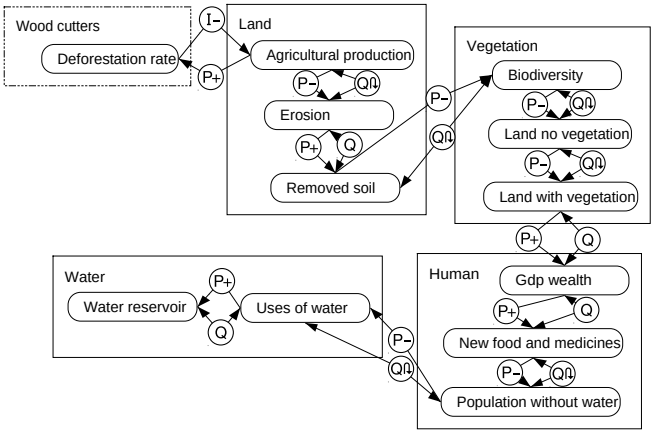


Fig. 4. The causal model generated by the automatic model building algorithm.

structural relations. This significantly reduces the amount of possible causal orderings of super clusters. By making the algorithm interactive, modellers could improve the causal ordering themselves. However, using the modeller-created state graph will also require the algorithm to deal with incomplete and inconsistent data. Dealing with such situations is input for future work.

References

1. Bredeweg, B., Salles, P., Bouwer, A., Liem, J., Nuttle, T., Cioaca, E., Nakova, E., Noble, R., Rios Caldas, A.L., Yordan, U., Varadinova, E., Zitek, A.: Towards a structured approach to building qualitative reasoning models and simulations. Ecological Informatics **3**(1) (2008) 1–12
2. Bridewell, W., Langley, P., Todorovski, L., Džeroski, S.: Inductive process modeling. Machine Learning **71** (2008) 132
3. Bratko, I., Šuc, D.: Learning qualitative models. AI Mag. **24**(4) (2004) 107–119
4. Forbus, K.D.: Qualitative process theory. Artificial Intelligence **24**(1-3) (December 1984) 85–168
5. de Kleer, J., Brown, J.S.: A qualitative physics based on confluences. Artificial Intelligence **24**(1-3) (December 1984) 7–83
6. Bredeweg, B., Salles, P.: Mediating conceptual knowledge using qualitative reasoning. In: Handbook of Ecological Modelling and Informatics. WIT Press (2008) (in press).
7. Salles, P., Bredeweg, B., Bensusan, N.: The ants garden: Qualitative models of complex interactions between populations. Ecological Modelling **194**(1-3) (2006) 90–101
8. Nuttle, T., Bredeweg, B., Salles, P.: R-star - a qualitative model of plant growth based on exploitation of resources. In Hofbaur, M., Rinner, B., Wotawa, F., eds.: 19th International Workshop on Qualitative Reasoning (QR'05), Graz, Austria (May 2005) 47–53