Ecological Informatics 4 (2009) 263–281

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

Garp3 — Workbench for qualitative modelling and simulation

Bert Bredeweg [*], Floris Linnebank, Anders Bouwer, Jochem Liem

Human Computer Studies, Informatics Institute, Faculty of Science, University of Amsterdam, Science park 107, 1098 XG Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 10 June 2009
Received in revised form 8 September 2009
Accepted 10 September 2009

Keywords:
Qualitative modelling and simulation
Qualitative knowledge and reasoning
Conceptual knowledge and models
Knowledge capture
Software and workbench

ABSTRACT

Easy to use workbenches for Qualitative Reasoning and modelling have been virtually nonexistent. This has a limiting effect on the use of this Artificial Intelligence technology and its uptake by a larger audience. We present Garp3, a user-friendly workbench that allows modellers to build, simulate, and inspect qualitative models of system behaviour. The workbench employs diagrammatic representations for users to interact with model content and simulation results, and provides seamless interoperability between the different modes of use. Domain experts can use Garp3 to create conceptual models in situations where numerical information is sparse or unavailable, or when they want to formalise their conceptual understanding of how systems behave. Garp3 can be applied to stakeholder management or dissemination activities to illustrate and explain phenomena, and facilitate discussion among participants. The workbench can also be used in formal education to have learners express concepts, or interact with existing models, and support them in developing their understanding of 'how things work'.

Garp3 incorporates a range of techniques from Artificial Intelligence known as knowledge-based techniques. The main goal of this paper is to present the representation and reasoning methods of these techniques as they have been developed and fine-tuned within the Garp3 workbench. The focus hereby is on the symbolic, non-numerical calculations that are required to generate the state-graph of a system's behaviour efficiently, while taking into account that users need to be able to track and understand this reasoning, both in terms of the end result and the intermediate results it delivers.

1. Introduction

Qualitative Reasoning is an approach from Artificial Intelligence that provides means to express conceptual knowledge such as the physical system structure, causality, the start and end of processes, the assumptions and conditions under which facts are true, qualitative distinct behaviours, etc. Qualitative models provide formal means to externalise thought on such conceptual notions (cf. Norman, 1993; Schwarz and White, 2005). There is growing interest from ecological experts to create qualitative models of phenomena for which numerical information is sparse or missing. There is also an interest to capture and simulate conceptual knowledge as such (cf., Jørgensen and Bendorrichio, 2001). However, building qualitative models is difficult and hampered by the lack of easy to use tools.

Tools are being developed that take a graphical approach to having learners build qualitative models (Bredeweg and Forbus, 2003). Graphical representations help reduce working memory load, and can support search and inference processes (cf. Larkin and Simon, 1987; Kulpa, 1994). Such external representations also help learners present their ideas to others for discussion and collaboration. This relates to the idea of using concept maps (Novak and Gowin, 1984). The main difference is the rich and detailed semantics used, which is based on Qualitative Reasoning formalisms. However, to further enhance usability, approaches such as Betty's Brain (Biswas et al., 2001) and Vmodel (Forbus et al., 2001) reduce the amount of modelling elements available in the model-building software. Although this is effective, it has the obvious drawback of not using the full potential of Qualitative Reasoning and the means it provides for representing conceptual knowledge.

In our approach, we want to preserve the full expressiveness of the Qualitative Reasoning formalism (Bredeweg et al., 2006). Moreover, we also want to address *domain experts* and support them in articulating and capturing their conceptual knowledge. We have therefore developed Garp3, a user-friendly workbench that allows modellers to build, simulate, and inspect qualitative models (http://www.Garp3.org). The software uses a diagrammatic approach for representing model content, and graphical buttons to communicate the available user options and manipulations.

This paper discusses the Garp3 workbench, particularly focussing on the internal representation and reasoning methods that are required for the functioning of this software. Section 2 provides a brief overview of what Qualitative Reasoning entails by discussing general characteristics, applications, and the overall working of such problem solvers. Section 3 describes the representation and reasoning employed in Garp3 from a user perspective. It highlights the ingredients modellers can manipulate and use to create models, and how the engine applies those ingredients for its key inference capability, namely the qualitative prediction of

[*] Corresponding author.
E-mail address: B.Bredeweg@uva.nl (B. Bredeweg).

system behaviour. This section also includes a concise example. Sections 4, 5 and 6 address the details underpinning this key inference. Section 4 discusses inequality reasoning, a capability that is central to Qualitative Reasoning and that is sometimes referred to as 'common sense arithmetic' (e.g. Simmons, 1986). Section 5 explains how Garp3 computes state descriptions of qualitatively distinct system behaviour, based on a scenario and a library of model fragments that capture partial knowledge of system components, processes and external influences (referred to as agents in Garp3). Section 6 describes how a state-graph can be constructed by determining transitions between the states that describe the qualitatively distinct behaviours.[1]

2. Background and characteristics

Qualitative Reasoning is a research area within Artificial Intelligence that appeals to both fundamental research on human cognition and to practical applications in industry (Bredeweg and Struss, 2003). Humans are very good in reasoning about the physical world using only fuzzy and qualitative information (cf. Gentner and Stevens, 1983; Winkels et al., 1998). At the same time experts, when discussing the occurrence of phenomena (e.g. represented in complex mathematical equations), typically refrain from using numerical details. Instead, they use qualitative arguments, including cause–effect relationships, to explain and convey their ideas (de Kleer, 1990). Observations such as these stimulate research on Qualitative Reasoning. The idea is that, ultimately, truly intelligent robots should have this kind of expertise in order to function properly in the real world (Hayes, 1979).

Meanwhile research on Qualitative Reasoning has established itself (e.g. Kuipers, 1994). Traditionally, the majority of research deals with physics and engineering (Weld and de Kleer, 1990). Successful application areas include autonomous spacecraft support (Williams et al., 2003), failure analysis and on-board diagnosis of vehicle systems (Struss and Price, 2003), automated generation of control software for photocopiers (Fromherz et al., 2003), and intelligent aids for learning about thermodynamic cycles (Forbus et al., 1999) as well as for learning about other systems and phenomena (Bredeweg and Forbus, 2003). With the support of the NaturNet-Redime project (http://www.NaturNet.org/) extra leverage was provided to the use of Qualitative Reasoning in ecology (Salles and Bredeweg, 2006), particularly concerning topics such as sustainability (see also other papers in this special issue).

Conceptual models are defined as models that improve our understanding of systems and their behaviour (e.g. Mylopoulos, 1992; Grimm, 1994; Guizzardi, 2005; Haefner, 2005). They can be used as a pre-mathematical modelling step or as standalone instruments for knowledge capture. Qualitative Reasoning technology is well suited to model and simulate such conceptual knowledge.

In Qualitative Reasoning, the quantities that describe the dynamic features of a system typically hold *qualitative* information concerning the current magnitude and direction of change, using an interval scale, consisting of an ordered set of labels (without any numerical information), e.g. {*Zero, Low, Medium, High*}. Such a set of labels is called a quantity space. Landmarks are specific labels within this set (actually points) that refer to situations in which the behaviour of the system changes significantly. For instance, a substance reaching its 'boiling temperature' will stop getting hotter and start boiling.

Qualitative simulations explicitly represent system behaviour and how it evolves over time. *Time is represented as a graph of states* (possibly including loops) that reflect qualitatively distinct system behaviour. States have duration, but the exact length of the duration is unknown (that is, not represented). State transitions occur when magnitudes of quantities change, because they increase or decrease. For instance, if in the current state (S_X) the temperature of a substance (T_S) is at boiling temperature (T_B) and decreasing ($\delta T_S < 0$), then the substance will move to a state (S_Y) in which its temperature is smaller than its boiling temperature: $S_X(T_S = T_B) \rightarrow S_Y(T_S < T_B)$.

Theory on Qualitative Reasoning has resulted in a set of dependencies that capture *cause–effect* relationships between quantities. These dependencies are defined such that on the one hand they represent conceptual notions that closely match human reasoning (Bredeweg and Schut, 1991; de Koning et al., 2000), while on the other hand they are grounded in mathematical formalisms allowing automated computation. Two typical examples of such dependencies are *direct influences* (changes caused by processes) and *proportionalities* (causal propagation of changes) (Forbus, 1984). These are the qualitative representations of ordinary differential equations and of monotonic functions, respectively. Proportionalities are sometimes referred to as *indirect influences*, as they propagate changes caused by direct influences to other quantities.

A typical Qualitative Reasoning engine has inference mechanisms to assemble the appropriate set of dependencies that describes a system in a certain state of behaviour, and to change this set for other states of behaviour when some of these dependencies no longer apply and/or new ones become applicable. Two aspects are important for this approach. Firstly, the idea that *behaviour can be inferred from the physical structure of the system*, and secondly, that knowledge about system behaviour is stored in (small) *model fragments* with conditional information detailing when such a fragment is applicable (including the physical structure to which it may apply). Preferably, these fragments implement *first principles* for the domain of knowledge from which they originate, empowering their reusability across systems. Note that in other modelling formalisms 'system structure' refers to the set of equations used to describe the system. In Qualitative Reasoning, however, system structure refers to the representation of the entities that constitute the system and how they are connected (hence, the term 'physical system structure').

Qualitative information may be inconclusive regarding opposing trends. For instance, consider pouring water from a tap into a partially filled bucket with a hole in the bottom. Will the water level decrease, stabilise, or rise? Qualitative Reasoning engines are designed such that they exploit such *ambiguity* to generate a solution space that represents *all possible solutions* (that is, all possible behaviours) as opposed to a single one. At the same time, ambiguity can often be resolved by providing inequality information regarding the relative impact of the relevant influence (e.g. if the tap inflow is larger than the outflow via the hole, the water level will rise). As such, when building qualitative models, ambiguity often triggers a knowledge discovery effort during which experts refine their knowledge, specifying priority for the phenomena involved and the conditions under which these models hold.

3. Representation and reasoning overview

This section introduces the Garp3 concepts of modelling (Section 3.1) and reasoning (Section 3.3). Section 3.2 provides an example that is added for illustration purpose only. For advanced models built with Garp3 see e.g. Salles et al. (2006), Tullos and Neumann (2006), Araújo et al. (2008), and the contributions to this special issue.

3.1. Knowledge representation

In Garp3 a distinction is made between basic model ingredients (Tables 1 and 2) and aggregates (Table 3) that can be built from the basic ingredients. There are two types of aggregates: scenarios and model fragments. Scenarios represent initial situations ready for simulation. Model fragments are partial models that represent individual parts of domain knowledge. The model fragments are stored in a library. When running a simulation the Qualitative Reasoning engine searches the library for fragments that are applicable to the conditions established in the scenario. In doing so it ultimately creates a state-graph (or behaviour-

[1] For further reading see also Bredeweg et al. (2008), which presents a structured method for constructing qualitative models. In Bredeweg and Salles (2009) the use of the Garp3 workbench in building a set of models and inspecting simulation results is explained and illustrated.

Table 1
Basic model ingredients (physical system structure).

Ingredient	Description
Entity	Entities are the physical objects or abstract concepts that constitute the system. Their relevant properties are represented as quantities that may change under the influence of processes. Entities are arranged in a subtype hierarchy.
Agent	Agents are used to model entities outside of the modelled system. Agents can have quantities influencing the rest of the system, which are called exogenous quantities or external influences.
Assumption	Assumptions are labels that are used to indicate that certain conditions are presumed to be true. They are often used to constrain the system behaviour generated by a model. They can be associated to structural and behavioural aspects of a system.
Configuration	Configurations are used to model relations between instances of entities and agents. Configurations are referred to as structural relations.

graph) that represents the possible behaviours of the system as initially described in the scenario. Users can inspect this state-graph using multiple views (Table 4).

Central to Qualitative Reasoning is the way a system is described during a period of time in which the *qualitative* behaviour of the system does not change (a qualitative distinct *state of behaviour*). The notion of change is subtle (and different from the notion of state in mathematical models), because numerical magnitudes of variables may change whereas from a qualitative point of view the behaviour of the system remains constant and does not change. When pouring water from a tap into a bucket, the amount and height of the water in the bucket both increase. From a qualitative point of view this is a single state of behaviour until a new landmark value is reached, e.g. when the bucket becomes fully filled and starts overflowing.

To determine the applicability of model fragments, a distinction is made between conditions and consequences. Conditions specify what must be present or hold in order for a model fragment to apply. Model fragments may also require other model fragments to be active in order to become active themselves. Consequences specify the knowledge that will be introduced when the model fragment applies. In the Garp3 interface conditional model ingredients are displayed in red, while consequences are displayed in blue.

Three kinds of model fragments are used in Garp3: static, process and agent. *Static fragments* are used to describe parts of the structure of the system, and the proportionalities that exist between the quantities. In static fragments all ingredients can occur except agents and influences caused by processes. *Process fragments* contain at least one direct influence, but no agents. Process model fragments are used to describe processes that take place within the system. A process can be defined as a mechanism that causes changes in the properties of some object, or that causes objects to disappear or new objects to appear. The notion of a process is a key concept in the Qualitative Process Theory approach to Qualitative Reasoning (Forbus, 1984). Finally, *agent fragments* contain an agent, i.e. an element that is external to the system and, although not influenced by that system, may impose one or more influences on it (Bredeweg et al., 2007).

3.2. Example of a simple qualitative model

Fig. 1 shows a scenario created in the Garp3 Build environment. This scenario defines an instantiated entity (named *Green frog*) of the class *Population*. This frog population has a quantity *Number of* with magnitude *Small*. The derivative is not specified and thus unknown.

Figs. 2, 3 and 4 show model fragments created with the Garp3 Build environment, which are stored in the model fragment library, and as such capture conceptual knowledge about population behaviour. The fragment shown in Fig. 2 represents how the size of a population (represented by the quantity *Number of*) determines the *Biomass* of that same population. The proportionality ($P+$) between *Number of* and *Biomass* specifies that changes in *Number of* propagate to changes (in the same direction) in *Biomass*. The correspondence (Q) specifies that the two quantities have co-occurring magnitudes. For instance, if *Number of* has magnitude *Large*, then so will *Biomass*.

Figs. 3 and 4 show model fragments representing the natality and mortality processes. In Fig. 3 the quantity *Birth (rate)* has a positive direct influence ($I+$) on *Number of*, which means that the magnitude of this rate determines the change in *Number of*. There is also feedback, implemented by the proportionality ($P+$), between *Number of* and *Birth*. Changes in *Number of* propagate to changes (in the same direction) in *Birth*. Finally, when there are no individuals there can be no birth. Hence, there is a value correspondence (V) from the magnitude *Zero* of *Number of* to the magnitude *Zero* of *Birth*. Notice that the line representing this correspondence has only one arrow. This means that the dependency is directed: only when *Number of* has magnitude *Zero*, the dependency becomes active and puts a constraint

Table 2
Basic model ingredients (behaviour).

Ingredient	Description
Quantity	Quantities represent changeable features of entities and agents. They are represented by their quantity value, which consists of magnitude (amount of stuff) and derivative (direction of change).
Quantity space	A quantity space specifies the range of possible values that a quantity can have. Each quantity has a user defined quantity space for the magnitudes, and a default quantity space for the derivatives, namely: $\{-, 0, +\}$. Values in a quantity space form a total order. Each qualitative value is either a point or an interval, and within quantity spaces these two types consecutively alternate.
Magnitude and derivative	The magnitude indicates the current value of a quantity. The derivative indicates how a quantity changes: decreasing ($-$), steady (*Zero*), or increasing ($+$). A value assignment indicates that a quantity has or should have a particular magnitude or derivative from the associated quantity space.
Direct influence	Direct influences are *directed* relations between two quantities, and are either positive or negative. Direct influences are the cause of change within a model, and are therefore said to model processes. Depending on the magnitude of the source quantity and the type of influence, the derivative of the target quantity either increases or decreases. An influence $I+$(Q2, Q1) causes the quantity Q2 to increase if Q1 is positive, decrease if it is negative, and remain stable when it is zero (assuming there are no other causal dependencies on Q2). For a negative influence $I-$ this is just the opposite.
Proportionality	Qualitative proportionalities are *directed* relations between two quantities. They propagate the effects of a process, i.e. they set the derivative of the target quantity depending on the derivative of the source quantity. For this reason, they are also referred to as indirect influences. Like direct influences, proportionalities are either positive or negative. A proportionality $P+$(Q2, Q1) causes Q2 to increase if Q1 increases, decrease if Q1 decreases, and remain stable if Q1 remains stable (assuming there are no other influences on Q2). For a negative proportionality $P-$ this is just the opposite.
Correspondence	Correspondences are relations between qualitative values of different quantities, and can be either *directed* or *undirected*. The former means that when value *A* of quantity *X* corresponds to value *B* of quantity *Y*, the simulator derives that quantity *Y* has value *B* when quantity *X* has value *A*. If the correspondence is undirected, it also derives the value *A* of quantity *X* when quantity *Y* has value *B*. There are 6 correspondence types, 12 if directedness is included.
Inequality	Inequalities ($<, \leq, =, \geq, >$) specify an ordinal relation between two items, i.e. that one item is different from (or equal to) the other item. Because inequalities specify an order between items, they are sometimes referred to as ordinal relations. There are eleven ways to use inequalities, also depending on the type of the two items related by them.

Table 3
Model ingredients (aggregates).

Ingredient	Description
Scenario	Scenarios describe the initial state of a system, and can consist of all the ingredients that can be used as conditions in model fragments, except for other model fragments. Scenarios are used as input for the qualitative simulator. The qualitative simulator interprets the scenario (finds applicable model fragments, incorporates their consequences, and derives values) to generate one or more initial states. These initial states are used to generate the rest of the behavioural graph.
Model fragment	Model fragments describe part of the structure and behaviour of a system in a general way. They are partial models that are composed of multiple ingredients. Model fragments have the form of a rule. This means that model ingredients are incorporated as either conditions or consequences. Model fragments themselves can be reused within other model fragments as conditions, called imported model fragments. Furthermore, subclasses of model fragments can be created, which augment the parent model fragment with new ingredients. The consequence ingredients of model fragments that match the current system situation will be added to that scenario. In that case, the scenario fulfils the conditions specified in the model fragment (which describes a general situation).

of the magnitude of *Birth* (and not vice versa). Fig. 4 shows similar knowledge for the mortality process. The main difference is that *Death (rate)* has a negative influence on *Number of*, which is represented by a negative direct influence (*I*−) from *Death* on *Number of*.

Having specified one or more scenarios and a library with relevant knowledge, simulations can be run in the Garp3 simulate environment. Simulations can in general be run incrementally or fully. In the case of an incremental simulation the user can select states and have the engine work on those. In the case of a full simulation the engine automatically infers all possible behaviours for a given scenario. The results of such a full simulation are shown in Fig. 5 (state-graph), Fig. 6 (value history) and Fig. 7 (causal model). The state-graph has 7 states, shown as black circles with numbers (unique identifiers). (Notice that the state numbers (1 to 7) are just identifiers generated by the software.) Each state reflects qualitatively distinct behaviour of the simulated system (see Fig. 6). The arrows between some of the states indicate state-transitions. For example, the behaviour represented by state 3 may change into the

Table 4
Simulation output.

Ingredient	Description
State	A state describes a particular situation of a modelled system, reflecting a qualitatively unique behaviour. A state (description) is an assembly of applicable model fragments and thus contains information about the physical structure, the associated quantities and their values (in this state), and inequalities and the causal dependencies between the quantities. In a Garp3 state-graph, a state can be either: interpreted, terminated, ordered or closed.
State-graph (or behaviour-graph)	A state-graph is a set of states, and the possible transitions between those states, which represents the behaviour of a modelled system. State-graphs are generated as a result of simulating a qualitative model.
Value history	The value history describes how quantity values change through a sequence of states (usually a behaviour path).
Equation history	The equation history describes how the ordinal relations change through a sequence of states (usually a behaviour path).
Causal model	A causal model is a diagram containing quantities and causal dependencies (proportionalities and influences). It describes how the quantities are causally related. In the context of Garp3 the term causal model is also used to refer to the dependency diagram in the Simulate context.

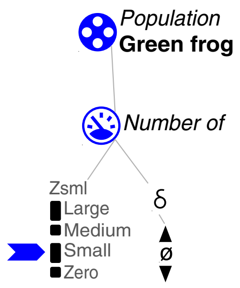


Fig. 1. Scenario for running the simulation of a population named *Green frog*. *Green frog* is thus an instance (a specific case) of the class *Population*. The entity hierarchy to which *Population* belongs is not shown. The *Green frog* population has been assigned the quantity *Number of*, with the quantity space *Zsml*. This quantity space consists of four values {Zero, Small, Medim, and Large}. The derivative quantity space is default, and consists of {−, 0, +}. The arrow pointing down refers to −, while the arrow pointing up refers to +. The quantity *Number of* has been assigned the magnitude *Small* as initial value, while the derivative (δ) is unspecified.

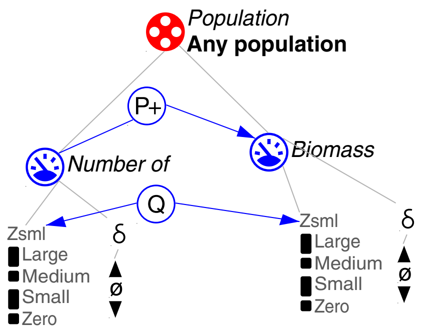


Fig. 2. A model fragment capturing knowledge about the relationship between the quantities *Number of* (size) and *Biomass* of populations. The fragment becomes active when an entity of type *Population* exists (e.g. because it was defined in the scenario). When active, this fragment introduces (or reuses if already known) the quantities *Number of* and *Biomass*, and their accompanying quantity spaces. The fragment specifies a positive proportionality (*P*+) between *Number of* and *Biomass* (the latter following changes happening to the former), and a full quantity space correspondence (*Q*) between the quantity spaces of these quantities. The correspondence is bi-directional meaning that whenever the magnitude of one quantity is known the magnitude of the other quantity can be inferred.

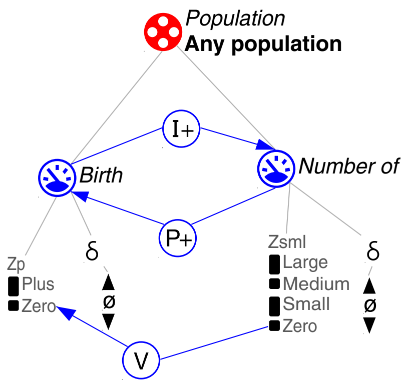


Fig. 3. A process model fragment specifying the natality process for populations. The fragment becomes active when an entity of type *Population* exists. The name of this population is irrelevant, hence the local dummy *Any population*. The fragment introduces the quantities *Birth* (rate) and *Number of* (or reuses them when already defined). *Birth* can take on two magnitudes {Zero, Plus}, and has a positive influence (*I*+) on *Number of*. Therefore, when *Birth* = *Plus*, it makes the *Number of* increase, and when *Birth* = *Zero*, it does not change the *Number of*. *Number of* has a positive proportionality (*P*+) with *Birth*, which implements a feedback loop (changes in *Number of* propagate to changes in *Birth*, in the same direction). The value correspondence (*v*) between the magnitudes *Zero* of the two quantities specifies the idea that in the case of a non-existing population (*Number of* = *Zero*), *Birth* is also *Zero*.

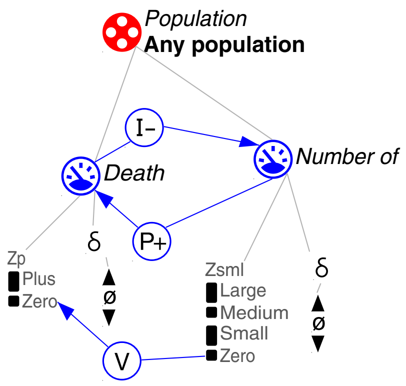


Fig. 4. A process model fragment specifying the mortality process for populations. The fragment becomes active when an entity of type *Population* exists. The name of this entity is irrelevant, hence the local dummy *Any population*). The fragment introduces the quantities *Death* (rate) and *Number of* (or reuses them when already defined). *Death* can take on two magnitudes {Zero, Plus}, and has a negative influence (*I*–) on *Number of*. Therefore, when *Death* = *Plus*, it makes the *Number of* decrease, and when *Death* = *Zero*, it does not change the *Number of*. *Number of* has a positive proportionality (*P*+) with *Death*, which implements a feedback loop (changes in *Number of* propagate to changes in *Death*, in the same direction). The value correspondence (*v*) between the magnitudes *Zero* of the two quantities specifies the idea that in the case of a non-existing population (*Number of* = *Zero*), *Death* is also *Zero*.

behaviour represented by state 4 or 5. Multiple arrows reflect ambiguity. Such ambiguity arises whenever the available information does not suffice to distinguish between qualitatively distinct outcomes. In fact, each unique sequence of states reflects a possible behaviour of the system. In the example shown here, there are four behaviour-paths, that is, four alternative behaviours that the system may manifest, given the initial details defined in the scenario and the knowledge specified in the model fragment library.

The value history (Fig. 6) provides a particular view on the simulation results. It works on a set of states selected by the user. To inspect the system behaviour one has to follow the behaviour-paths. For the details shown in Fig. 6, all states are selected. For these selected states the value history shows the quantities, their possible magnitudes (right-hand side), and their current magnitude and derivative in each state. Notice that the derivatives are graphically represented as black arrows. Pointing upward represents an increasing quantity (positive derivative) and pointing downward represents a decreasing quantity (negative derivative). Zero (steady) is represented as a circle. For instance, the quantity *Number of* can take on magnitudes {*Zero, Small, Medium, Large*}. In state 1 it has the magnitude *Small* and it decreases, hence its qualitative value is <*Small*, −>. If we focus on the behaviour-paths, the following behaviours can be observed:

- Path [1 → 6]: *Number of* and all other quantities go to *Zero*; the population ceases to exist.
- Path [2]: The population does not change. *Birth* and *Death* rate are in balance.
- Path [3 → 4]: The population increases in state 3 and stabilises in 4 at magnitude *Medium*.
- Path [3 → 5 → 7]: The population increases in state 3 and 5, and keeps increasing in state 7 while being in the 'extreme' interval *Large*.

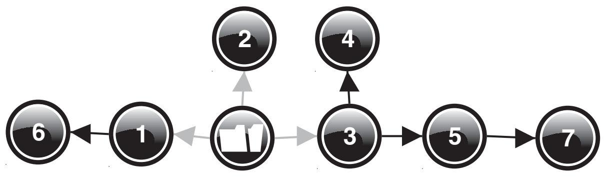


Fig. 5. State-graph for simulating the scenario shown in Fig. 1. The simulation has 7 states in total. State 1, 2, and 3 are initial states. State 2, 4, 6, and 7 are end states. The simulation shows 4 behaviour-paths. ([1 → 6], [2], [3 → 4] and [3 → 5 → 7]).

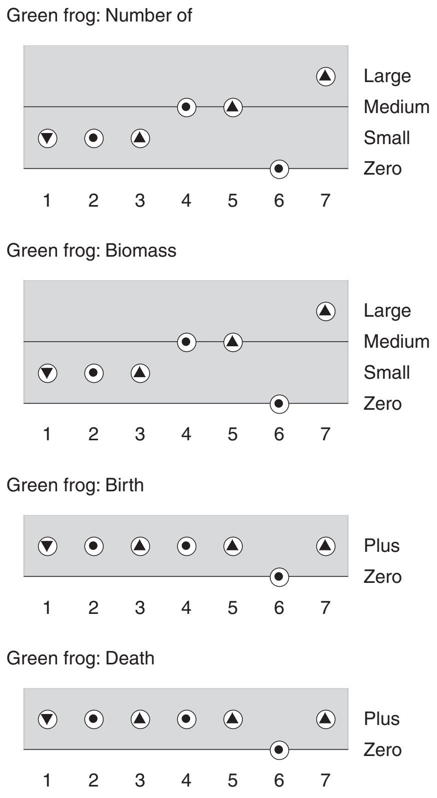


Fig. 6. Value history when simulating the scenario shown in Fig. 1. Consult the state-graph (Fig. 5) to know which states are successors in a behaviour-path. The longest path starts in state 2 and proceeds via state 5 to state 7. In this path *Number of* starts being <*Small*, +>, becomes <*Medium*, +>, and finally moves to <*Large*, +> in state 7.

Alternative views on the simulation results include the inequality history and the transition history (which are both not shown here), and the causal model (Fig. 7).

The causal model is an overview of all the behavioural details represented in a specific state. As such the causal model shows an

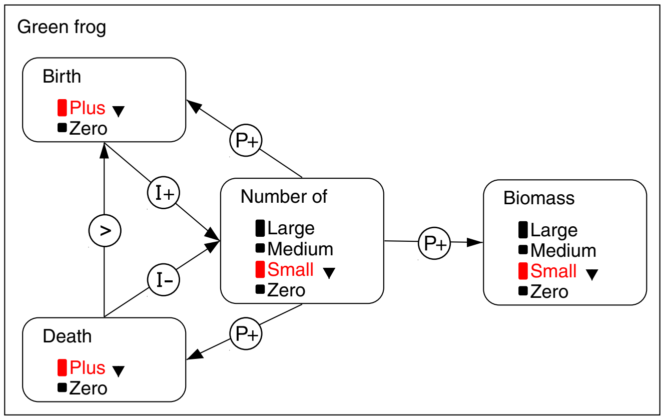


Fig. 7. This figure shows the dependency view (also referred to as causal model) as generated by the simulator for state 1. The dependency view shows entities (in this case *Green frog*) as squares containing the quantities and their quantity spaces, the current magnitudes (colored red) and derivatives (black arrows), and the (causal) dependencies between the quantities. The visualization of simulation results (Garp3 Simulate environment) is different compared to the one used during model building (Garp3 Build environment). Notice that each state has it own unique model details: an aggregation of the fragments that have become active and the calculations made possible by those details. Here in state 1, *Death* > *Birth*. Hence the influences of Death is dominant in this state which results in *Number of* decreasing, which indirectly (*P*+) also causes the other quantities (*Biomass*, *Birth* and *Death*) to decrease.

integrated view of the details present in all the model fragments that are active in the state. Fig. 7 shows the casual model obtained in state 1 from the state-graph shown in Fig. 5. In this state the model fragments shown earlier in Figs. 2, 3, and 4 are active. Hence, the details in Fig. 7 are an assembly of the contents of those fragments. When opening a causal model view in Garp3, users can select certain types of details to be shown or hidden from view. For example, Fig. 7 does not show the correspondences between magnitudes.

3.3. Reasoning: state-graph generation

The simulation of a scenario is done by the Garp3 reasoning engine. It uses a 'state-by-state', strategy to construct the state-graph. Fig. 8 gives an overview of the main inferences. The library of model fragments and the scenario are input for the reasoning engine. From this input, one or more states are constructed by the *Find states* procedure. These are the initial nodes in the state-graph (or behaviour graph). Then, possible changes are determined by the *Find transitions* procedure for each new state. These changes form 'transition scenarios' which are again input for the *Find states* procedure. Scenarios and transition scenarios can lead to already existing states or to new states. In the former case a transition link is added to the state-graph. In the latter case a link and a node (i.e., a new state) are added. This process repeats until all states have been analysed for possible changes.

Another possibility would be to determine all possible states first (using exhaustive search) and then all possible transitions between them. However, the state-by-state approach has two advantages. Firstly, the user has the possibility to control the simulation and decide which behaviour of the system to investigate further. This is particularly useful as an instrument for model building and for operating large models in general. Secondly, the reasoning of the engine is more 'humanlike' in the sense that it considers the situation and determines how this is changing. This particularly provides leverage for usability of the software in educational contexts (cf. Bredeweg and Schut, 1991; De Koning et al., 2000).

As shown in Fig. 8, the *Inequality reasoning inference* is used throughout the reasoning process. Section 4 discusses this inference in detail. Sections 5 and 6 cover the *Find states* and *Find transitions* procedures, respectively.

4. Inequality reasoning: the mathematical engine

Solving inequality, or ordinal, relations is a cornerstone of the Garp3 reasoning engine. Compared to structural and causal relations (which usually remain in place during an entire simulation), inequality relations typically change during the simulation, thereby representing the dynamic aspects of the modelled system. For each state a coherent mathematical model is constructed and maintained using the inequality reasoning capabilities of the engine.

4.1. User-level representation and the engine-specific mathematical model

Inequalities play a crucial role in representing specific system states. Fig. 9 illustrates a connected containers system in the process of levelling out. In this example, quantity *A* represents the amount of liquid in the container on the left and quantity *B* represents the amount of liquid in the container on the right. Although similar, both quantities have their own quantity space, because landmarks can never be assumed to be equal without previous specification. As shown in Fig. 9, the maximum amounts are in fact quite different in this example.

To increase user friendliness Garp3 has a set of simulation preferences that users can use to steer the working of the simulator. One of these options is to assume that if two qualities use the same quantity space, their values are equal. This means that equality between points in quantity spaces of the same generic type is automatically specified.

Remember that points on a quantity space are called landmarks. Intervals in quantity spaces are defined by adjacent landmarks. As discussed below, the core inequality reasoning only considers these landmarks and not the intervals. Note that landmarks are in principle stable. In specific models however, landmarks may represent changing magnitudes, for example, the freezing/melting point of water in an environment with changing pressure or salinity.

Inequalities can specify the relation between several types of model ingredients. The following five types of inequality relations are used in Garp3:

1. Quantity ↔ Quantity
2. Quantity ↔ Landmark

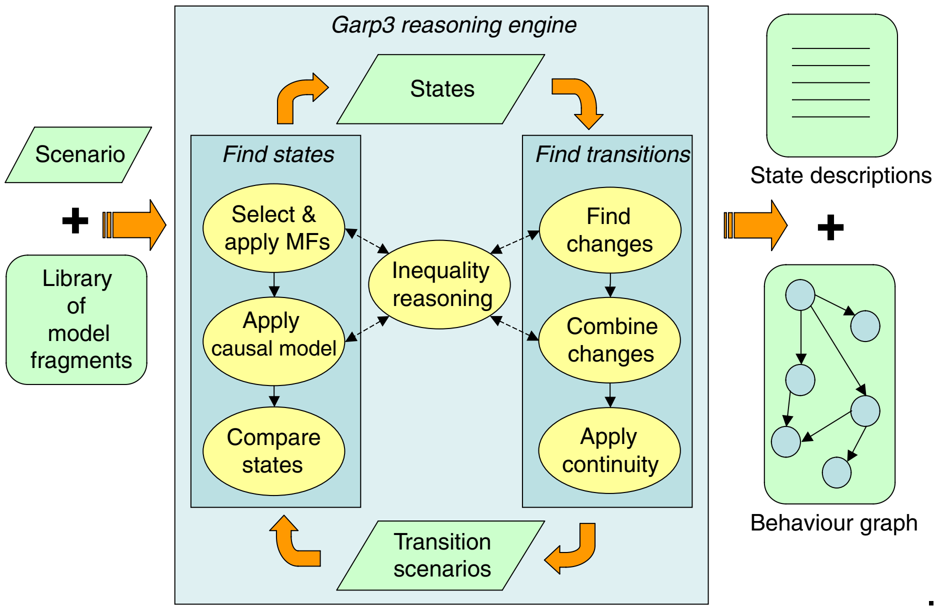


Fig. 8. Garp3 reasoning engine overview and context. The two main inferences are *Find states* and *Find transitions*. A simulation takes as input a scenario and a library of model fragments, and generates a state-graph (or behaviour-graph) as output (e.g. shown in Fig. 5).

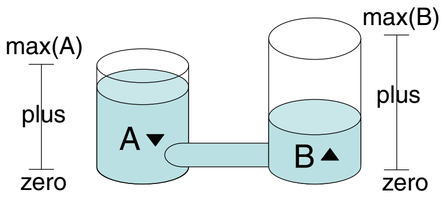


Fig. 9. An equalizing connected containers system. Due to pressure difference the liquid in the left container (A) is decreasing, while the liquid in the right container (B) is increasing.

3. Landmark ↔ Landmark
4. Derivative ↔ Derivative
5. Derivative ↔ Zero

Note that derivatives use the simple 'sign' quantity space {*Minus, Zero, Plus*}, so *Zero* is the only landmark for derivatives. Furthermore, the sets of magnitudes and derivatives are strictly separated: regular quantity space magnitudes cannot be related (by an inequality) to derivatives, also not to the *Zero* landmark on the derivative quantity space. However, *Zero* is a universally shared landmark within each set as such (magnitude/derivative). For the connected containers example (Fig. 9), the inequality relations listed in Table 5 describe the state of the system.

Each quantity space will supply only a minimal set of ordering relations as input for the engine-specific mathematical model. This set consists of inequalities between *adjacent* landmarks. The full transitive closure of the quantity space is derived by the system. For example, consider the following temperature quantity space: {*point (Absolute–Zero), Solid, point(Freeze–Melt), Liquid, point(Evaporate–Condense), Gas*}. The following two relations are added by the engine as inequalities between adjacent landmarks: *Absolute–Zero <Freeze–Melt* and *Freeze–Melt <Evaporate–Condensate*. The relation *Absolute–Zero <Evaporate–Condensate* is not explicitly added to the mathematical model. Instead this relation is derived by the system (when needed).

Quantity magnitudes (e.g. *Small* as shown in Fig. 1) and derivatives are represented internally by relating them to landmarks on their quantity space. The mathematical model (i.e. the collection of landmarks) is translated to the user-level representation (i.e. points and intervals) using the ordered set of qualitative magnitudes from the quantity space and the set of rules listed below. The translation from user-level representation to the engine-specific mathematical model uses a similar set of rules.

- If a quantity (magnitude or derivative) is equal to a certain landmark, then this qualitative magnitude is applied.
- If a quantity is smaller than the landmark directly above an interval and larger than the landmark directly below this interval, then the qualitative magnitude between these two landmarks is applied.
- If a quantity is smaller than the lowest landmark, then the qualitative magnitude below this point is applied.
- If a quantity is larger than the highest landmark, then the qualitative magnitude above this point is applied.
- In all other cases the specific quantity magnitude or derivative is unknown.

Table 5
Inequality relations describing two connected containers. The *Type-nr* refers to the 5 types of inequality relations that can be specified (discussed earlier in Section 4.1).

Example relation	Type-nr.
$A>B$	1
$A>$Zero	2
$B>$Zero	2
$A<$Max(A)	2
$B<$Max(B)	2
Max(A)$>$Max(B)	3
Zero$<$Max(A)	3
Zero$<$Max(B)	3
$\delta A<\delta$B	4
$\delta A<$Zero	5
$\delta B>$Zero	5

Together, all internal relations of the state description form a partial ordering. This ordering is the engine-specific mathematical model. As an example, consider the partial ordering shown in Fig. 10 for the connected containers state from Fig. 9. In this example, each quantity again has the quantity space {*Zero, Plus, Max*}, but now the system description is not fully specified (compared to the details listed in Table 5). Firstly, the relation between the heights of the containers is not specified, so the relationship between *Max(A)* and *Max(B)* is unknown. Secondly, it is not known how the liquid heights in both containers compare, so the relation between *A* and *B* is also unknown. However, some information about the magnitudes can be found in this partial ordering. Quantity *A* is larger than *Zero* and smaller than *Max(A)*, so it must have the magnitude 'Plus'. Quantity *B* is larger than *Zero*, but smaller than or equal to *Max(B)* so its precise magnitude is unknown.

To assure a quantity remains within its quantity space some constraints are automatically added. For instance, in the connected containers example the height of *A* cannot be lower than the bottom landmark or larger than the top landmark, so the relations $A\geq Zero$ and $A\leq Max(A)$ are added. If the last value of a quantity space is an interval, such an additional constraint is not added. For derivatives, specific constraints are added for maximum and minimum points of the quantity space. This is because a quantity cannot be increasing in the top point of a quantity space or decreasing in the bottom point. For example, in the case of the connected containers the following conditional relation (Section 4.3) is added: 'if $A=Max(A)$ then $\delta A\leq Zero$'. Again, if the top of a quantity space is an interval, no constraint is needed, since this interval will be infinitely extended. However, these derivative constraints are optional (via Simulation preferences) in order to allow users a flexible approach to modelling. A model may be a partial description of a larger phenomenon where the quantity space is also part of a larger domain. E.g. overflow in a container may be modelled as an increasing amount in the top landmark.

Two further remarks can be made about the user-level representation and the internal mathematical model. Firstly, the former uses all operators: {>, ≥, =, ≤, <}, whilst the latter uses a canonical form only including the >, ≥ and = operators. This ensures that only one form of a particular relation can be present in the system and no effort is wasted on computations involving alternative forms. For example the relations $A=B$ and $B=A$ are equivalent, and both willbe translated to the same internal form. Similarly the relations $A+B>0$ and $0<B+A$ are equivalent, and will therefore be translated into a canonical internal form. Also, subtractions are rewritten to sums: $A-B=C$, translates into $A=B+C$. Secondly, at the user-level representation, quantity values, landmarks and derivatives are different types of model ingredients. In the mathematical model only landmarks are represented. This holds for both magnitudes and derivative quantity spaces. Therefore, the general inference method described in the next section also applies to both.

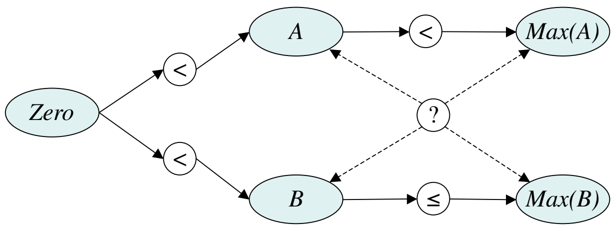


Fig. 10. Sample partial ordering for the two quantity magnitudes (*A*) and (*B*) and the landmarks on their quantity spaces.

4.2. General inference method inequality reasoning

All the relations in the internal mathematical model of Garp3 have the following form:

$Sum1\ rel\ Sum2$

with $rel \in \{>, \geq, =\}$. *Sum1* and *Sum2* are additions of variables, possibly containing a single element. The following three basic principles are used to make inferences in the Garp3 inequality reasoning engine:

1. Basic algebraic simplification
2. Anti-symmetry
3. Transitivity

We use constant elimination as our algebraic simplification principle. The Garp3 inference rule is:

1) $(Sum1 + V)\ rel\ (Sum2 + V) \rightarrow Sum1\ rel\ Sum2.$

From the property of anti-symmetry, the following inference rule derives:

2) $(Sum1 \geq Sum2)\ \&\ (Sum2 \geq Sum1) \rightarrow Sum1 = Sum2.$

Garp3 uses the rules given in Table 6 for doing transitive inference. These are based on the rules used in the Quantity Lattice proposed by Simmons (1986). They are more general than standard transitive inferences, which can only deal with relations that involve a single variable.

Finally, as a last reasoning step, any derived empty summation is replaced by the variable representing zero.

An important function of this general inference method is to detect inconsistencies. A contradiction is found whenever a relation of the following form is derived[2]:

$Sum > Sum$

Some examples illustrate the way the general inference method works:

Ex1) $A > B,\ B > C$	(given)
$(A > B)\ \&\ (B > C) \rightarrow A + B > B + C$	(transitivity)
$A + B > B + C \rightarrow A > C$	(simplification)
$A > C$	(result)
Ex2) $P > Q,\ (Q = P)$	(given)
$(P > Q)\ \&\ Q = P \rightarrow P + Q > P + Q$	(transitivity)
$P + Q > P + Q \rightarrow 0 > 0$	(simplification)
Contradiction	(result)
Ex3) $R > S,\ T > 0$	(given)
$(R > S)\ \&\ (T > 0) \rightarrow R + T > S$	(transitivity)
$R + T > S$	(result)
Ex4) $X \geq Y,\ Y \geq X$	(given)
$(X \geq Y)\ \&\ (Y \geq X) \rightarrow X = Y$	(anti-symmetry)
$X = Y$	(result)

4.3. Conditional relations: correspondences

Correspondences indicate co-occurring magnitudes or derivatives. All user-level correspondence types (Table 2) are represented as sets of conditionally linked relations in the Garp3 mathematical model. These have the form: *if* relations set *A* is derivable *then* assert relations set *B*,

[2] In the implementation, all relations of this form are simplified to the relation *Zero > Zero*.

Table 6
Transitive inference rules based on the Quantity Lattice proposed by Simmons (1986). R1 and R2 represent the inequality input relations, while the table shows how these can be combined. If, for instance, R2 represents $A = B$ and R1 represents $C > D$, then a possible new relation is $A + C > B + D$.

R1 & R2 →	$A = B$	$A \geq B$	$A > B$
$C = D$	$A + C = B + D$	$A + C \geq B + D$	$A + C > B + D$
$C \geq D$	$A + C \geq B + D$	$A + C \geq B + D$	$A + C > B + D$
$C > D$	$A + C > B + D$	$A + C > B + D$	$A + C > B + D$

where set *A* describes a magnitude (or derivative) on the quantity space of one quantity (Section 4.1) and set *B* describes a magnitude (or derivative) on the quantity space of another quantity. These conditional pairs can be directed or undirected, and multiple pairs may be needed to describe correspondences between whole quantity spaces. A routine is present in the inequality reasoning engine to check for firing conditional relations every time new relations are added to the mathematical model.

4.4. Computational efficiency

Qualitative Reasoning is in general computationally intensive, and the complexity can be NP-complete (Dormoy, 1988; Veber et al., 2006). The rich representation of inequality relations in Garp3, although powerful, causes the inequality reasoning inferences to be vulnerable to combinatorial explosion. To detect all possible contradictions it is necessary to compute the full transitive closure of a partial order that can be extended with new nodes via additions and subtractions (notice that this is a none-standard approach). For example, in regular graph theory the transitive closure of $\{A > B,\ B > C,\ C > D\}$ is $\{A > B,\ B > C,\ C > D,\ A > C,\ B > D,\ A > D\}$, but our system would also derive at least $A + B > C + D$ and $A + C > B + D$ which leads to the creation of the new nodes $A + B$, $C + D$, $A + C$ and $B + D$. This means that the potential graph to be investigated becomes (much) larger. Also, the state description is built incrementally, so every time a new relation is added the whole process potentially needs to be redone, as previously dead ends may have become valid points of inference. Other factors contributing to the computational complexity are (1) the fact that for each state and transition the derivable relations need to be recalculated, and (2) that each rejected transition scenario requires calculation of at least a partial state description before it can be disposed of. Also, all correspondences need to be assessed each time new information is available. The techniques that are discussed in the following subsections have been applied to keep computations with the Garp3 mathematical model tractable. Additional detail is provided for several of these concepts in Section 4.5.

4.4.1. Bitmaps for fast set operations

The sums of quantities on each side of an inequality statement are internally represented by bitmaps (also known as bit arrays, bit vectors or bit sets). These allow efficient computation of set operations like union and intersection, which are basic operations in the three inference types mentioned above (Section 4.2). The translation of pointer sets to bitmaps has costs, but these are much smaller than the efficiency gain in any simulation of reasonable size.

4.4.2. Representation of equality by reference to reduce relation sets

A reduction of the number of quantities and relations to be analysed is obtained by assigning equal quantities ($A = B$) a single internal pointer. This ensures that the equality relation stays known, and moreover, if for quantity *A* a set of *n* relations is known, then an equal set will be derivable for quantity *B*. These relations are unnecessary when using a shared pointer. The only exception is when equal quantities appear together in a sum. In this case each of them must have a separate pointer because bitmaps represent sets and therefore cannot accommodate the sum of both quantities if they share a single pointer.

4.4.3. *Avoidance of irrelevant 'non-simplifiable' new nodes to reduce inferences*

By removing results of transitive inferences that cannot be simplified a reduction in the amount of possible derivations is made. For example, if the relations $A>B$ and $C>D$ are combined, the resulting relation $A+C>B+D$ is not saved or used for further inferences. Such a relation typically has no relevance from a modelling perspective. This effectively stops the generation of many meaningless nodes in the graph. There is no absolute proof that this approach does not exclude any meaningful derivations, but in practice it has been a strong heuristic that does not interfere with the derivation of meaningful relations.

4.4.4. *Restriction of inference application to certain relation sets to reduce inferences*

The number of possible derivations is restricted by not allowing inferences to be made on all combinations of relations. This heuristic works by only combining incoming new relations with asserted relations and not with all derivable relations. Moreover, all newly derived relations are combined with each other. The hypothesis is that this *combine new with asserted* heuristic works because relations asserted by the model typically provide a graph structure with full (but sparse) connectivity and relations can often be derived in many ways. Again there is no absolute proof that meaningful derivations are not missed, but in practice this method has proven very valuable.

4.4.5. *Separation of unconnected partial orderings to reduce inferences*

Common combinatorics show us that computing the results for several smaller sets of relations is a much simpler task than computing the results for the single combined set. This approach of splitting the Garp3 mathematical model into 'quantity contexts' is feasible since in many models multiple sets of completely unrelated quantities can be distinguished. A 'quantity context' can be established if for every variable in the context there is a relation that connects it to another variable of the context. The main point here is that two relations can only lead to a meaningful derivation if they have related variables. This is correct, because no contradiction can be derived if there are no shared variables (a contradiction is always of the form $X>X$). In addition, because of the restriction on non-simplifiable results described above, relations without shared variables will not yield valid inference results. Note that the landmark *Zero* is a special case here that is not used for determining contexts since it would tie almost all contexts together. Because it is a single variable it does not affect the statements made above. Also note that because of this approach the separation between magnitude and derivative information (mentioned in Section 4.1) requires no additional mechanism. Since there are no user relations connecting magnitudes to derivatives, these two aspects of a quantity value will always form separate contexts.

4.4.6. *Restriction of inference depth to reduce inferences*

An optional measure to control very large simulations is to set a limit to the amount of parent inferences a derivation can have. If the amount of intermediate derivations is limited, a very significant improvement in efficiency is achieved. Experiments run in Garp3 have shown that above certain model-specific limits, no meaningful derivations are made. However, no guarantee can be given that all possible derivations are made when any particular limit is applied. If the theoretical function mapping depth to derivations would be monotonically decreasing, then the algorithm could simply stop when no more new relations were derived, and it would be certain that no more relations could be found. This is not the case however; one never knows if new derivations are possible. The user should be informed of the class of derivations that cannot be made given a certain limit, but no solid information about this point exists. Most models in the Garp3 repository (http://www.Garp3.org) show correct behaviour with a depth-limit of 5. Models with moderate inequality reasoning generally show significant efficiency gains in this case. However, there are known cases of models that require a depth of up to 10 to show correct behaviour. The following hypotheses can be considered concerning the correct depth-limit for models with certain properties:

- Each transitivity operation adds, replaces, or removes at least one variable. If in a model the most complex inequality relation has n variables this relation can be completely transformed in n steps and thus a limit of $n+1$ could be used. However, experimental evidence suggests that occasionally more derivations are required.
- Each partial ordering has a certain 'diameter' d and it can be traversed in d transitive inferences. These measures may very well correlate to the number of landmarks in the quantity spaces used. So a limit of d should be used. If this hypothesis holds, then the problem is how to calculate the maximum diameter given a model.

4.5. *Inequality reasoning engine algorithm*

The inequality reasoning engine implements the functionality discussed in the previous sections. Fig. 11 gives an overview of the algorithm used. It can be invoked by adding as input either a new inequality relation or a new correspondence. In the latter case, the new

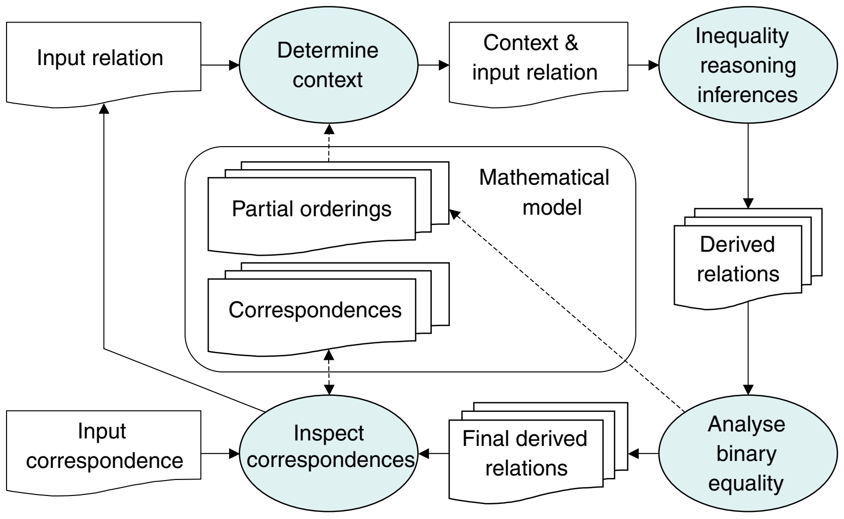


Fig. 11. Overview of the inequality reasoning, which is an important procedure in the Garp3 reasoning engine shown earlier in Fig. 8.

correspondence is checked to see whether it fires. If it does, then this results in an input relation, which brings us to the former case. Here, firstly, the applicable partial ordering is selected by the *Determine context* inference. At this point the input relation is also checked to see if it is already derivable. In that case, the algorithm finishes immediately. In the case of a truly new relation, it is sent on to the core inequality reasoning inference along with the specific partial ordering of related variables. This *Inequality reasoning inferences* step, which is further discussed below, applies the inferences discussed in Section 4.2. It returns a set of newly derived relations that is the input for the *Analyse binary equality* step. This step applies the principle of shared pointers for equal variables V_1 and V_2 if $V_1 = V_2$ is known (Section 4.4). As a result the Garp3 mathematical model is updated and the resulting trimmed set of newly derived relations is input for the *Inspect correspondences* step, which checks for firing conditional relations. If such a relation is found then the consequential relation is a new input relation and the algorithm starts a new cycle. Notice that sets of new relations are processed one by one, using the same procedure as if each relation was a separate call.

For new inequality relations the algorithm finishes with three possible outputs:

- The input relation is a derivable relation: it is already inferred or known.
- The input relation causes a contradiction.
- The input relation is added to the mathematical model.

Some relations tie multiple partial orderings together. In that case, these contexts can be merged by simply taking the union of the relation sets. Any new inferences between the relations in the contexts do not need to be made since all relevant inferences will be made based on the new input relation.

Because of the restrictions mentioned above the application of the general inequality reasoning inferences is not straightforward. The algorithm used by Garp3 is illustrated in Fig. 12. Each input relation is added to the set of base relations (explicitly represented in the model) and the set of derivable relations (base relations and relations inferred by the reasoning engine). The input relation is then combined with an empty list of parent relations and sent to make an *Inference* with each base relation. Anti-symmetry or transitivity is applied, followed by basic algebraic simplification. The resulting relation is *Tested* against the following criteria:

- Simplifiable (Section 4.4)
- Not circular (result relation is not a parent)
- Not derivable (no known or already inferred relations)
- Not evident (no tautologies)
- Not invalid (no contradiction).

If the relation resulting from the *Inference* step passes these tests it is added to the set of new relations with the input relation and the used base relation being added as its parents. The relation is also added to the set of derivable relations, but only if it is not a relation involving addition or subtraction. This last restriction keeps the set of derivable relations small for fast testing on the latter. A downside is

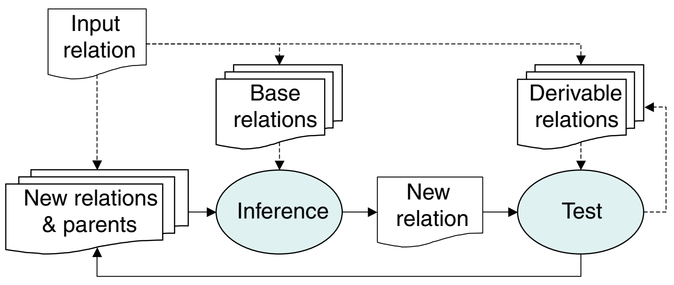


Fig. 12. Inequality reasoning inference a specific step within the inequality reasoning procedure shown in Fig. 11.

that these ‘not saved relations’ may be derived multiple times, but our experience is that the advantages outweigh the disadvantages. The loop continues with the initial input relation until all base relations have been tried. Then the next new relation (the first new derived relation) is combined with all base relations. This process continues until no more new derivations are made and all new relations have been combined with the base relations. If active, the depth-limit mechanism (Section 4.4) can stop the process for relations with more parents than this limit.

This algorithm effectively has the outcome that each new relation (inferred or from model input) is combined with all base relations (from model input) present at the time of inference. Since all base relations were added one by one this implies that all base relations are combined with each other. The *combine new with asserted* heuristic mentioned in Section 4.4 therefore primarily prevents inferred relations from being combined with each other. The complete application of this concept has proven to be too strict for some models. Therefore the inference algorithm is invoked a second time after each added input relation is processed. Now, the result of this process, the set of newly derived binary relations (no additions/subtractions) is combined with itself (replacing the base relations, see Fig. 12). This only causes a small subset of the inferred relations to be combined, but because the ultimate parent relation is the same, the likelihood of having related variables and meaningful inference is high.

5. Find states/state construction

Three phases can be distinguished in the Garp3 state construction procedure called *Find states* (Fig. 8). Input for these phases is a partial state description in the form of initial input scenarios or ‘transition scenarios’ (specifications resulting form a state transition, Section 6). Output is a set of resulting ‘successor’ states and the transitions from the predecessor state or input scenario.

5.1. Selection and application of model fragments

Selection and application of model fragments is a complex task since they may be interdependent. Each model fragment can possibly assert information that is the condition for another fragment to fire, so a naïve method of going over each fragment once is not suitable. Our solution to this problem is described below.

5.1.1. Model fragment selection

Scenarios and model fragments add new information to the model and these new ingredients can trigger the application of new model fragments. Such ingredients are referred to as *conditionals*. Structural conditionals (entities, agents, configurations, attributes, assumptions, and quantities) are distinguished from behavioural conditionals (value assignments, derivative assignments, and inequalities). The causal ingredients (correspondences, influences, and proportionalities) cannot be conditionals; they can only be added to the model as consequences in model fragments. Model fragments may also depend on another model fragment being active, therefore accepted model fragments are also considered to be structural conditionals.

Model fragment selection is done using the algorithm illustrated in Fig. 13. Firstly, the scenario (by definition) supplies a set of conditionals; the *Add consequences* step adds all its behavioural ingredients to the mathematical model and adds all its structural ingredients to the set of un-inspected ingredients. The *Select ingredient* step takes one element from this set using a first-in, first-out strategy. Notice that this order only has an impact on the order in which model fragments are accepted and not on the final model result. The *Test structural conditions* step matches this single structural ingredient with the model fragments in the library. A model fragment is a candidate if the inspected structural ingredient is a condition and all other needed structural conditionals are in the set of

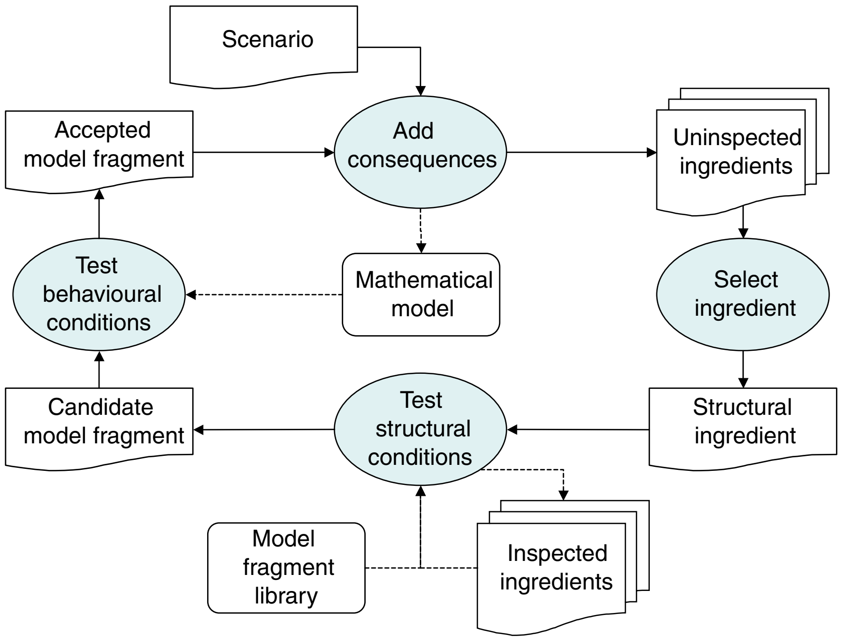


Fig. 13. Algorithm for model fragment selection. The first step of *Find states* shown in Fig 8.

already inspected ingredients. So each structural conditional in the model triggers a round of examination in which all model fragments in the library are inspected. And a model fragment turns into a candidate only when the last of its structural conditionals is considered. Note that multiple candidate model fragments can be found upon examination of a single conditional. The *Test behavioural conditions* step tests each candidate using the mathematical model and the inequality reasoning engine. There are three possible outcomes:

1. If the behavioural conditions are known, then the model fragment is accepted and its consequences are applied by the *Add consequences* step.
2. If the behavioural conditions cause a contradiction, then the model fragment is rejected and not inspected anymore in this branch of the state search.
3. If the behavioural conditions are not known but also not contradictory, then the model fragment is retested later to see if the behavioural conditions have become true.

When all candidates are assessed and applied (if appropriate), the model fragment selection process continues with the *Select ingredient* step, until there are no more un-inspected behavioural conditionals. At that point all pending model fragments (following from point 3 mentioned above) are re-tested and applied if appropriate, possibly initiating the selection process again. Garp3 features a mechanism to make reasoning assumptions on model fragments still pending after this final re-test. This topic is discussed in the next section.

5.1.2. Model fragment reasoning assumptions

When no more progress is being made in the selection process, reasoning assumptions can sometimes be made. Reasoning assumptions are implemented as alternative procedures in Garp3. In this case candidate model fragments with behavioural conditions that are not conflicting with the current partial state description can be accepted on the basis of such an alternative procedure. This reasoning assumption consists of adding the behavioural conditions of the model fragment to the mathematical model. The procedure uses assumption branches and levels, to deal with alternative mutually exclusive assumptions and assumptions over different behaviour aspects. This way all possible behaviours of a system are generated. Moreover, when a reasoning assumption excludes another pending candidate model fragment because its behavioural conditions became contradictory, this second candidate can be assumed in an alternative branch. Using reasoning assumptions is optional; each individual procedure can be turned off (and on) via a simulation preference in Garp3. Switching the reasoning assumption off would in this case result in finding only fragments with conditions that can be fully inferred.

Derivative information can also be a behavioural conditional and therefore these ingredient types can become reasoning assumptions as well. Such assumptions can be problematic because derivatives are typically determined by applying the causal model (later in the state construction procedure). If not all possible derivatives are modelled in alternative model fragments then the state search may reach a dead end if the causal model does not correspond with the assumptions taken. To handle this situation, an extra simulation preference is present to specifically turn off reasoning assumptions concerning derivatives and derivative inequalities.

While derivatives are ultimately branched and specified if they are ambiguous (Section 5.2.2), magnitudes may remain unknown unless fully specified by the model. An assumption mechanism can be used to generate all magnitudes for quantities in these cases. When activated this mechanism acts in the following two situations. Firstly, when the user specifies this behaviour for a quantity in the scenario, and secondly, when a magnitude is determined by an addition or a subtraction that is ambiguous. This mechanism ensures that quantity magnitudes do not remain unknown in a sparsely specified model and that the full behaviour is still generated. Note that having many extra assumable magnitudes is a computational burden and therefore this preference is turned off by default.

5.1.3. Model fragment consequence application

Once a model fragment is accepted its consequences are applied. The application of structural ingredients is done by instantiation of the generic concepts described in the model fragment with the matching information present in the partial state description. The application of behavioural ingredients is simple in the case of the causal primitives. These are added to the partial state description and will be evaluated in the *influence resolution* procedure (Section 5.2). The remaining behavioural ingredients (magnitudes, derivatives, inequalities, and correspondences) are added to the mathematical model of the partial state description and all mathematical consequences are computed.

Some model fragments may have contradictory consequences. In that case the state search fails because this situation indicates an inconsistent model. However, if the contradictory model fragment is applied on the basis of a reasoning assumption at some level (Section 5.1.2), then only the current assumption branch fails and alternative assumptions at this level may be explored if present. In the case of a model fragment firing normally, the state search ends with the message of an inconsistent model.

5.2. Determining state dynamics (Apply causal model)

Once the model fragment selection process is finished, the state description is complete with regard to the structural ingredients and also all information on quantity values is known. At this point the derivatives, and in some cases 2nd order derivatives (Section 5.2.3), are determined using influence resolution on the causal model (Section 5.2.2), as well as by using built-in exogenous behaviour patterns (Section 5.2.1). Fig. 14 outlines the context of this process. Exogenous quantities should always occur at the start of causal chains and therefore their dynamics are determined first.

5.2.1. Exogenous quantity dynamics

Garp3 has seven built-in behaviour patterns for exogenous quantities (Bredeweg et al, 2007). These allow the model to be connected to processes outside the model scope. These behaviour patterns, given in Table 7, are generated by a procedure that sets the exogenous quantity derivative according to the current quantity value and the associated quantity space. All changes of exogenous derivatives happen explicitly by means of a special purpose termination (Section 6.1).

Note that for single states most exogenous patterns are ambiguous, as in these cases all possible states with correct derivatives are generated. The full behaviour with correct transitions is generated as the state-graph is completed. For example an exogenous quantity with 'parabola' behaviour currently having the middle magnitude of its quantity space, will initially produce an increasing and a decreasing state. A reduction of ambiguity can be achieved by specifying the quantity's derivative in the initial scenario.

5.2.2. Influence resolution

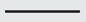
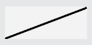
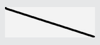
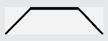
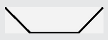
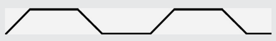
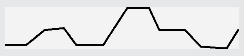
In this process the correct derivative for each quantity is inferred. To determine the effects of the causal model all influences (either direct influences or proportionalities) on each affected quantity must be considered. Notice that in the case of a chain of proportionalities to infer the derivative of a certain quantity, information about the derivative of the propagating quantity is required. In other words, at the start of the procedure many propagating quantities may lack information about their derivatives, which needs to be inferred first. To perform this process of influence resolution efficiently, the causal model is therefore first sorted in such a way that causal chains are lined up as much as possible. Because the causal model does not need to be a perfect order (and it may contain loops) the sorting procedure is not absolute. Therefore, the resolution mechanism reconsiders the unresolved parts as long as some progress is made that may propagate through the causal model. As noted in the previous section, magnitudes are already determined and have an effect through direct influences. Hence, these are always considered first. Note that proportionalities can also form the start of a causal chain if these initial quantities are exogenously determined. Finally, a user preference is available in Garp3 that provides an alternative way of assuring that all causal paths start with known magnitudes and derivatives. This mechanism assumes all unknown and uninfluenced influences to be zero (either magnitude or derivative), stating that they 'have no significant influence on the system behaviour'.

The effect of a single direct influence or proportionality can be determined given the following definitions (Section 3.1):

- Direct influence $I+$(Q2, Q1) causes the quantity Q2 to increase if the magnitude of Q1 is positive, decrease if it is negative, and remain steady when it is zero. For a negative influence, $I-$, the opposite occurs.
- Proportionality $P+$(Q2, Q1) causes Q2 to increase if Q1 increases, decrease if Q1 decreases, and remain steady if Q1 remains steady. For a negative proportionality, $P-$, the opposite occurs.

The effect of multiple direct influences or multiple proportionalities is determined by looking at their combined effect. For direct

Table 7
Exogenous quantity behaviour types in Garp3 (Bredeweg et al, 2007), which can be used to specify external influences on a system.

Type	Description	Diagram
Steady	Derivative starts zero and remains zero.	
Increasing	Derivative starts plus and becomes zero in upper point of quantity space.	
Decreasing	Derivative starts minus and becomes zero in lower point of quantity space.	
Parabola (positive)	Derivative starts plus, becomes zero in upper point of quantity space, then becomes minus.	
Parabola (negative)	Derivative starts minus, becomes steady in lower point of quantity space, then becomes plus.	
Sinusoidal	Derivative changes between minus and plus in the extreme values of the quantity space, intermediately becoming zero in these extremes.	
Random	Derivative may change freely within continuity regime.	

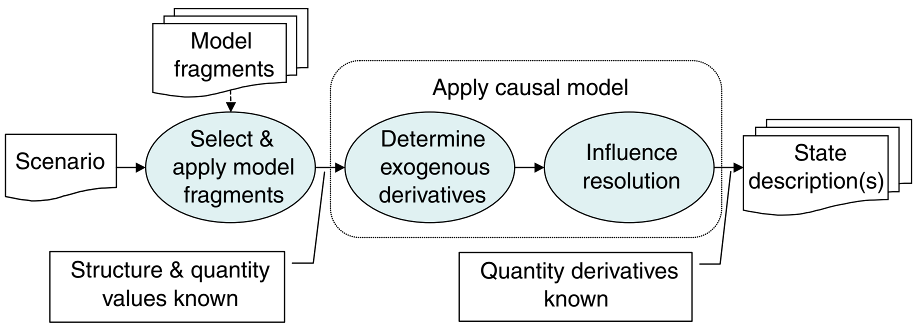

Model fragments
Apply causal model
Scenario
Select & apply model fragments
Determine exogenous derivatives
Influence resolution
State description(s)
Structure & quantity values known
Quantity derivatives known

Fig. 14. Apply causal model inference in context. The second step of *Find states* shown in Fig 8.

influences the nature of the following relation is tested against the Garp3 internal mathematical model:

$\Sigma I+ \; rel \; \Sigma I-$.

With: $rel \in \{>, \geq, =, \leq, <, ?\}$. This weighs the effect of the sum of positive influences against the effect of the sum of negative influences to determine the resulting effect. This method, called the 'influence balance', subsumes simple sign addition. Because the nature of each cause is unknown in Qualitative Reasoning, this comparison is done under the assumption that the processes involved are similar types of processes having comparable effects. The affected quantity may be assigned a particular derivative form the quantity space {*Minus, Zero, Plus*}, or multiple states are generated when a range is possible in case the relation is not strong, i.e. $\{\geq, \leq, or\ ?\}$. This causes *branching* in the state search.

For multiple proportionalities a similar approach is taken. The nature of the following relation is tested against the Garp3 internal mathematical model:

$\Sigma P+ \; rel \; \Sigma P-$

With $rel \in \{>, \geq, =, \leq, <, ?\}$. This again weighs the effect of the sum of positive proportionalities against the effect of the sum of negative proportionalities to determine the resulting effect. The affected quantity may be assigned a particular derivative or state search *branching* may occur. Again, the comparison is performed under the assumption that 'similar causes have similar effects'. For the proportionalities the argument is as follows: In general, for a quantity that is functionally proportional to multiple other quantities, it is not true that the comparison of derivatives determines which effect is dominant, because the nature of the underlying function is unknown. However, in many cases (e.g., addition $A=B+C$, where $P+(A, B)$ and $P+(A, C)$) the role of both propagating quantities in the functional proportionality is equal. In such cases the comparability assumption holds and one would like to be able to express the dominance of either influence. A practical argument validating the default use of this assumption is that inequality information between derivatives is typically only present if it is explicitly added by the user. Therefore the assumption only comes into play in models where the user specifically chooses to add information determining the dominant influence.

5.2.3. 2nd order derivatives

To correctly capture the behaviour of a system, 2nd order derivatives are sometimes needed. Consider the situation in which two opposing direct influences create an ambiguous effect on a quantity (Fig. 15). In that case three states are generated, each with a certain derivative for this influenced quantity (Fig. 16).

Each state has a different derivative for the ambiguously influenced quantity and represents a certain relation between the two influencing

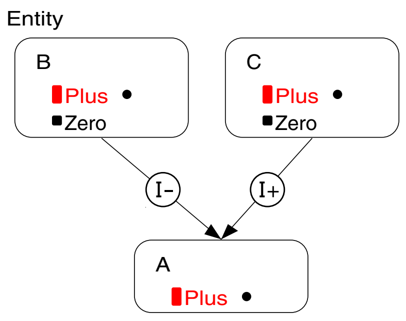


Fig. 15. An ambiguous situation caused by two opposing direct influences. The quantities *B* and *C* both have a positive magnitude, but *B* imposes a negative direct influence on *A*, while *C* imposes a positive direct influence on *A*.

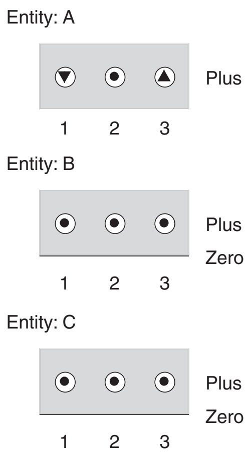


Fig. 16. Value history for the situation shown in Fig. 15. This situation is ambiguous which results in quantity *A* obtaining three possible derivatives: *Negative* in state 1 (decreasing), *Zero* in state 2 (steady), and *Positive* in state 3 (increasing). Quantities *B* and *C* are steady in all states.

quantities. In state 1, the inequality $B>C$ holds, in state 2, $B=C$ holds, and in state 3, $B<C$ holds. In the example, quantity *B* and *C* are stable; they have a zero derivative. Therefore, commonsense suggests that the relation between them should not change over a transition, and consequently the derivative of *A* should also not change. This knowledge about the behaviour of the derivative of quantity *A*, given the behaviour of its influencing quantities *B* and *C* can be captured in the 2nd order derivative of *A*. In the example the 2nd order derivative generates constraints ensuring the correct behaviour.

In other cases, 2nd order derivatives are needed to trigger explicit derivative terminations. Suppose there is an ambiguous derivative δQ generating three states: (1) $\delta Q=$*Minus*, (2) $\delta Q=$*Zero* and (3) $\delta Q=$*Plus*. Suppose furthermore that this derivative becomes unambiguously *Minus* over an otherwise unrelated state transition. Then this causes a dead end for state 3, because of breaking the *continuity* regime (discussed in Section 6.3): the transition from $\delta Q=$*Plus* directly to $\delta Q=$*Minus* is illegal. This is a problem unless there is a *derivative termination* from $\delta Q=$*Plus* to $\delta Q=$*Zero* that can happen first. These derivative terminations, needed to make the transition smoothly, can be generated using 2nd order derivatives.

To calculate 2nd order derivatives a procedure similar to the regular influence resolution is used. Fig. 17 shows a typical causal model, starting with a direct influence, followed by two proportionalities, together implementing a feedback loop. The calculation of the effects of this causal model is shown in Fig. 18.

As can be seen in these figures, 2nd order derivatives can only be calculated if at least one direct influence is present in the causal chain to generate the first 2nd order derivative information using the derivative of the influencing quantity. Also, all of these influencing quantities must have a determined derivative. Therefore the procedure is carried out right after the regular influence resolution step. Hereby, its effects are allowed to propagate. In the case of ambiguity no *branching* is done and the 2nd order derivative remains unknown since *branching* would result in numerous states that are hard to distinguish from a user perspective.

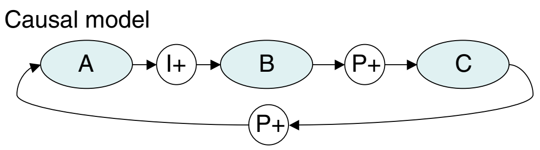


Fig. 17. A typical causal model with a feedback loop: *A* directly influences *B*, which propagates via *C* back to *A*.

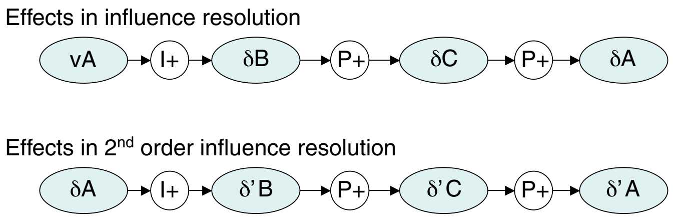

Effects in influence resolution
vA
I+
δB
P+
δC
P+
δA
Effects in 2nd order influence resolution
δA
I+
δ'B
P+
δ'C
P+
δ'A

Fig. 18. Determining causal effects. In this example *v* indicates a magnitude, δ indicates a derivative, and δ′ indicates a 2nd order derivative. The figure illustrates how 2nd order information can be inferred from 1st order information.

Sometimes however, a 'weak' result: {≥ or ≤} can be derived. This weak 2nd order derivative is recorded because a weak 2nd order derivative may still supply strong constraints as will be shown below. Notice that a consequence of not branching ambiguous 2nd order derivatives is that these are always determined fully by the 1st order derivatives and the causal model.

Singularly influenced quantities have no need for any of the 2nd order derivative functionality because of the following reasons:

- Constraints on the derivative are not necessary since the derivative is directly dependent of the single influence. Thus if the influence does not change, the derivative will not change either.
- Derivative terminations are not necessary since the derivative can always change directly with the changing single influence. There is no hidden underlying inequality that can be changing without an explicit termination happening.

Therefore, 2nd order derivatives are only actively used for quantities affected by multiple influences. However, 2nd order derivatives of singularly influenced quantities *are* still calculated, because they may be part of a causal chain needed to calculate a 2nd order derivative of a quantity with multiple influences. In the Garp3 interface 2nd order derivatives are visible in the value history as small arrows and dots next to the derivative symbol.

5.3. Compare states

The third phase of the *Find states* step of Fig. 8 completes the state description by determining if the state is a new state or if it is equivalent to an existing state (also an appropriate transition link from the predecessor state is generated). To compare states, a matching process is applied that compares all model ingredients in the state description to those in all existing states. Since magnitudes and derivatives are most descriptive of a state in typical models these are tested first. Inequality relations are matched last. The inequality relations added by the 'Close' step (to ensure continuity, Section 6.3) do not need to be an exact match. For these relations, consistency is sufficient, because states resulting from different predecessor states will have different continuity constraints, even though they represent the same behaviour.

Determination of the successor states from a transition scenario by fully specifying all resulting states is computationally expensive. Hence, mapping such transition scenarios (or partial state descriptions) directly onto existing states may seem an alternative and even better solution because it saves work. Unfortunately, this approach is not suitable because it may produce erroneous results. Notice that each transition scenario may potentially lead to multiple successor states and if matched onto a single existing state, valid behaviour may be lost. There are various reasons that only a subset of the potential successors of a transition scenario is already in the partial state-graph. For instance, these states may result from an initial scenario with constraints. Or, the state can be reached via various paths where some paths are more constrained than others, because of continuity.

6. Find transitions/state transitions

The *Find transitions* step determines how states change (Fig. 8). Three sub-steps are distinguished here:

- Terminate (*Find changes*). Find model ingredients in the current state that may change and thereby cause the state to terminate. These elements are called terminations and they consist of the current and the changed ingredient.
- Order (*Combine changes*). Analyse the terminations to determine the order in which they may happen, and determine all valid combinations.
- Close (*Apply continuity*). Apply the continuity rules to each combination to produce a complete transition scenario. A transition is found if a transition scenario leads to a valid state.

Note that the *Find transitions* step itself produces only transition scenarios, and that a valid transition is only found once the full resulting state is determined by the *Find states* step. So the Close step is broader than the *Apply continuity* step found in Fig. 8 because it includes also the full *Find states* inference. The following sections discuss the Terminate, Order and Close steps.

6.1. Terminate: determining potential changes

A state (representing the behaviour of modelled system) terminates once the system shows one or more changes in behaviour (compared to that described by the state). These changes in behaviour include: A changing quantity magnitude (value termination), changing inequality relations between quantities (inequality termination), and quantities that stop or start changing (derivative termination). Additionally, an exogenous behaviour pattern may enforce a termination (exogenous terminations). Note that if an inequality is used to describe the current quantity magnitude, then the necessary inequality change will be incorporated in the value termination.

The conditions for terminations are as follows. A value termination is produced if the derivative indicates that the quantity magnitude is moving (either increasing or decreasing) towards the next adjoining magnitude in its quantity space. There is also a simulation preference that can be used to force the Garp3 engine to generate value terminations even if the derivative is unknown. An inequality termination is produced if either one (or both) of the derivatives of the involved quantities indicate that the inequality should change from $>$ to $=$, from $=$ to $<$, or vice versa. There is also a simulation preference that can be used to force the Garp3 engine to generate inequality terminations even if both derivatives are moving in the same direction so that their combined effect is unknown. The conditions for derivative terminations concern 2nd order derivatives and are given in Table 8. Note that in the conditions marked with a *, a change is also possible. However, we decided not to let these cases trigger a termination analogous to the case of value terminations that are not triggered if derivatives are not definitively known.

6.2. Order: precedence and validity of combinations of changes

In the context of Qualitative Reasoning, a perfect order in which terminations will happen cannot always be given. The *epsilon ordering* concept (de Kleer and Brown 1984) provides information on terminations taking precedence over others. Given this distinction, the remaining terminations can potentially happen in all possible combinations. Hence, a mechanism is necessary to determine these sets. The brute force solution of taking the cross-product is unfeasible since this potentially results in very large sets of transition scenarios making simulations intractable. For *n* terminations there would be *n!* transition scenarios that would need to be applied. The Garp3 ordering procedure is illustrated in Fig. 19.

Table 8
Derivative terminations. Derivatives may change depending on 2nd order information. If, for instance, the derivative is steady, $\delta(X) = \text{Zero}$, and the 2nd order positive, $\delta'(X) > \text{Zero}$, then the derivative will change, $\delta(X) > \text{Zero}$. The * refers to changes not definitively known, which are therefore not considered.

Derivative / 2nd order Derivative	$\delta(X) > 0$	$\delta(X) = 0$	$\delta(X) < 0$
$\delta'(X) > 0$		1: $\delta(X) > 0$	4: $\delta(X) = 0$
$\delta'(X) \geq 0$		*	*
$\delta'(X) = 0$			
$\delta'(X) \leq 0$	*	*	
$\delta'(X) < 0$	3: $\delta(X) = 0$	2: $\delta(X) < 0$	

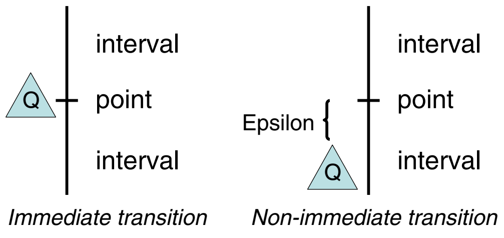


Fig. 20. Epsilon concept. Quantity Q is increasing in a point (left-hand side), and increasing over an interval (right-hand side). These transitions are immediate and non-immediate, respectively.

The epsilon ordering concept distinguishes between immediate terminations (from a point, and from equality) and non-immediate terminations (to a point, and to equality). This concept is based on the idea of dealing with quantities behaving as continuous functions of time and that a point occupies no space. Therefore if a quantity is on a point and it changes, it will leave that point immediately without any passage of time. On the other hand, a quantity changing towards a point will always have some epsilon amount of space between itself and the point. Therefore the transition to that point is not immediate. As a result of this, immediate transitions take precedence over non-immediate transitions. The idea is illustrated in Fig. 20, which shows quantity Q is increasing, on a point (left-hand side) and in an interval (right-hand side). Note that a similar argument can be made for inequality terminations and derivative terminations (that is, 'from equality' precedes 'to equality').

The *Epsilon ordering* step (Fig. 19) constructs a set of immediate terminations and non-immediate terminations. If the first set is not empty then it will be used and the non-immediate terminations will not be used. In the other case the non-immediate terminations will be used.

Given a set of terminations, the following three inferences in the ordering step determine which combinations are valid. This activity is 'looking ahead' towards new states and therefore requires information sources to remain valid over state transitions. The following sources are considered to be stable:

- Mutually exclusive termination types:
  - Exogenous terminations in opposite directions
  - Assumed terminations in opposite directions
- Correspondences
- Mathematics:
  - Behavioural ingredients without a firing termination
  - Inequalities between landmarks
  - Inequalities involving additions or subtractions

Note that both correspondences and inequalities involving additions or subtractions could in principle be changing over transitions, and that these are heuristic assumptions that modellers should be aware of.

We use the notion of *combination concept* to refer to an expression assigning validity and/or invalidity to a set of possibly co-occurring terminations. When applying the *combination concepts* following from the sources mentioned above, considering all terminations at once is still computationally inefficient. Therefore, the algorithm inspects only a subset of terminations at each step and supplies some information about this subset in the form of *combination constraints*. This information is later integrated in the *Constrained cross-product* step. In order to do this, *combination constraints* must have a predetermined form and scope. In Garp3 they are given the form of a table of valid and invalid combinations. Regarding scope, each combination of terminations is 'innocent until proven guilty', and a *combination concept* cannot independently prove a combination to be valid, but it can independently prove a combination to be invalid. *Combination concepts* must have this property, and in their discussion (which follows below) it is shown for each one that this requirement is met.

Assumed terminations as well as exogenous terminations may be mutually exclusive. For example, a quantity with an unknown derivative may change to a higher or a lower magnitude. Also, an exogenous

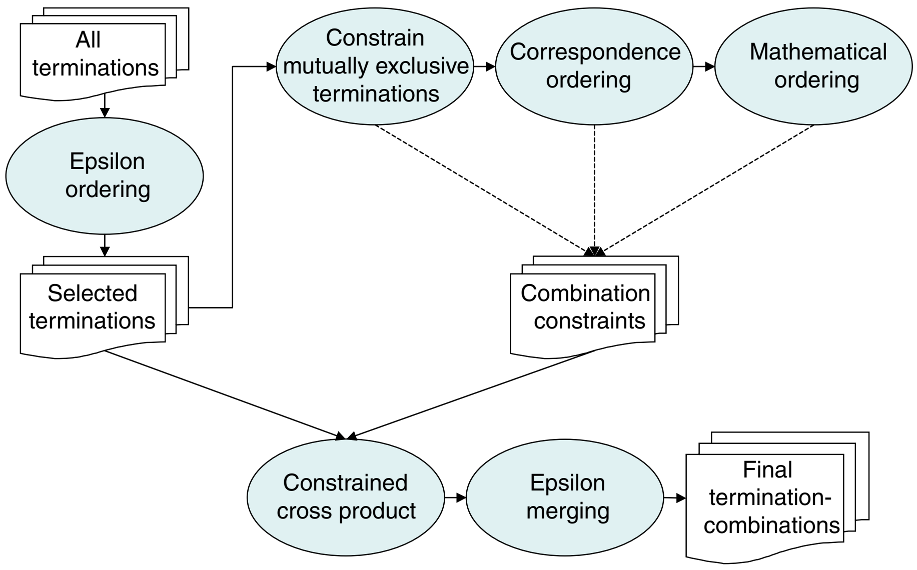


Fig. 19. The *Ordering procedure*. After doing epsilon ordering the remaining terminations are analysed to produce constraints on valid combinations. These are then used to determine sets of combinations, which are finalised by letting all epsilon immediate terminations occur simultaneously.

quantity with the 'random' behaviour pattern that is in the current state stable may become increasing or decreasing in the next state. These are two different behaviours for the same ingredient and therefore their combination is inconsistent. The *Constrain mutually exclusive terminations* step (Fig. 19) applies this *combination concept* and generates *combination constraints* for mutually exclusive terminations. Table 9 shows the combination constraint table for two such terminations T1 and T2. Separately these terminations are valid, but their combination is an invalid item in the table.

The *Correspondence ordering* step (Fig. 19) generates *combination constraints* using correspondences, which are a very valuable information source. This *combination concept* works as follows. Each correspondence is considered at the level of a single directed value correspondence (conditionally linked magnitude pairs, Section 4.3). The generic situation is that there are two quantities Q1 and Q2 with magnitudes V1A and V2A. There may be one or two value terminations involved: T1 from V1A to V1B, and T2 from V2A to V2B. And there is a correspondence concerning at least one of these four magnitudes (Fig. 21 illustrates this situation). In this situation, any termination combination that has resulting magnitudes for Q1 and Q2 that contradict the correspondence present is inconsistent by definition. The resulting *combination constraints* in the example of Fig. 21 are that {T1, T2} is invalid, but that {T1} and {T2} are valid. In other words, T1 and T2 may happen individually, but not at the same time. T1 would result in Q1 having magnitude V1B, which in turn would require Q2 to have magnitude V2A because of the correspondence. However, the latter is inconsistent with T2 happening, because T2 implies that Q2 gets magnitude V2B.

In general, five correspondence *combination concept* situations can be distinguished, as shown in Table 10. These are all based on the principle outlined above. In this table the current magnitude of Q1 is V1A and the current magnitude of Q2 is V2A. One or two terminations may be present, one for each quantity. *X* is used in case 5 to indicate a magnitude not equal to V2A and not within reach of Q2, either because the termination for Q2 does not involve *X* or because there is no termination for this quantity at all. Note that in case 2 and 5 a single termination is invalid; Garp3 immediately removes such terminations from the total set to prevent any further inference effort being spent on them.

A similar procedure is used for derivative terminations, with one difference however. For magnitudes, terminations are the only reason for change, but for derivatives this is not the case. Derivatives may also be allowed to change over transitions by the continuity regime if they do not have a derivative termination (Section 6.3). This leads to some exceptions. A correspondence may falsely seem to remain active, for example because the conditional derivative has no termination. Therefore correspondences concerning derivatives without a termination are not considered.

The *Mathematical ordering* step (Fig. 19) is the next *combination concept* that is applied. This inference uses the inequality reasoning engine to determine the validity of termination combinations by evaluating their mathematical consistency. Fig. 22 illustrates the idea. The two quantities Q1 and Q2 are both at a point (V1A and V2A) and have an equality Q1 = Q2. Three terminations are present: T1 from V1A to V1B, T2 from V2A to V2B and T3 from Q1 = Q2 to Q1 > Q2. Table 11 gives the *combination constraints* derived in this context. These constraints express that if either quantity changes its magnitude, then the equality should change as well.

Table 9
Mutually exclusive terminations table example. T1 and T2 may each happen individually, but not together.

Valid	Invalid
{T1}	{T1, T2}
{T2}	

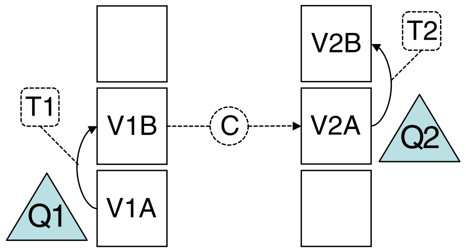


Fig. 21. Asynchronous correspondence activation. T1 cannot co-occur together with T2. Separately, the terminations are both valid.

To implement the mathematical *combination concept*, the cross-product is taken (actually the 'constrained cross-product', as discussed below) and each combination in this set is tested for consistency. The combination constraints table is filled according to the results from these tests. Note that by taking the constrained cross-product, previous *combination constraints* are taken into account. This way the mathematical *combination concept* does not waste effort on combinations that already have proved invalid in previous steps.

Since the application of the mathematical *combination concept* is relatively expensive, sets of terminations are carefully selected. Firstly, each inequality termination (concerning two quantities) triggers a check to verify that there is at least one inequality relation between landmarks known that relates two landmarks either concerning the current magnitudes of the two quantities, or the magnitudes attainable in potential value terminations. At least one of these relations is needed for a contradiction to possibly occur. Note that this is the case in the example (Fig. 22), as V1A = V2A is derivable. Secondly, each inequality involving addition or subtraction is considered a stable information source and can therefore form the basis of a test, if there are terminations concerning the involved quantities.

The *Constrained cross-product* step (Fig. 19) has the task of efficiently applying the combined knowledge of all *combination constraint* tables produced by the previous inferences. To achieve efficiency, firstly all *combination constraint* tables that have no invalid items are deleted, because these supply no information. Secondly, we consider an item in a combination constraint table to be 'fatal' if there is no valid superset item in the table to prove it wrong. For example, the invalid item in Table 9 is fatal, while Table 11 has no fatal items (all invalid items are subset of the valid item {T1, T2, T3}. Fatal items are important because they convey strong information. If such an item matches a combination then this combination can be rejected without further investigation. The algorithm for generating a constrained cross-product is given below. The general idea is to generate a full cross-product and test each element in it. Testing is done by first checking applicability (subset) of fatal items and then checking each *combination constraint* table. To do this, the set of the largest (superset) applicable items is found and if it contains an invalid item then this element of the cross-product is rejected.

```
Let CP be a cross-product of a set of terminations
Let CS be a set of combination constraint tables
Let F be a set of fatal combinations

For every combination $TC \in CP$:
    if: there is an $X \in F$ where $X \subseteq TC$ then reject TC
    else: For every combination table CT in CS:
        find all items Xi where:
            $Xi \in CT$ and $Xi \subseteq TC$ and there is no Xj where $Xi \subset Xj$
            ($Xj \in CT$ where $Xj \subseteq TC$)
        if: there is an X left labelled invalid then reject TC and exit For
    if: CS is fully inspected then accept TC
if: CP is fully inspected then return the set of all accepted TC
```

Table 10
Correspondence combination concept. All situations and their resulting constraints are shown. See main text for explanation.

Nr.	Description	Termination	Correspondence	Valid	Invalid	Comment
1	Possible Deactivation	T1, T2	V1A → V2A	{T1}, {T1, T2}	{T2}	Q1 may move individually, or in combination with Q2.
2	Impossible Deactivation	T2	V1A → V2A		{T2}	T1 not present, remove T2
3	Synchronous Activation	T1, T2	V1B → V2A	{T1}, {T2}	{T1, T2}	See text and Fig. 21
4	Asynchronous Activation	T1, T2	V1B → V2B	{T2}, {T1, T2}	{T1}	Q2 may move individually, or in combination with Q1.
5	Impossible Activation	T1	V1B → *X*		{T1}	T2 may be present but not relevant, remove T1

The epsilon concept has another consequence besides precedence, which is that immediate terminations will happen all at once as a single combination. The *Epsilon merging* step (Fig. 19) implements this concept. It groups immediate terminations by taking the constrained cross-product and removing all combinations for which there is a superset combination. This way only the largest unique combinations will remain.

The order of the application of inferences as shown in Fig. 19 is carefully chosen. The argument for this particular sequence is based on the following two criteria:

- Semantics: Does a particular sequence lead to different, possibly incorrect, results?
- Efficiency: Does a particular sequence have an effect on efficiency?

These criteria lead to the following approach. *Epsilon ordering* and *Epsilon merging* both reduce the number of states. They should therefore be placed at the beginning of the sequence from an efficiency point of view. Moreover, *Epsilon ordering* produces correct results, immediate and non-immediate terminations should not be combined. Hence, it is in fact applied first. *Epsilon merging* potentially produces incorrect results if done before the procedures that check for incorrect combinations (such as *Mutually exclusive terminations*). A strong merge is thus not done at the beginning. Instead, a softer merge is done later. Application of the mathematical concepts is relatively expensive. It is done as late as possible to allow preceding concepts to constrain the 'search space' of the mathematical concepts. Therefore the *Mutually exclusive terminations* step (relatively simple strict split) and *Correspondence ordering* (benefits from the former) are applied before the *Mathematical ordering*. Finally, for the mathematical *combination concepts*, binary inequality terminations are assessed before assessing more complex quantity contexts, because the former concept can provide useful constraints for the latter.

6.3. Close: finalising and applying transition scenarios

The Close step applies continuity rules to each compound termination to turn it into a transition scenario. Consequently successor states are determined. This is done (as noted before) by the *Find states* step. Continuity rules specify a 'one step of freedom' for derivatives over each transition. The set of rules is given in Table 12. This regime is needed because in general derivatives do not have explicit terminations. Only quantities with multiple influences have these terminations (Section 5.2.3 on 2nd order derivatives). The reason for not using derivative terminations for all quantities is that this would result in a much larger set of single terminations, which would be an unnecessary computational burden. Note that the continuity procedure only gives freedom to derivatives that do not have an explicit derivative termination for that state. If both the normal continuity regime and a derivative termination would work on the same derivative, then a transition scenario with this derivative termination might lead to the same state as a similar transition scenario without this derivative termination.

Next to the normal continuity regime, which provides a mechanism for the correct progression of not explicitly terminating derivatives, two additional continuity concepts are applied that provide extra constraints on valid behaviour.

Firstly, the epsilon concept (Section 6.2) has a consequence for continuity. In immediate transitions, derivatives may not change to *Zero*. The rationale behind this rule is that it is a non-immediate event since it is a transition towards a point. Moreover it represents the event that the balance of influences on this quantity (whose derivative is going to *Zero*) changes from unequal to equal. And this type of inequality change is of course non-immediate too. Therefore, in any immediate transition derivatives may not change from *Minus* or *Plus* to *Zero*.

Secondly, 2nd order derivatives provide extra constraints on derivatives. Table 13 shows the constraints that are active in the successor state given a derivative and a 2nd order derivative.

As can be seen, weak 2nd order derivatives can supply strong constraints. For example, when a derivative is positive: $\delta Q > \text{Zero}$ (1st column), and it has a weak 2nd order derivative: $\delta' Q \geq \text{Zero}$ (2nd row), still a strong constraint is derived: the derivative must remain positive: $\delta Q > \text{Zero}$. An issue here is that the causal model might change over a behaviour path. If this is the case then constraints should not be applied

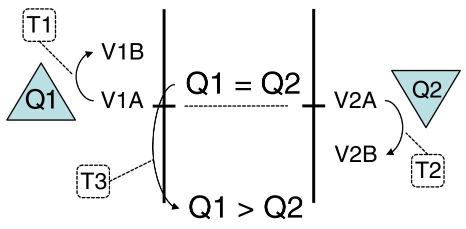


Fig. 22. Mathematical combination concept example for binary inequality. It illustrates that an inequality change may also require a change of the involved quantities.

Table 11
Combination constraints for binary inequality example. T1, T2, T3, and T1 & T2 cannot occur independently, but they can happen when they co-occur in the following combinations: T1 & T3, T2 & T3, and T1 & T2 & T3.

Valid	Invalid
{T1, T3}	{T1}
{T2, T3}	{T2}
{T1, T2, T3}	{T3}
	{T1, T2}

Table 12
Normal continuity rules. *Derivative* refers to the current state and *Constraint* refers to allowable changes when moving to the next state.

Derivative	Constraint
$\delta Q = \text{minus}$	$\delta Q \leq \text{zero}$
$\delta Q = \text{zero}$	None
$\delta Q = \text{plus}$	$\delta Q \geq \text{zero}$

at this transition. Therefore, these extra constraints are not simply added to the mathematical model immediately, but instead these are checked later, right before the new causal model is evaluated. Each constraint is applied only if all influences on the quantity in question have remained the same. Note that theoretically this change-check would not be necessary if new influences would always start at *Zero* (not having an effect) and if removed influences would always go to *Zero* first. In that case, the derivatives would still comply with the constraints posed by its predecessor's 2nd order derivative. And in the new state the new 2nd order derivative is determined using the new causal model. Such a ‘continuous’ entry and exit of processes (and external influences by agents) is a good style of modelling, but it is not enforced in Garp3. Moreover, modelling practice has shown that this theoretical position cannot be held at all times. On some levels of abstraction one might for example need to model a transition where 2nd order derivative constraints of the previous causal model are not met. This happens for example when modelling a car crashing into a concrete wall without taking deformation into account.

6.4. Fastest path heuristic

Simulations may result in very large state-graphs, thereby presenting two problems. Firstly, the amount of information may become overwhelming such that the overall picture of the system's behaviour becomes hard to perceive (cf. Bouwer, 2005). Secondly, large simulations may be computationally expensive. Garp3 has a simulation preference called the *fastest path heuristic* that brings relief in many of these cases.

Simulations often have multiple paths to each end state. In many cases these multiple paths are strictly speaking distinct, yet practically very similar in the behaviour they represent: the same set of changes will occur, only the order in which these changes take place differs. The fastest path heuristic exploits this fact. The idea of the heuristic is to give precedence to those transition scenarios that apply *as many as possible of these terminations at once*. If these transition scenarios produce a valid state then the transition scenarios that are ‘subsets’ are discarded. The fastest path algorithm is given below:

```
Let TS be a set of transition scenarios
Let $S_1$ … $S_n$ be transition scenarios
Let $T_i$ be the set of terminations in $S_i$

1) Sort TS in decreasing order to the size of T
2) Apply the maximal scenario $S_i$ and remove it from TS
3) If a valid state is produced then for every $S_j \in TS$ remove $S_j$ from TS if
   $T_j \subseteq T_i$
4) Continue with (2) until TS is empty
```

The fastest path heuristic is applied in the ‘Close’ procedure on the basis of a simulation preference that by default is inactive. However, usage should be done with caution, because in principle there is a risk of losing valid end states. A simulation may include a specific type of behaviour that is triggered only under specific circumstances and that does not include all ‘changes’ at once. Therefore, this heuristic can potentially cause valid end states not to be found. In our tested model set there was only a reduction of the number of states, but not a reduction of the number of end states. Therefore the most abstract significant behaviour was conserved. It is our impression that the fastest path heuristic is a useful feature, at the very least during the model development phase.

Table 13
2nd Order derivative continuity constraints, detailing how derivatives may change during state transition depending on 2nd order information.

Derivative / 2nd order derivative	$\delta(X)>0$	$\delta(X)=0$	$\delta(X)<0$
$\delta'(X)>0$	$\delta(X)>0$	$\delta(X)>0$	$\delta(X)\leq 0$
$\delta'(X)\geq 0$	$\delta(X)>0$	$\delta(X)\geq 0$	$\delta(X)\leq 0$
$\delta'(X)=0$	$\delta(X)>0$	$\delta(X)=0$	$\delta(X)<0$
$\delta'(X)\leq 0$	$\delta(X)\geq 0$	$\delta(X)\leq 0$	$\delta(X)<0$
$\delta'(X)<0$	$\delta(X)\geq 0$	$\delta(X)<0$	$\delta(X)<0$

7. Discussion

Garp3 does not discriminate explicitly between inequality statements that should be treated as always true in a model (e.g. inflow − outflow = net-flow), and those that may change during simulation (e.g. $T_x < T_{x\text{-boil}} \rightarrow T_x = T_{x\text{-boil}}$). According to the simulator, all inequality statements can in principle change. This has consequences for the search space, which may become large. Hence, the default simulation preference in Garp3 is to treat inequalities involving additions and subtractions as stable. In many models this assumption holds and thereby provides a mechanism to significantly constrain the cross-product of terminations. However, in some models the termination of calculus relations would be helpful. In future work, this functionality could be implemented along with a label to indicate stable model ingredients. This way, modellers would be able to specify explicitly which ingredients (specific inequalities, correspondences, etc.) should be used in the ordering procedure.

The heuristics used to aid the inequality reasoning could be further investigated to formally establish their position (e.g. concerning soundness and completeness). Especially the restriction on the recombination of derivable relations and the depth-limit on parent inferences are interesting issues for further research.

The 2nd order derivative functionality is important for Garp3 to produce sound simulations. However, complete validity cannot be guaranteed, because ambiguous 2nd order derivatives might be moving incorrectly if we would take 3rd order derivatives into account. Because of its recursive nature, this issue is in principle impossible to solve. However, practically this is not a problem for the level of detail of models built in Garp3. Ambiguous 2nd order derivatives are not branched into multiple states, as the reasoning engine generalises these states into one state so that problematic state transitions do not occur. Moreover, modelling is simplified because in many cases it is no longer necessary to explicitly model all possible inequalities between two influences.

Future work will focus on supporting collaborative modelling and model reuse, and will include a repository for uploading, indexing and downloading models and model parts. Automatically preserving model consistency and supporting model debugging are essential for this, and need further development.

8. Conclusions

This paper has presented the Garp3 workbench for building, running and inspecting qualitative models. The software has a graphical user interface for users to interact with the software, facilitating a seamless interoperability between the different modes of use. The Garp3 workbench offers easy access to sophisticated qualitative simulation software, providing users the possibility to use Qualitative Reasoning technology without having to understand low-level implementation details of such automated reasoners. An increasing number of domain experts are using the workbench to capture qualitative knowledge of system dynamics.

Central to the Garp3 software is the inequality reasoning capability. It is used to determine the applicability of model fragments, the net-effect of causal dependencies, and it optimises the search for state transitions. Particularly, the transitivity inference is worth mentioning, because it computes a full transitive closure of not a standard partial order but one that can be extended with new nodes via additions and subtractions. Heuristics are employed to keep the search space within reasonable boundaries.

Two sets of reasoning steps implement the main engine: *Find states* and *Find transitions*. The former starts by finding model fragments applicable to a scenario, by reasoning about structural and behavioural details. When all applicable model fragments have been found, the causal model is calculated, from which in turn the tendency of change is determined of each quantity (decrease, steady, or increase). When a state is fully specified, it is compared to already existing states to determine whether it is a unique state. Unique states are added to the state-graph. The *Find transitions* step implements three procedures to establish whether a state terminates or transits to a new state. Particularly the *Combine changes* step has the potential to become large. Smart 'look ahead' mechanisms known as combination concepts have therefore been developed and implemented that successfully reduce the search space to reasonable proportions. Finally, reasoning with 2nd order derivatives plays an important role in guaranteeing that the state-graphs generated by Garp3 correctly reflect the behaviour of the systems being modelled.

Although not discussed in this paper, it is worth mentioning that the Garp3 workbench is accompanied by a set of support measures for users, such as a Glossary, FAQ, and on-line modelling help pages (http://www.Garp3.org). Moreover, the workbench supports multiple languages (e.g., English, Portuguese, and others) and has means for modellers to copy and paste model ingredients between models. In addition, each ingredient created during modelling can be given a unique 'comment', which is shown throughout the workbench as a tool-tip for the ingredient. This supports users in understanding and appreciating the models.

Acknowledgements

The research presented in this paper is co-funded by the EC within FP6 (2002–2006) (project NaturNet-Redime, number 004074, http://www.NaturNet.org), and FP7 (2009–2012) (project DynaLearn, 231526, http://www.DynaLearn.eu). Jelmer Jellema and Dirk Bertels have provided important contributions to the development of Garp3. Tim Nuttle is acknowledged for his critical appraisal of the software and correcting our broken English. We are grateful to the Garp3 users who have contributed valuable suggestions, requests and ideas. Special thanks are directed to Paulo Salles for his continuous support and inspiration.

References

Araújo, C.S., Salles, P., Carlos, H., Saito, C.H., 2008. A case study on qualitative model evaluation using data about river water quality. Ecological Informatics 3 (1), 13–25.
Biswas, G., Schwartz, D., Bransford, J., The Teachable Agents Group at Vanderbilt, 2001. Technology support for complex problem solving: from SAD environments to AI. In: Forbus, K., Feltovich, P. (Eds.), Smart Machines in Education. AAAI Press/MIT Press, Menlo Park California, USA, pp. 71–97.
Bouwer, A., 2005. Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments. PhD Thesis. Faculty of Science, University of Amsterdam, Amsterdam.
Bredeweg, B., Forbus, K., 2003. Qualitative modeling in education. AI Magazine 24 (4), 35–46.
Bredeweg, B., Salles, P., 2009. Mediating conceptual knowledge using qualitative reasoning. In: Jørgensen, S.V., Chon, T-S., Recknagel, F.A. (Eds.), Handbook of Ecological Modelling and Informatics. Wit Press, Southampton, UK, pp. 351–398.
Bredeweg, B., Schut, C., 1991. Cognitive plausibility of a conceptual framework for modeling problem solving expertise. In: Hammond, K.J., Gentner, D. (Eds.), Proceedings of the 13th Conference of Cognitive Science Society. Lawrence Erlbaum, Hillsdale, New Jersey, pp. 473–479. August.
Bredeweg, B., Struss, P., 2003. Current topics in qualitative reasoning (editorial introduction). AI Magazine 24 (4), 13–16.
Bredeweg, B., Bouwer, A., Jellema, J., Bertels, D., Linnebank, F., Liem, J., 2006. Garp3 — A New Workbench for Qualitative Reasoning and Modelling. In: Bailey-Kellogg, C., Kuipers, B. (Eds.), Proceedings of 20th International Workshop on Qualitative Reasoning, pp. 21–28. Hanover, New Hampshire, USA, 10–12 July.
Bredeweg, B., Salles, P., Nuttle, T., 2007. Using exogenous quantities in qualitative models about environmental sustainability. AI Communications 20 (1), 49–58.
Bredeweg, B., Salles, P., Bouwer, A., Liem, J., Nuttle, T., Cioaca, E., Nakova, E., Noble, R.A.A., Caldas, A.L.R., Uzunov, Y., Varadinova, E., Zitek, A., 2008. Towards a structured approach to building qualitative reasoning models and simulations. Ecological Informatics 3 (1), 1–12.
De Kleer, J., 1990. Qualitative physics: a personal view. In: Weld, D.S., de Kleer, J. (Eds.), Readings in Qualitative Reasoning about Physical Systems. Morgan Kaufmann, San Mateo, California, pp. 1–8.
De Kleer, J., Brown, J.S., 1984. A qualitative physics based on confluences. Artificial Intelligence 24 (1–3), 7–83.
De Koning, K., Bredeweg, B., Breuker, J., Wielinga, B., 2000. Model-based reasoning about learner behaviour. Artificial Intelligence 117 (2), 173–229.
Dormoy, J.L., 1988. Controlling qualitative resolution. In: Mitchell, T.M., Smith, R.G. (Eds.), Proceedings of the 7th National Conference on Artificial Intelligence, pp. 319–323.
Forbus, K.D., 1984. Qualitative process theory. Artificial Intelligence 24, 85–168.
Forbus, K.D., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J., Kuehne, S., 1999. CyclePad: an articulate virtual laboratory for engineering thermodynamics. Artificial Intelligence 114, 297–347.
Forbus, K.D., Carney, K., Harris, R., Sherin, B.L., 2001. A qualitative modeling environment for middle-school students: a progress report. In: Biswas, G. (Ed.), The 15th International Workshop on Qualitative Reasoning. St. Mary's University, San Antonio, Texas, pp. 65–72.
Fromherz, M.P.J., Bobrow, D.G., de Kleer, J., 2003. Model-based computing for design and control of reconfigurable systems. AI Magazine 24 (4), 120–130.
Gentner, D., Stevens, A.L. (Eds.), 1983. Mental Models. Lawrence Erlbaum, Hillsdale.
Grimm, V., 1994. Mathematical models and understanding in ecology. Ecological Modelling 75/76, 641–651.
Guizzardi, G., 2005. Ontological Foundations for Structural Conceptual Models. PhD Thesis, University of Twente, CTIT No. 05-74, Enschede, The Netherlands.
Haefner, J.W., 2005. Modeling Biological Systems: Principles and Applications. Springer, New York, N.Y.
Hayes, P., 1979. The naive physics manifest. In: Michie, D. (Ed.), Expert Systems in the Micro-Electronic Age. Edinburgh University Press, Edinburgh, pp. 242–270.
Jørgensen, S.E., Bendorrichio, G., 2001. Fundamentals of Ecological Modelling, 3rd edition. Elsevier Science, Oxford.
Kuipers, B.J., 1994. Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge. MIT Press, Cambridge, Massachusetts.
Kulpa, Z., 1994. Diagrammatic representation and reasoning. Machine graphics & vision 3 (1/2), 77–103.
Larkin, J., Simon, H., 1987. Why a diagram is (sometimes) worth ten thousand words. Cognitive science 11, 65–99.
Mylopoulos, J., 1992. Conceptual modeling and Telos. In: Loucopoulos, P., Zicari, R. (Eds.), Conceptual Modeling, Databases, and CASE. Wiley, pp. 49–68. chapter 2.
Norman, D.A., 1993. Things That Make Us Smart. Perseus Books, Cambridge, Massachusetts.
Novak, J.D., Gowin, D.B., 1984. Learning How to Learn. Cambridge University Press, New York, New York.J.D.
Salles, P., Bredeweg, B., 2006. Modelling population and community dynamics with qualitative reasoning. Ecological Modelling 195 (1–2), 114–128.
Salles, P., Bredeweg, B., Bensusan, N., 2006. The ants' garden: qualitative models of complex interactions between populations. Ecological Modelling 194 (1–3), 90–101.
Schwarz, C.V., White, B.Y., 2005. Metamodeling knowledge: developing students' understanding of scientific modeling. Cognition and Instruction 23 (2), 165–205.
Simmons, R., 1986. Commonsense arithmetic reasoning. Proceedings of AAAI-86, pp. 118–124, AAAI Press, Menlo Park, USA.
Struss, P., Price, C., 2003. Model-based systems in the automotive industry. AI Magazine 24 (4), 17–34.
Tullos, D.D., Neumann, M., 2006. A qualitative model for analyzing the effects of anthropogenic activities in the watershed on benthic macroinvertebrate communities. Ecological Modelling 196 (1–2), 209–220.
Veber, P., Borgne, M.Le, Siegel, A., Lagarrigue, S., Radulescu, O., 2006. Complex qualitative models in biology: a new approach. Complexus 2004–05 (2), 140–151.
Weld, D., de Kleer, J. (Eds.), 1990. Readings in Qualitative Reasoning about Physical Systems. Morgan Kaufmann Publishers, Palo Alto.
Williams, B.C., Ingham, M.D., Chung, S., Elliott, P., Hofbaur, M.W., Sullivan, G.T., 2003. Model-based programming of fault-aware systems. AI Magazine 24 (4), 61–76.
Winkels, R., Bredeweg, B., 1998. Qualitative models in interactive learning environments. Interactive Learning Environments (special issue) 5 (1–2), 1–134.