Common Sense Reasoning

Towards Mature Qualitative Reasoning Engines

Master Thesis Artificial Intelligence

Floris Eke Linnebank (9750207)

University of Amsterdam

Faculty of Science

Laboratory of Social Science Informatics
Roeterstraat 15, 1018 WB Amsterdam, The Netherlands

July 2004

Supervised by Dr. Bert Bredeweg

Abstract

Qualitative Reasoning captures human common sense understanding of the behavior of real world physical systems. Qualitative models use a rich explicit conceptual vocabulary and therefore a qualitative simulation provides an explanation as well as a description of the system's behavior. These properties make Qualitative Reasoning suitable for use in several application fields like education and ecology. The design of an elaborate, detailed Qualitative Reasoning engine is a complex task. Existing simulators lack reasoning capacities which are required by present day applications, such are extensive use of inequality reasoning and efficient computation of state transitions. This thesis describes an effort to develop fundamental solutions for several problems in such reasoning engines, particularly focussing on the GARP reasoning engine.

Table of Contents:

1. Introduction

Qualitative Reasoning is a field of AI that is concerned with knowledge-based computer models of the behavior of physical systems (Weld & De Kleer, 1990). Qualitative Reasoning offers a rich vocabulary for describing behavior of systems at a conceptual level which is akin to the kind of information humans use in tasks involving common sense reasoning about system's behavior (Forbus, 1988). Fundamental aspects and distinctions of a system and its behavior are modeled, while extra (numerical) detail is left out. Qualitative Reasoning approaches typically use explicit ontologies with descriptive concepts, incorporate the notion of causality, and infer behavior from the structure of a system. This way a simulation using a qualitative model provides an explicit explanation for the behavior of the system. Often a compositional modeling approach is taken (Falkenhainer & Forbus, 1991), using small reusable building blocks to form a complete model. Today several application fields exist for Qualitative Reasoning, including: the space industry (Williams et al., 2003), education (Bredeweg & Forbus, 2003), ecology (Salles & Bredeweg, 2003), and the automotive industry (Struss & Price, 2003).

The GARP reasoning engine (Bredeweg, 1992) is a domain independent qualitative simulator that follows the compositional modeling approach and has its roots in the component- and process centered qualitative reasoning approaches (De Kleer & Brown, 1984; Forbus, 1984). The core GARP engine uses a command line interface and originally a text-editor was used for model construction, which required some programming skills. A graphical interface: VisiGarp (Bouwer & Bredeweg, 2001) has been constructed for running simulations, as well as a graphical model building tool: HOMER (Bessa Machado & Bredeweg, 2003). This more accessible 'workbench' has enabled the user community to grow and the GARP environment is now used in several research programs (e.g. Salles & Bredeweg, 2003; Goddijn et al., 2003; QRSER, 2003; MONET, 2003; QR04, 2004). Consequently larger and more diverse models are being constructed. These growing demands have revealed several shortcomings in the current implementation of GARP, version 1.7.2. It does not always make maximal use of available knowledge to derive specific behavior and to focus the reasoning process. In some situations it does not generate the full spectrum of possible behaviors of the system. Also it cannot generate behavior in case of missing knowledge. Lastly it has no facility for generating the behavior of exogenous variables (Rickel & Porter, 1997). Goal of the project described in this thesis is to analyse these reasoning capabilities missing in GARP 1.7.2, to develop fundamental solutions to these problems, and to implement these solutions in an upgraded version: GARP 2.0.

The next section will introduce the GARP engine and the qualitative reasoning process in more detail. Section 3 provides an overview of the required new reasoning capabilities and the associated problems. Solutions implementing these reasoning capabilities are described in section 4. Several models were used to evaluate GARP 2.0. These models and results are presented in section 5. The last section contains a conclusion and discussion.

2. Qualitative Reasoning in GARP

2.1 Overview

GARP (Bredeweg, 1992) is a domain independent qualitative reasoning engine implemented in SWI-Prolog. A simulation of a system's behavior in GARP starts with a scenario, which contains a structural description of the system. A library of model fragments (Forbus, 1984) captures domain knowledge. The scenario is augmented with knowledge from applicable model fragments resulting in one or more possible states of behavior. A state represents a segment of time in which the qualitative description of the system remains steady. Each state is analysed with respect to changing aspects; these 'behaviors' form transitions that lead to new states of behavior. A complete simulation consists of a graph with states as nodes and transitions as vertices. Figure 2.1 illustrates the simulation process.

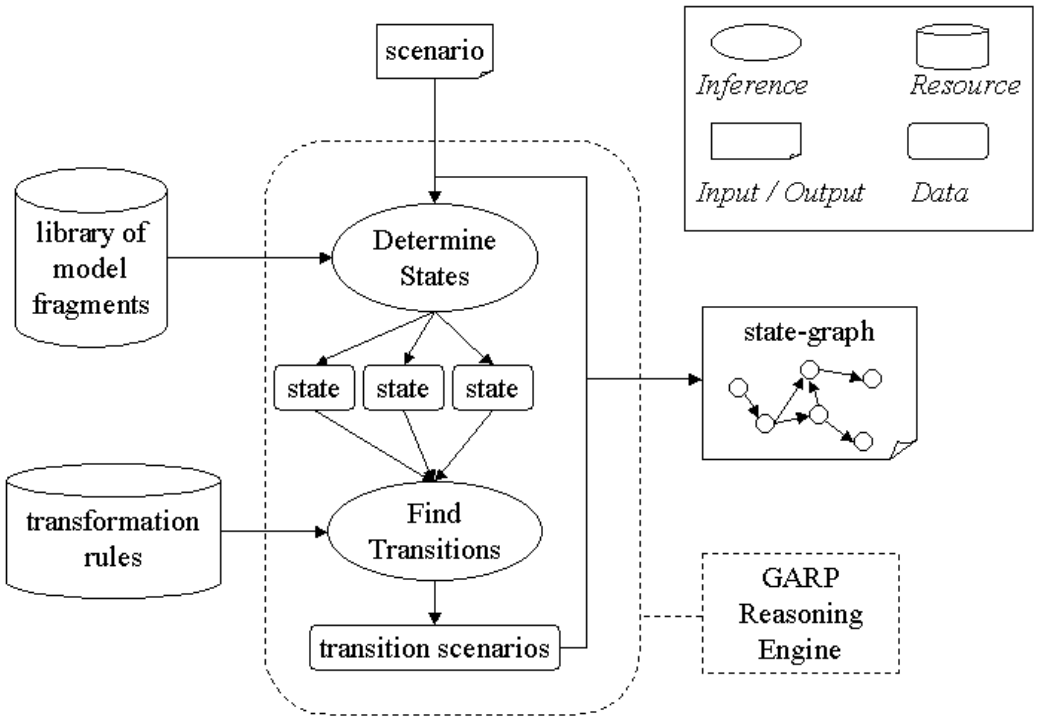


Figure 2.1: Main Inferences In GARP

Section 2.2 gives a description of the constructs used in GARP. These building blocks are referred to throughout this thesis. Section 2.3 describes the inferences from figure 2.1 in more detail. Section 2.4 then further describes the inequality reasoning capacities of GARP. These play an important role throughout the GARP reasoning engine.

2.2 Descriptive concepts

In GARP, scenarios, state descriptions, model fragments and transition rules all use the same descriptive concepts. A scenario or state description consists of a single set of these constructs. Both model fragments and transition rules use an *if-then* format and consist of a conditional and a consequential set of these constructs[1]. An introduction to each concept is given below:

- Entities:

Entities represent the physical, real world parts of a system. Every entity is an instance of a generic entity defined in an isa hierarchy. The structural configuration between entities can be declared and static properties of entities can be captured using attributes. Consider for example a leaking bucket being filled at a water tap. Figure 2.2 illustrates this situation. In a simple model of this situation there are two entities involved: a tap and a bucket. A configuration will say that the bucket is placed under the tap. An attribute of the bucket is that it has a few holes in the bottom.

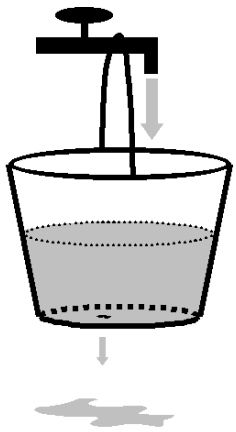


Figure 2.2: Filling A Leaking Bucket

- Assumptions[2]:

In modeling one may need to make assumptions. Examples of types of assumptions are: simplifying assumptions (e.g. zero friction) and operating assumptions (e.g. leakage is smaller than inflow from the tap) (Falkenhainer & Forbus, 1991). Assumptions are used in GARP as conditions for model fragments. Several model fragments can describe the same phenomenon but on different levels of abstraction or different levels of detail etc. These different model fragments will trigger on the presence of different assumptions. In this way assumptions facilitate a flexible library capable of modeling a phenomenon in varying ways.

---

[1] Besides the constructs listed in this section model fragments can also have the activity of other model fragments as a condition. Some transition rules also use other special purpose constructs. Section 2.3 provides more information on transition rules.

[2] In the core GARP engine *assumptions* have the same status as *entities*. They are treated equally, yet in HOMER no configurations or attributes can be associated with assumptions. A third class: a*gents,* is also distinguished in HOMER. Agents are not relevant to the work presented in this thesis and are therefore not discussed here.

- Quantities[3]:

Quantities are dynamic properties of entities, they represent changing aspects of a system. In our running example the amount of water in the bucket, the water level and the pressure at the bottom are quantities associated with the bucket. As water flows into the bucket the amount of water may increase, and the water level may rise as well as the bottom pressure. A discrete set of qualitative values is used to capture this continuous behavior.

- Quantity values:

For each quantity a quantity space defines the qualitative values it can take and their partial ordering. In GARP this is an ordered list of alternating points and intervals. From here on we will refer to points on a quantity space as landmarks[4]. A carefully chosen quantity space retains exactly those distinctions that are relevant. In different values of the quantity space different behavior should occur. The sign quantity space: {min, point(zero), plus}, typically satisfies this idea and is used very often. However it is also a very rigorous abstraction and more detail may be needed. In GARP all quantity spaces associated with the quantities are in principle unrelated. The exception being the landmark zero, this is the same for every quantity space using it. Other relations between quantity spaces must be stated explicitly in a model through inequalities between landmarks. The two quantity spaces for the temperatures of two substances pictured in figure 2.3 illustrate the concept of the quantity space. Note that melting points and boiling points can be unequal. The actual value of a quantity is defined in a value statement. This value must be in the quantity space specified for the quantity. Besides a current value a quantity also has a derivative indicating how the quantity is changing. Derivatives always use the sign quantity space: {min, point(zero), plus}. A *zero* derivative indicates a steady quantity: no change. A derivative that is *min* or *plus* indicates the quantity is moving up or down its quantity space respectively. In the bucket example the quantity *amount of water* may have the quantity space {point(zero), plus, point(maximum)}. When empty and first placed under the tap, it will have the value *zero* and it's derivative will be *plus*[5].

[3] In both VisiGarp and HOMER the term quantity is used, but originally the term parameter was used. In GARP this is still the case. The term quantity will be used in this thesis.
[4] See (Forbus, 1988) for an elaborate discussion of limit points and landmarks.
[5] Causal dependencies determine this derivative to be plus as explained below.

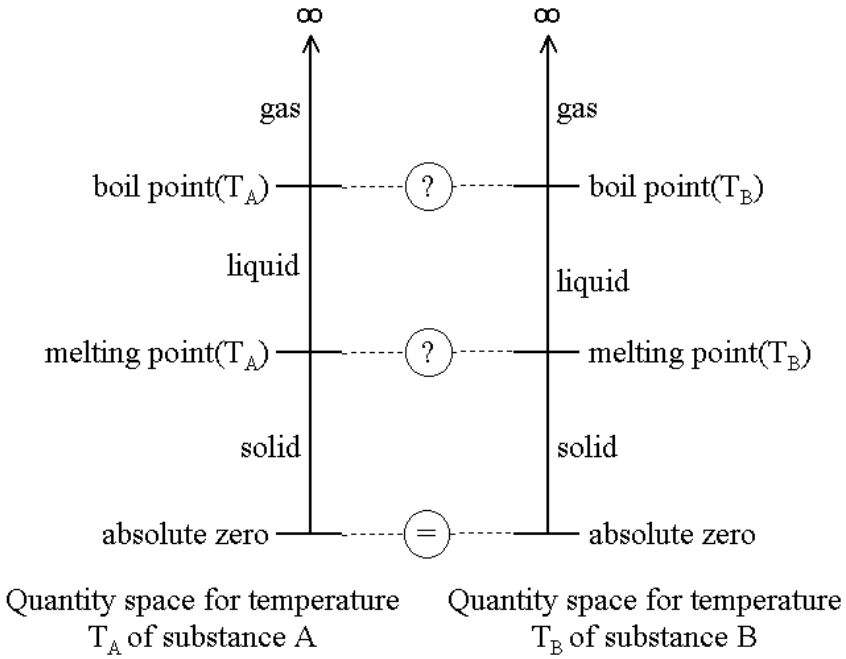


Figure 2.3: Quantity Spaces And Landmark Relations[6]

- Mathematical Dependencies:

Mathematical dependencies are inequalities {>, ≥, =, ≤, <} including additions and subtractions. Inequalities can be defined between quantities, between derivatives of quantities, and between landmarks. But also inequalities between quantities and landmarks and between derivatives and landmarks are allowed. Inequalities between quantities and derivatives cannot be defined. Note that inequalities provide a second way of defining a quantity's current value or derivative besides using a value statement. Given for example the situation in figure 2.3, the inequalities: T_A < *boiling point*(T_A) and T_A > *melting point*(T_A), together state that T_A has the value *liquid*. The inequality T_A > *melting point*(T_A) alone will only constrain the current value of T_A to *liquid*, *boil point*(T_A) and *gas*.

- Causal Dependencies:

These are Influences, Proportionalities and Correspondences, similar to the concepts used by (Forbus, 1984). Influences represent direct effects of a process; a causal chain always starts with an Influence. One quantity has an effect on the other, making it change. These changes are then propagated through Proportionalities. If a quantity is changing, then another quantity will also want to change if it is proportional to the first. More then one influence or proportionality may be affecting a quantity, but a quantity may not be both affected by an influence and proportionality at once (Forbus, 1984). Both Influences and Proportionalities can be a positive relation where a positive value has a positive effect and vice versa or a reversed relation where a positive value has a negative effect and vice versa. Figure 2.4 shows a causal model for the leaking bucket situation. The *water flow* from the tap has a positive effect on the *amount of water* in the bucket; the *water flow* from the bucket has a negative effect. The increase or decrease of the *amount of water* will propagate to the *water level* and

[6] The quantity spaces in this example commit to the Kelvin temperature scale where absolute zero is equal to 0 degrees K.

the *bottom pressure*. The *water flow* from the bucket is in turn proportional to the *bottom pressure* in the bucket. The last[7] type of dependency is the correspondence. If two values correspond for two quantities it means that when one quantity has that value, the other quantity will have the other value and vice versa. An example of a correspondence is that the *water level* in the leaking bucket must be *zero* when the *amount of water* is *zero*. GARP[8] recognizes four types of correspondences: a correspondence can be defined for two values, but also for all values in the quantity spaces of two quantities if they have the same number of points and intervals. Effectively the latter is like defining value correspondences for each value pair in the quantity spaces. Of both value and quantity space correspondences a directed form exists. This means its activity depends only on one of the quantities. If the other quantity is in the corresponding value this has no consequence.

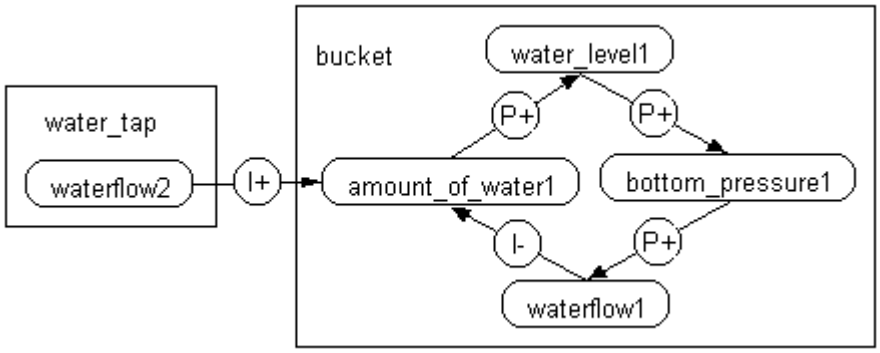


Figure 2.4: Dependencies Of The Leaking Bucket Situation

2.3 Inference Structure

In the top-level inference structure of GARP, as depicted in figure 2.1, two inferences can be seen: Determine States and Find Transitions. These inferences are similar to the concepts of intrastate analysis and interstate analysis as introduced by (De Kleer & Brown, 1984), although the approach is different. De Kleer & Brown use a breadth first approach; firstly all states of the system are generated. Intrastate analysis then determines which states are internally consistent and interstate analysis determines which valid states may change into other valid states. These two steps result in a total envisionment of all possible behaviors of a system. GARP adopts a depth first approach; firstly all valid states following from the scenario are computed. Then their changing aspects are analysed which leads to 'transition scenario's' that are in turn processed leading to new states. This two-step process continues until all states and transitions have been found that are attainable from the original scenario. The resulting state-graph is called an attainable envisionment.

[7] (Bredeweg, 1992) also defines the if and iff relation. These predicates have conjunctive sets of dependencies as arguments and have the normal semantics of the directed and undirected implication. However they are not incorporated in the HOMER and VisiGarp environments and are therefore in general not used.
[8] (Forbus, 1984) only introduced the undirected value correspondence.

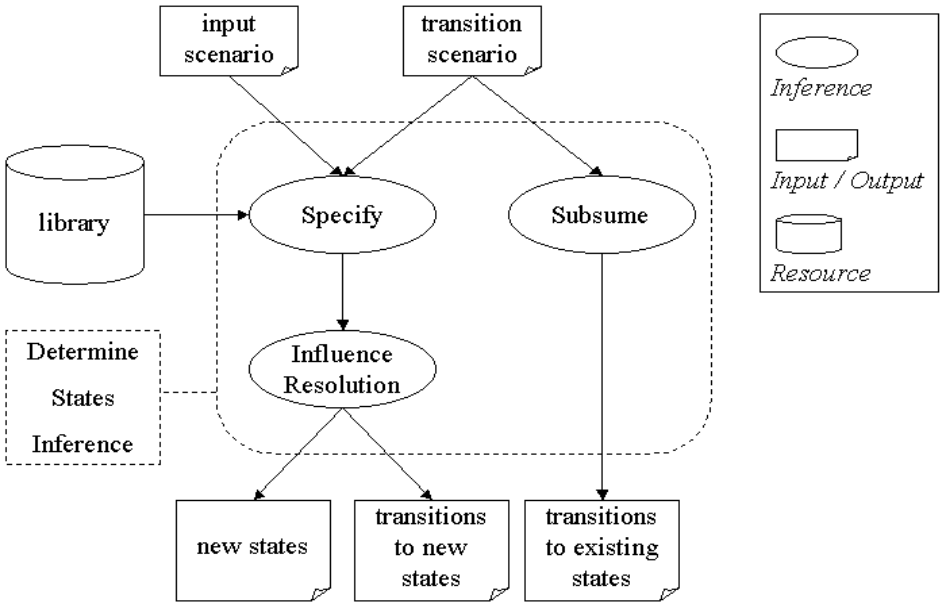


Figure 2.5: Determine States Inference

A more detailed view of the Determine States inference is shown in figure 2.5. It takes as input a scenario and a library of model fragments, and outputs transitions to new or existing states and the new states themselves. There are two possible routes in this inference. Firstly the Subsume step tries to match the scenario with existing states, if one ore more matching states is found, transitions to these states are created and the inference is finished. If no existing states match the scenario it must lead to one or more new states. These states are fully computed by the Specify procedure. It applies the model fragments, and is described in depth in (Bredeweg, 1992). An important aspect of the Specify procedure is that when a conditional quantity value or inequality is not known it tries to assume this condition. After all model fragments are found the causal model is complete and its effects can be computed by the Influence Resolution procedure. Once a state description is complete it is saved and transitions from the predecessor state to the new state are made. Note that both Specify and Influence Resolution are procedures with possibilities for branching, in other words: generating more then one solution. There can be mutually exclusive model fragments and ambiguous causal results. After all these branches are generated the inference ends.

The second inference step from figure 2.1, Find Transitions, analyses which aspects of a state are changing and outputs transition scenario's for all possible courses of behavior. As can be seen in figure 2.6, this is a three-step process. In the first step termination rules are applied, these detect single aspects of behavior that may cause a state of behavior to terminate and turn into a different one. Forbus (1984) calls this process Limit Analysis.

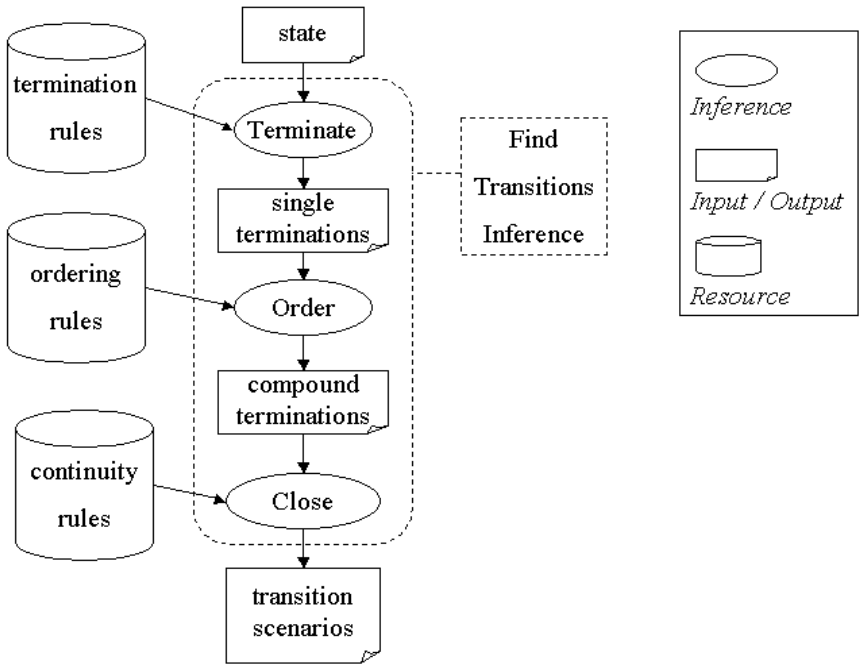


Figure 2.6: Find Transitions Inference.

An example is that when our bucket is being filled it may become full. Technically speaking: *Amount of water* is *plus* and increasing (its derivative is *plus*), thus *amount of water* may reach the next point on its quantity space: *maximum*. Such a change ends the current state of behavior and is called a 'termination'. In the next step ordering rules are applied to determine which terminations will actually occur or take precedence over others and in what combinations these terminations will take place. For example the epsilon ordering rule (De Kleer & Brown, 1984) states that terminations from a point to an interval happen immediately whereas terminations from an interval to a point always take some finite amount of time. Therefore the first type takes precedence over the second type. An example of a necessary combination of terminations in the bucket system is the case where *amount of water* and *water level* both have a value termination to *zero*. Because there is a value correspondence[9] between these quantity values the terminations are only valid together. The Close procedure completes these compound terminations into 'transition scenarios' by applying continuity rules. These rules ensure that values and derivatives will not show discontinuous behavior. For derivatives there are no indications whether they will change or not, therefore these are given one step of freedom, allowing them to change in the next state. For example, a derivative which is *plus* may be *zero* or *plus* in the next state. Quantity values that are not changing through a termination will have to keep their current value in the next state. A complete transition scenario consists of the scenario of the original state augmented with all the results of the compound termination and all results of the continuity rules.

[9] See section 2.2 on descriptive concepts.

2.4 Inequality Reasoning: The Mathematical Model

The Mathematical Model and the Inequality Reasoning procedure in GARP have a specialized form to suit the needs of a qualitative reasoning engine (Reinders, 1989). They play an important role in GARP and therefore also in this thesis. There is a distinction between the mathematical model at user level and the mathematical model that is used in computations. In this section a description of this internal representation is given as well as a description of the inference capacities of GARP on sets of inequalities.

The mathematical model at user level consists of all value statements and inequalities present in the scenario and applied model fragments. If the separation between user level- and internal mathematical model would not be made the user would be confronted also with all relations derivable from the former set. These derived relations are needed in reasoning, but they do not help to make the simulation insightful. Furthermore a representation that is computationally efficient is generally not a user-friendly representation. The internal mathematical model of GARP is built up entirely of inequalities. As described in section 2.2 each quantity value can be expressed using one or two inequalities. The same is true for derivatives. The partial ordering between landmarks implicit in each quantity space definition (a list of points and intervals) can of course also be expressed using inequalities. These inequalities are of the following form:

relation([pointer-set left], rel, [pointer-set right]), with: $rel \in \{\geq, >, =\}$

Quantities[10] in the inequality are represented by a unique integer working as a pointer. An index keeps track of the quantity/integer mapping. Only the '>', '≥' and '=', relations are used, the '≤'and '<' relations are inverted. The sets of pointers represent positive sums of quantities. Summations may contain the same quantity twice, which is not expressible using a set; substituting one of the double quantities with a new but equivalent pointer solves this. Subtractions are expressible as additions by transferring quantities from one side to the other. Zero maps to the empty list [ ]. For example, the relation:

$A + B + B < C - D,$

translates to the internal relation:

relation([3], >, [1, 2, 4 , 5]), with: *A / 1, B / 2, C / 3, D / 4, B / 5*

The inference rules used in GARP are similar to those in the Quantity Lattice proposed by (Simmons, 1986). Table 2.1 shows all possible inferences, these can be computed using set operations.

	A = B	A ≥ B	A > B
C = D	A + C = B + D	A + C ≥ B + D	A + C > B + D
C ≥ D	A + C ≥ B + D	A + C ≥ B + D	A + C > B + D
C > B	A + C > B + D	A + C > B + D	A + C > B + D

Table 2.1: Transitivity Rules

An example may illustrate the inference process. Take two relations:

[10] In this section on inequality reasoning the word *quantity* refers to *current values*, *derivatives* and *landmarks* in general. When performing internal inequality computations the distinction is irrelevant.

$X > Y$ and $Y \geq Z$

Using table 2.1 and the set union operation[11] for both left and right sides the following intermediate result is obtained:

$X + Y > Y + Z$

Algebraic simplification by subtracting equal elements from left and right sides yields:

$X > Z$

In the inequality reasoning procedure, contradiction is concluded when a relation of the form $X > X$ is derived. The full closure under these inference rules is computed every time a relation is added to the input set of inequalities. Having the full closure always at hand means that checking if a relation is derivable in the mathematical model only involves a simple presence check. Computing the full closure is a large operation as the mathematical model grows and constraints are placed on the inferences to avoid combinatorial explosion. Examples are the following:

- Derived relations involving sums are only kept for intermediate use in deriving simple relations not involving sums.
- No transitivity inferences are made with zero as intermediate variable. Many relations concerning values, derivatives and quantity spaces include zero and these results are not very informative. For the derivation of relevant results this is not a problem, in general relations can be derived in many ways.
- Parents of a derived relation may not be combined with the derived relation again.

Correspondences[12] also play an important role in the construction a coherent mathematical model for they supply information on valid value combinations. Correspondences are translated to directed and undirected implications on conjunctive sets of inequalities[13]. This is quite a natural translation, as said before, there are directed and undirected correspondences and values can be expressed using one or two inequalities. Every time an inequality is added to the model and the new full closure is computed, this set of implications is checked. If an implication is found to hold the resulting set of inequalities is added to the mathematical model in the normal way. This mechanism that checks these implications also accommodates a subtle kind of information present in quantity spaces where an extreme value is a landmark. The idea is that in the highest point a value cannot increase anymore and in the lowest point it cannot decrease anymore. In our bucket example this means that when *amount of water* has the value *zero*, its derivative must be greater or equal to *zero*.

This concludes the description of qualitative reasoning in GARP 1.7.2. The next section introduces the reasoning capabilities missing in this reasoning engine.

[11] As said before GARP is built using SWI-Prolog. In prolog, set operations on lists are quite slow. For faster set manipulation the pointer lists are transformed into bitmaps. Calculation takes place at machine level with routines using the C programming language.
[12] See section 2.2 on descriptive concepts.
[13] Note that this form also accomodates the *if* and *iff* relations mentioned in section 2.2.

3. Required Reasoning Capabilities

A Qualitative Reasoning environment should provide the user with all modeling primitives needed to capture the behavior of a system and it should produce a correct simulation given a system model by making the strongest inferences possible. This section describes areas in GARP 1.7.2 that do not meet these requirements.

3.1 Influence Resolution Calculus

The method used in GARP 1.7.2 for determining the effect of multiple influences or proportionalities on a quantity is based on sign calculus. This means that in cases where two opposing influences are active the result is ambiguous (NB. *plus* + *min* = *?*). Consider for example again filling a leaking bucket with water as pictured in figure 2.2. The water flow from the tap has a positive affect on the amount of water and the water flow from the bucket has a negative effect, therefore it cannot be said if the water level will be decreasing, steady or increasing. GARP generates a separate state for every one of these options in such a case. Human reasoners often resolve such ambiguity by comparing the magnitudes of the two flows. This is done with the working assumption that the impact of an influencing flow is equal to its magnitude. If it is known, for example, that the leakage is smaller than the inflow from the water tap, then it is obvious that the water level will rise in our bucket. Such inferences cannot be made by GARP 1.7.2. Therefore a new method for determining the effects of influences is needed that makes use of such extra information about the magnitudes of influences. This will prevent spurious behavior and unnecessary branching of the simulation.

3.2 Transition Inferences

Reasoning about how states change into other states of behavior is important in behavior prediction. The procedure used in GARP 1.7.2, is not completely adequate. Several problems and questions are still open and need to be solved. As described in section 2.3 a rule interpreter is used, but the rule format and semantics are problematic:

- Rules do not have access to the inequality reasoning procedure and the internal mathematical model[14]. Instead the user level representation is used, which is less powerful and complete. As a result, the strongest possible inferences are not always made.
- The rule format does not distinguish between purely conditional statements and conditional statements that need to be replaced with the results of the implication. This is awkward in writing down rules, because every non-changing statement in the conditions has to be repeated in the results[15].
- The rules have the form of an implication, but negation is not expressible. As a result it is for example impossible to say that a certain change is only possible if a specific other change is not in the set of possible changes.
- Ordering rules used to determine valid combinations of changes have only two possible results, either two changes are merged into a compound change or a change is

[14] See section 2.4 on inequality reasoning.

[15] In the past this has even led to problems where instances of entities in the isa hierarchy were replaced by isa-subtypes in the next state. This problem was solved already in GARP 1.7.2.

completely removed from the set of possible changes. More subtle ordering concepts cannot be expressed. For example: change A can take place only together with change B, but change B can also happen alone.

- Ordering rules can only process two changes at a time, making rule construction involving three or more changes hard. If for example a certain ordering concept suggests that 3 changes should be merged into one, then two rules are needed to do this.
- As noted in section 2.2 both a value statement and an inequality can define a quantity's value. Sometimes only one is present, sometimes both. Therefore separate rules are needed to find the changes for both statements. Because they concern the same event, the quantity taking on a different value, these twin-changes need to be located and merged into one whenever present together. Since a merged transition name is more complex then a single one this makes transitions less insightful for the user.

The set of transition rules has been updated many times during the existence of GARP. During this process many concepts involved in transitions have become clear, yet a comprehensive effort to "get things right" has not been made. It is therefore likely that the set of concepts is quite mature, but still incomplete. For example, in a large model like the Cerrado Succession Hypothesis (Salles & Bredeweg, 1997; Salles & Bredeweg, 2003), domain specific ordering rules are needed to reduce the number of possible combinations of changes. In this model, the quantities, *Number_of_individuals, Born, Dead* and *Emigrated* correspond in the value zero. The domain specific rules remove changes to zero for *Born, Dead* and *Emigrated* if *Number_of_individuals* is still at least 2 values above zero, because obviously it cannot reach zero from such a point. Without these extra ordering rules combinatorial explosion takes place. It seems there are domain-independent ordering concepts present in these domain-specific rules. It is important to study the concepts involved in state transitions and make an effort to come to a complete implementation of these concepts. Given that not many users actually design rules it seems worthwhile to give up the flexibility of the rule interpreter for the speed and power of dedicated procedures for each transition step.

The last problem lies in the application of changes. In many cases such a 'transition-scenario' theoretically leads to multiple successor states because of ambiguity and continuity[16]. It is possible that these states have already been generated in the simulation and the transition leads to these existing states. But if for some reason only a subset of the possible successor states is already present, then the subsumption procedure described in section 2.3 will find the transitions to these states after which further search for successor states is abandoned. The result is an incomplete simulation. A procedure is required that always finds *all* possible successor states.

3.3 Exogenous Quantities

The *sole mechanism assumption* (Forbus, 1984: p.109) states that 'all changes in a physical system are caused directly or indirectly by processes'. In some modeled systems however, not all processes affecting the system are within the modeling scope. The sun for example has an effect on the earth climate, but many models of the earth climate will not model the processes that cause the sun's varying radiation. In such a model the solar radiation is right at the system boundary, it affects the system but is not affected by any quantity within the system. Such a variable is called an exogenous variable (Rickel & Porter, 1997). In GARP 1.7.2 there is no

[16] See section 3.1 on influence resolution calculus and section 4.3.1 on continuity over transitions.

facility for generating the behavior of 'exogenous quantities'. In existing models this is done using domain specific transition rules and model fragments. A generic way of dealing with exogenous quantities is required. Different regular and irregular, simple and complex behavior patterns have to be defined for exogenous quantities. Of course a mechanism to implement these behavior patterns in the transition procedure is needed as well.

3.4 Correspondence Primitives

As explained in section 2.2 quantities can correspond in some or all values of their quantity space. Correspondences can be directed or undirected. When two values correspond it means they always occur simultaneously, it does not mean these values are equal. Take for example the length of an elastic band as a function of the exerted force. Although we may not know the exact form of this function, when the elastic band is at rest length the exerted force is zero and vice versa. This can be modeled by an undirected value correspondence. An example of a directed correspondence is that between the flow through a valve and the valve opening. When the opening is zero (the valve is closed), there is zero flow. But when there is zero flow this does not always mean the valve is closed. When a correspondence mapping exists for every point on both quantity spaces, a quantity correspondence can be used. Note that this notion is also expressible using multiple value correspondences. In practice this method is used to define inverted quantity correspondences: a mapping of the highest point of one quantity space to the lowest point of the other and so on. Incorporating this correspondence type in GARP will ease the modeling effort and it is interesting to see if other types are useful to the user, or if new types are needed because they are inexpressible given the current correspondence types.

3.5 Modeling Assumptions Theory

Every real world phenomenon can be modeled in varying ways. Different aspects of behavior can be considered at varying levels of detail. In essence every model is an abstraction of the complex real world phenomenon. It is valuable to explicitly state which abstractions, simplifications, assumptions etc are used in a certain model. Not only to be clear about the explanatory scope of a model, but also to know which model fragments can be used to construct the model. The generic knowledge present in each model fragment is always intended for use with a certain view of a situation.

In GARP 1.7.2 only a simple mechanism is available to implement assumptions. Assumptions defined in a model are unrelated to other assumptions or other elements in the system and are valid for the system as a whole. They serve as conditions for model fragments. However, more distinctions between types of assumptions can be made and relations between assumptions constraining valid combinations can be identified. Simplifying and operating assumptions can be distinguished (Forbus & Falkenhainer, 1991). Ontology-, grain size- and approximation assumptions are examples of the first type, Operating mode-, steady state- and behavior restriction assumptions are examples of the second type. Different time perspectives can be distinguished (Rickel & Porter, 1997). These lead to use of the quasi- static approximation in case a phenomenon occurs on a time scale significantly smaller than the time scale of interest. And if a phenomenon occurs on a time scale significantly larger than the time scale of interest it can be left out of consideration. Assumptions are also often interdependent and a formalism is needed to capture these relations. Assumption classes (Forbus & Falkenhainer, 1991) can be used for this purpose. Constraints define activity of an assumption class and once active, one of the mutual exclusive assumptions in the class must

be used. Besides relevance and mutual exclusion, other important issues include: default assumptions, assumptions requiring certain other assumptions, and associating different assumptions with different parts of the modeled system. Lastly, given these constraints, the question of generating a coherent set of assumptions and model fragments capable of modeling the system in question is not trivial. It can be seen as a dynamic constraint satisfaction problem.

3.6 Order of Magnitude Reasoning

Order of Magnitude Reasoning is a very natural form of Qualitative Reasoning. Several frameworks have been proposed to implement this kind of reasoning. (Raiman, 1991; Mavrovouniotis & Stephanopoulos, 1988; Travé-Massuyès, 2003). Since rough order of magnitude information is more often available then precise inequality information this capacity is very useful in a qualitative reasoning engine. However, the integration of such a formalism in a complex system such as GARP is not trivial A suitable formalism has to be found and the relation to other inference capacities such as influence resolution and model fragment application has to be determined.

The next section of this thesis describes the developed solutions to the problems outlined in the previous sections. The subjects of Modelling Assumptions and Order of Magnitude reasoning however, have not been treated and are therefore not further discussed.

4. Implementing New Reasoning Capabilities

4.1 Flexibility

In GARP 1.7.2 the rule interpreter provides flexibility when reasoning about transitions. Users can add and remove rules to influence how the reasoning engine behaves. For example, rules implementing epsilon ordering[17] can be removed when all values in the quantity spaces are interpreted as intervals. The Cerrado Succession Hypothesis (Salles & Bredeweg, 1997; Salles & Bredeweg, 2003) is an example of such a model that does not use the normal alternating point-interval interpretation for quantity spaces. Adapting the rule set is not easy though and requires some programming skills. GARP 2.0 features algorithm option switches for compatibility with different modeling styles. These switches determine reasoning engine behavior by (de)activating certain functionalities and by choosing between alternative algorithms with different properties. Examples of these options are the application of epsilon ordering, assuming behavior in undetermined situations, and making a closed world assumption during influence resolution[18]. All algorithm option switches are described in their relevant context in this thesis. Default settings favorable for most models have been carefully chosen. Settings can be manipulated through an interface menu. As models can be built depending on specific settings, the possibility of associating specific settings with a model is included.

4.2 Influence Resolution Calculus

A desired property of a qualitative reasoning theory is that it allows graceful extension (Forbus, 1984), which means that more precise data should at least yield equivalent results and that, if possible, ambiguities following from weak data should be resolved by the stronger data. This means that in case of imprecise model behavior we can distinguish two cases. Either there is a lack of information or there is a lack of inference capacity. In GARP 1.7.2 the effects of the causal dependencies (Influences and Proportionalities) are determined using sign calculus (De Kleer & Brown, 1984; Travé-Massuyès et al., 2003). Opposing effects will therefore always result in ambiguous behavior. Consider again the leaking bucket situation. As can be seen in figure 4.1 the *amount of water* in the bucket is influenced both by the *water flow* from the tap and by the *water flow* from the bucket (the leakage).

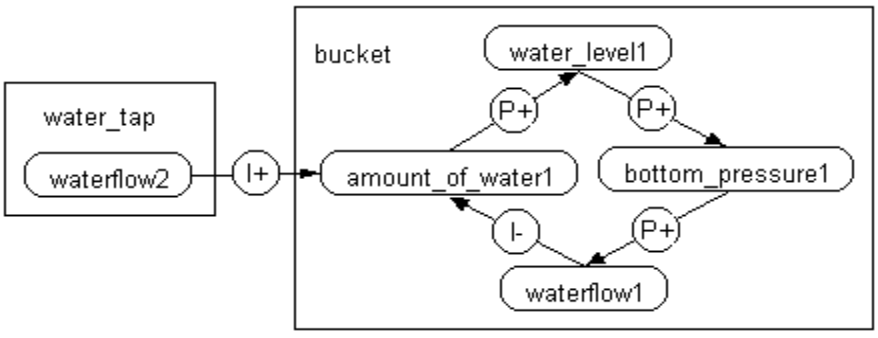


Figure 4.1: Dependencies In The Leaking Bucket Situation

[17] See section 4.5.3 on epsilon ordering.
[18] See section 4.4 on determining change and section 4.2.2 on the closed world assumption.

This situation is ambiguous, the *amount of water* can be rising, steady or decreasing because of the opposing effects of the two influences. However, in reality this situation is sometimes not ambiguous. Turn the faucet wide open on a bucket with a small hole and it will be full in no time. The extra knowledge in this case is that the water flow from the tap is greater then the water flow from the bucket. If this inequality is added to this model GARP 1.7.2 will still infer ambiguity. This lack of inference capacity needs to be dealt with.

4.2.1 Balancing Influences & Proportionalities:

To solve the problem of resolving opposing influences, positive and negative forces are summed and inequality information is used to determine which composite force is bigger. It is assumed in GARP 2.0 that the effect of each influence is equal to its magnitude. This assumption holds when the functions represented by the influences are of a similar nature. In most models it is the case that multiple influences are instances of the same phenomenon or generic process. For example the influences on the amount of water in the bucket are both water flows. (Forbus, 1984: p.112) similarly defines the derivative of a directly influenced quantity to be equal to the sum of its influences. The summation of influences can be seen as placing the influences on a balance scale. In GARP 2.0 an equation is constructed implementing this balance scale. Given multiple influencing quantities; *I+* and *I–*, on quantity *X*, the equation looks like:

$$BalanceResult = \sum value(I+) \; - \; \sum value(I-)$$

This equation is evaluated in the mathematical model[19] using the inequality reasoning procedure. The result is recorded and the derivative of *X* is given this value. It is important to use an intermediate result instead of simply asserting that the derivative of the influenced quantity is equal to the right hand side of the balance equation. Adding such a relation to the mathematical model would change the semantics of the directed asymmetrical influence operator to that of a non-directed symmetrical constraint[20]. Besides using an intermediate result the balance equation is not even added permanently to the mathematical model. Hereby the number of relations being evaluated throughout the reasoning process is minimized. A similar solution is used to solve the problem of resolving opposing proportionalities. Again a balance equation is constructed which is used in the same way as the equation used in influence resolution. Given multiple quantities; *P+* and *P–*, to which quantity *X* is proportional, the equation looks like:

$$BalanceResult = \sum derivative(P+) \; - \; \sum derivative(P-)$$

This means that it is also assumed in GARP 2.0 that the effect of proportionalities is equal to the magnitude of their derivatives. This assumption is not as easily justified as the one made earlier. In a model of population ecology for example, the rate at which individuals are born may be proportional to the size of the population but also to the resources available. The population size seems to have a linear sort of effect: more individuals means more mating and consequently a higher birth level. But resources seem to have more of a limiting effect: If they are abundant a change will not matter much, if they are at a normal level a change will have a moderate effect and if resources are very low a change will have a more dramatic effect. Likewise, Forbus (1984: p.112) states that opposing proportionalities cannot be resolved for

[19] See section 2.4 on inequality reasoning

[20] Construction of causal accounts from symmetrical constraints has been a subject of discussion (De Kleer & Brown, 1984; Iwasaki & Simon, 1986).

we lack detailed information about the form of the function described by these proportionalities. Still, GARP 2.0 does make this working assumption, if not only to provide the user with the tools needed when assuming a dominant effect. This is useful in constraining large simulations.

The following issue is not solved in GARP 2.0, but an approach is outlined that can be implemented in the future: If a situation is still ambiguous because there is insufficient inequality information between influences on a quantity *X*, then simulation branches for all three[21] alternative values of δX[22] are generated. In ambiguous cases one can argue that each different state represents a situation with a different ratio between positive and negative effects. An inequality can be constructed that captures this relation:

3) $\sum Positive \ \ rel \ \sum Negative$

with: rel *is* > *when* δX *is plus,*
rel *is* = *when* δX *is zero,*
rel *is* < *when* δX *is min.*

Such a relation can be added to the internal mathematical model[23] of that particular state or it can be added to the user level mathematical model. In the first case it provides an extra constraint on that state, but is invisible to the user. This is undesirable because its effects cannot be explained by the user. In the second case the constraint will be visible to the user and will even remain active after a transition to a next state unless it explicitly changes. Although this regime is theoretically correct and desirable GARP 2.0 does not construct and add such a relation because it can involve three or more quantities and there is no facility in GARP 2.0 for changing such inequalities[24]. The inequality would remain in effect throughout the simulation branch and this is clearly incorrect and undesirable.

A minor issue is that to make use of the above-described inference capabilities the proportion between the magnitudes of the influencing quantities is required. In case of quantity spaces with values both in the negative and the positive range, a simple inequality is not always enough. Consider for example the situation shown in figure 4.2. In this case the statement: $A > B$, is trivially known in GARP because positive values are always higher in the '>' partial ordering than negative values. To resolve ambiguity in this case, information about the magnitudes or absolute value of *A* and *B* is required. No primitive is available for expressing absolute values: $|A| > |B|$. The required information is expressible however with a little more effort: $A - B > 0$.

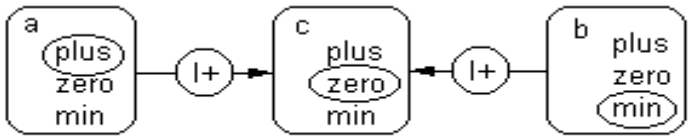


Figure 4.2: Two Influencing Quantities

[21] In case the balance result is ambiguous, but a weak constraint (*BalanceResult* ≥ or ≤ *zero*) is obtained the two corresponding states are generated.
[22] The symbol 'δ' prefixed before a quantity indicates the derivative of the quantity is considered.
[23] See section 2.4 on inequality reasoning.
[24] See section 4.5.3 constants and section 4.6.2 on inequality changes.

4.2.2 The Closed World Assumption:

Resolution of influences and Proportionalities is impossible if one of the driving variables[25] (Rickel & Porter, 1997) is unknown. This situation can arise because of two reasons:

1) The quantity with unknown derivative is in the middle of a causal chain and the preceding influences and/or proportionalities have not been resolved.
2) The quantity is at the start of the causal chain but the model is under specified and its value or derivative is simply unknown.

In situation 1 the preceding influences and/or proportionalities have to be resolved first. In situation 2 a closed world assumption can be made: "All that is known in the model is all there is in the world." The meaning in this context is that an unknown effect equals no effect and the unknown driving variable is set to zero. The model presented in section 5.2 will illustrate the need for this assumption in the process-centered approach to qualitative reasoning (Forbus, 1984). This assumption is reasonable at this point in the specification of a state because all model fragments have been applied and apart from influence resolution the state description is complete, so no new information is likely to become available. However, in some cases it turns out to be problematic: A partial causal model is shown in figure 4.3 that serves as an example. In this situation quantity *Y* is the begin- and endpoint of two causal chains. There is a proportionality from quantity *X* and an influence to quantity *Z*. Note that the driving variable of the influence, the value of *Y*, is unknown. If this value is assumed to be zero a problem occurs: Its quantity space requires the derivative to be greater then zero because a quantity cannot be decreasing in the lowest point of its quantity space, but this is in contradiction to the result of resolving the proportionality. Quantity *X* is increasing and through a negative proportionality *Y* should be decreasing.

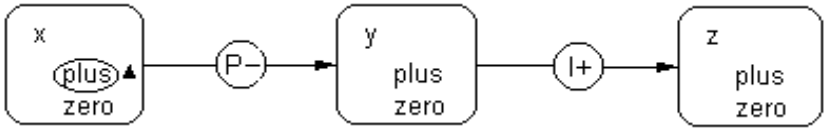


Figure 4.3: Partial Causal Model

In this example quantity space constraints interfered with the closed world assumption. Both correspondences and equalities can also supply new conflicting information when influence resolution is being performed.

GARP 1.7.2 deals with this problem by only making closed world assumptions one by one as the resolution process makes no more progress. These assumptions are retracted upon encountering contradiction. In such a case driving variables remain unknown and the resolution result is: 'undetermined'.

The following counter-example shows that this solution is not favorable. Firstly, a technical argument explaining a crucial property of the algorithm: The Determine States inference (figure 2.5) implements a depth first search for new states. After a closed world assumption

[25]For the influence pictured in figure 4.3 the value of Y is the driving variable, for the proportionality the derivative of X is the driving variable. Note that loops of proportionalities are not allowed (Forbus, 1984) so causal chains always have a starting point. GARP has no explicit mechanism detecting loops, but will not produce resolution results in case of a loop.

has been made and it has proven not to be contradictive it has to be fixed[26] in place. This is to prevent the search from generating almost the same state again (without the assumption, as if it had proven contradictive). However, a closed world assumption has only proven it does not lead to contradiction if a complete valid state description is found. And furthermore, fixing the assumption in place cuts-of all other alternative branches in the search for states between the moment of the assumption and the moment of fixation. Now consider the situation where first a closed world assumption is made and later on an ambiguous resolution result is obtained: In this case only the first ambiguous alternative is found by the search algorithm! After this first alternative succeeds the assumption is fixed and the other alternatives cannot be found. In figure 4.4 a causal model is shown that produces such a situation. In this model δA cannot be determined without the δC, which in turn depends on the value of quantity D. If this value is assumed to be zero, then δC is found to be plus and the resulting effect on δA is ambiguous.

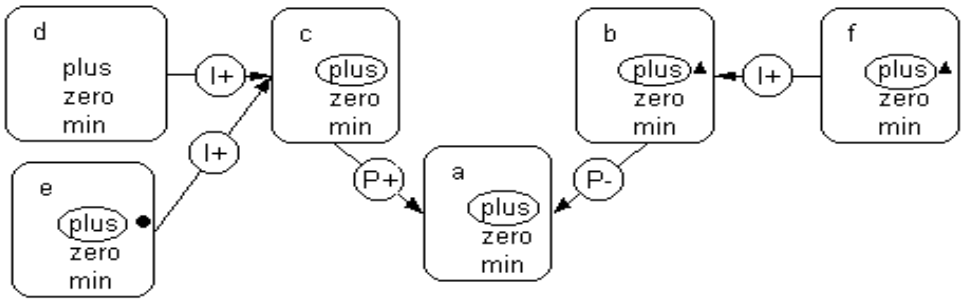


Figure 4.4: A Hypothetical Causal Model

To deal with the problems outlined above GARP 2.0 features a simple version of the closed world assumption. Before starting the influence resolution procedure it can assume all unknown driving variables at the start of a causal chain to be zero. This has the drawback of possibly generating conflicting situations but not the drawback of sometimes generating only partial ambiguity. To solve the first problem, application of the closed world assumption is optional[27] and the default setting is off. This setting is chosen because it is easy for users to understand that influence resolution does not work because of an unknown variable. And in general it is not hard to adjust a model so that all its initial driving variables are known. It is hard work however, to find out what contradiction caused a particular expected state not to be generated.

4.3 Transitions: General

The Find Transitions inference (figure 2.6) consists of three steps: Terminate, Order and Close, that find single changes, determine valid compound changes and apply these 'transition scenario's' respectively. The implementation of these steps in GARP 2.0 is described in sections 4.4, 4.5 and 4.6. Section 4.3 deals with general issues that are important throughout the transition procedure.

4.3.1 Constants:

In the transition procedure an important distinction between two types of inequalities needs to be made. Some inequalities between quantities describe a particular state of the system and

[26] For prolog users: The depth first search is implemented through a findall call and a succesfull closed world assumption is followed by a '!' (cut).
[27] See section 4.1 on flexibility.

should be allowed to change as their derivatives indicate this. Other inequalities between quantities have the function of a modeling constraint, or boundary condition for the simulation. Take for example the classic U-Tube system (Forbus, 1984) of which models are discussed in sections 5.2 and 5.3. The inequality saying the *water level* is higher in the left than in the right tube changes during the course of the simulation into equality in the final equilibrium state. However, the *flow* in the connection pipe is defined to be equal to the *pressure* in the left tube minus the *pressure* in the right tube. This inequality represents a modeling constraint and does not change during the course of the simulation.

Inequalities of the last category that remain valid throughout a simulation can be called constants. They can be used as constraints in the ordering procedure providing a valuable heuristic in determining which combinations of changes will produce a consistent state. For example take the following situation in the pipe of the U-Tube: *flow* is *plus* and decreasing, *pressure-L* > *pressure-R* and *flow* = *pressure-L* – *pressure-R* is constant. Then *flow* can only become zero if *pressure-L* becomes equal to *pressure-R* In GARP 2.0 constant inequalities can be explicitly defined using a keyword mechanism that is described in section 4.8. Furthermore GARP 2.0 assumes inequalities between three or more quantities (involving addition and/or subtraction) to be constant as well. No changes are generated for constants and they are used in the ordering procedure as constraints.

4.3.2 Landmark relations:

A simulation can sometimes incorporate multiple possible worlds. Relations between landmarks can be said to describe the specific possible world subject of the simulation. For example, in a simple U-tube system three possible worlds are possible, one where the right tube is higher, one where the left tube is higher and one where the tubes are of equal height. This is shown in figure 4.5. A model of such a system incorporates all three possible worlds if the landmark relation between the maximum heights of the tubes is not specified. This means that the simulation models different possible worlds at once. It is possible that different branches of the simulation model different possible worlds. Since possible worlds cannot change, the simulation should not be allowed to move from one such a branch to another one. But, as noted, the relations specifying the possible world are often unknown. However, landmark relations can be derived from time to time from the mathematical model. Consider again the three u-tubes shown in figure 4.5. In the situation on the left it is derivable that the right tube is higher than the left tube because the water in the left tube is at the top of the tube, the water levels are equal and the water in the right tube is still under the top. Often the relation between both maximums is not derivable as illustrated by the situations in the middle and right of the picture.

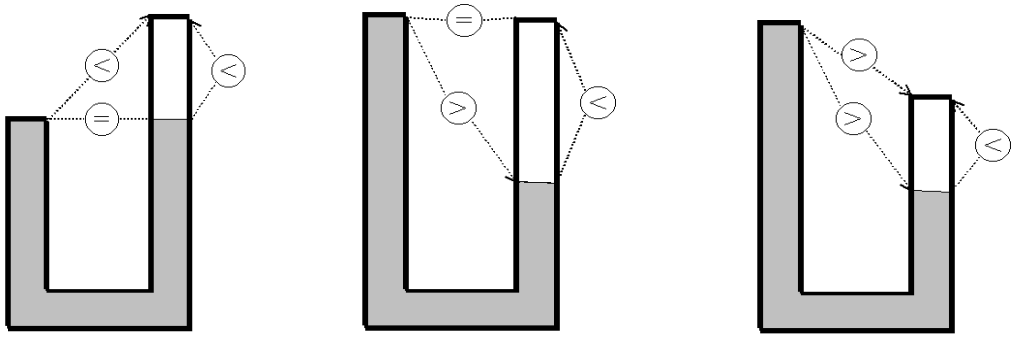


Figure 4.5: Three Different U-Tubes With Different Levels.

In GARP 2.0 landmark relations that are derived after specification of a state are extracted from the internal mathematical model and are passed on in possible transitions to prevent interacting possible worlds. Section 4.6 further elaborates this subject. A drawback of this approach is that the amount of newly derivable landmark relations can become quite large. To the user these newly derived relations can be confusing. Therefore an algorithm option switch is provided to disable deriving new landmark relations.

Landmark relations also provide valuable information in the ordering procedure. This is another reason to derive and store landmark relations. An illustrative example is the heating of two liquids with unequal boil points as is done in distillation. A schematic view of such a situation is shown in figure 4.6.

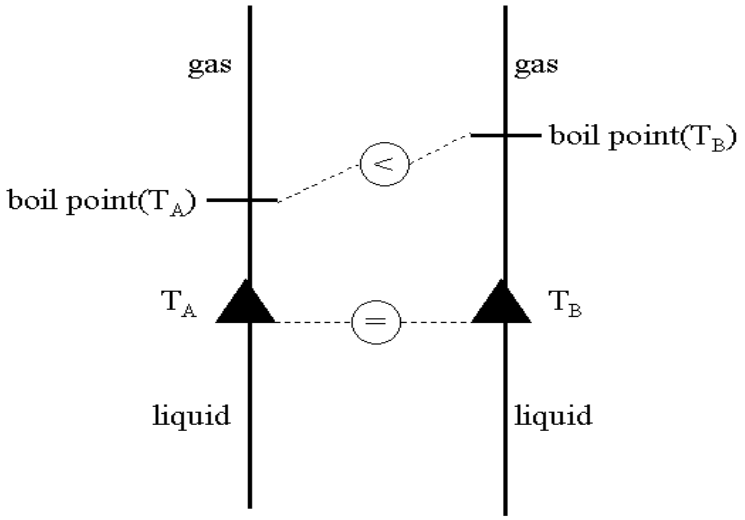


Figure 4.6: Two Liquid Substances Being Heated.

Because the substances are mixed their temperatures are equal. The landmark relation between the boil points of both substances indicates that liquid A must reach it's boil point before liquid B. And as long as their temperature remains equal and substance A has not turned completely into gas, liquid B will not reach its boil point. The use of this principle in the ordering procedure is discussed in detail in section 4.5.

4.3.3 Continuity over transitions:

An important intuition about change in the world is that it happens gradually, in a continuous way. Discontinuous jumps in values, derivatives or inequality relations do not represent valid behavior. GARP is an explicit mechanism theory (Forbus, 1988) and the central principle for changes is that the causal primitives determine the derivatives of quantities. Changes in values and inequalities follow from these indications of movement. When a value or inequality is not explicitly altered by a transition it must remain unchanged. Derivatives themselves can also change of course, but no explicit mechanism is present to determine these changes. No second order derivatives are present in GARP so changes of derivatives cannot be predicted. Therefore derivatives are given one degree of freedom over each transition. For example, a derivative that is *plus* in one state must be greater or equal to *zero* in the next. Note that this means derivatives can change whenever some other change causes a transition to another

state, but they cannot cause a state transition themselves while they are one of the variables upon which different states are distinguished. In GARP 2.0 a strict implementation of the continuity principle is applied and also inequalities concerning derivatives are subject to this regime. For example: an inequality $\delta X > \delta Y$ changes into $\delta X \geq \delta Y$ in the next state unless it is explicitly labeled as a constant[28].

4.4 Transitions: Determining Change

The first step in the Find Transitions inference (figure 2.6) is the termination procedure. The task of this procedure is to identify elements in the state description that are changing and can cause that particular state of behavior to terminate and turn into another state. For each changing element a so-called 'termination' is generated. These terminations form the building blocks of actual transitions. Important elements of each termination are its name, conditions, and results. The conditions are replaced by the results when applying the transition.
Three types of terminations can be distinguished:

- Values change and cause a termination when they are moving towards the next value on their quantity space.
- Inequalities change and cause a termination when involved quantities are moving in a different direction and/or at a different speed.
- The rule interpreter in GARP 1.7.2 provides the user the possibility of designing termination rules for capturing domain specific behavior. An example is that in a model of a boiler assembly, water and steam may be modeled as separate entities. A termination could remove one of these entities as it turns into the other.

As mentioned in section 3.2 the termination procedure in GARP 1.7.2 has it's shortcomings. Firstly the user level representation is used as a reference instead of the internal mathematical model[29]. Secondly the rule format does not distinguish between purely conditional statements and conditional statements that need to be replaced with results. And thirdly value changes and inequality changes describing the same event are generated separately. To solve these problems GARP 2.0 uses a dedicated termination procedure instead of a rule interpreter. The internal mathematical model is used to check derivative conditions. Equivalent value and inequality terminations are detected together. In the next two sections a more detailed description of all possible value- and inequality terminations is given. Some extra terminations are provided in GARP 2.0, as well as a mechanism for assuming change in undetermined cases. Constant[30] values and inequalities cannot terminate and are therefore kept out of the termination procedure. The possibility of interpreting domain specific termination rules is retained in GARP 2.0. Using this extra interpreter is optional[31] and by default it is not active. This setting is chosen not to mistake rules in older models for domain specific rules. The format of terminations is identical in GARP 2.0 to that of GARP 1.7.2 but the conditions field is only used for descriptive elements that need to be replaced by results. Purely conditional statements are present in the algorithm itself. To provide the user with information on the reasoning steps made the tracer is updated to show these conditions in action. An example trace for the value change of a quantity X and the change of an inequality $X < Y$ is given below:

[28] See section 4.3.1 on constants.
[29] See section 2.4 on inequality reasoning.
[30] See section 4.3.1 on constants.
[31] See section 4.1 on flexibility.

```
determining terminations for state 1
value termination found for: value(x1, unk, plus, min)
-- quantity space conditions: meets(x1, [point(zero), interval(plus)])
-- equivalent inequality found, terminating: greater(x1, zero), to:
equal(x1, zero)

termination found for: smaller(x1, y1), to: equal(x1, y1),
-- derivative conditions: d_greater(x1, y1) is known or derivable.
```

4.4.1 Value terminations:

The termination procedure in GARP 2.0 finds all value changes defined in the rules used with GARP 1.7.2. Table 4.1 shows an overview of these standard value terminations.

Name:	**Value conditions:**	**Derivative conditions:**	**Quantity space conditions:**	**Value Results:**	**Derivative Results:**
to_point_above(Q)	*Q is at interval(I)*	$\delta Q > zero$	*point(P) is directly above interval(I)*	*Q is at point (P)*	$\delta Q \geq zero$
to_point_below(Q)	*Q is at interval(I)*	$\delta Q < zero$	*point(P) is directly below interval(I)*	*Q is at point (P)*	$\delta Q \leq zero$
to_interval_above(Q)	*Q is at point (P)*	$\delta Q > zero$	*interval (I) is directly above point (P)*	*Q is at interval(I)*	$\delta Q \geq zero$
to_interval_below(Q)	*Q is at point (P)*	$\delta Q < zero$	*interval (I) is directly below point (P)*	*Q is at interval(I)*	$\delta Q \leq zero$

Table 4.1: Value Terminations

When a value termination is found, an equivalent inequality for that value is searched. If such an inequality is found it will terminate as well. This extra termination is integrated in the original value termination. The extra conditions and results for these augmented value terminations are shown in table 4.2. For example: quantity X has the landmark value V, and the termination *to_interval_above(X)* is found. If the relation $X = V$ is present in the user level mathematical model it will terminate to $X > V$. As said before, in the rule-based procedure of GARP 1.7.2 this produces two terminations that need to be merged in the ordering procedure.

Name:	**Relation conditions:**	**Relation Results:**
to_point_above(Q)	*Q < point(P)*	*Q = point(P)*
to_point_below(Q)	*Q > point(P)*	*Q = point(P)*
to_interval_above(Q)	*Q = point(P)*	*Q > point(P)*
to_interval_below(Q)	*Q = point(P)*	*Q < point(P)*

Table 4.2: Augmented Value Terminations

4.4.2 Inequality terminations:

To determine if a binary inequality is changing it is crucial to know if the two quantities involved are moving relatively to each other. In GARP 1.7.2 the user level mathematical model was used to determine if these derivative conditions hold. This requires an enumeration of all possible combinations of derivatives. For example, the inequality $P = Q$, changes to $P > Q$ in the following situations:

- $\delta P = plus$ *and* $\delta Q = min$,
- $\delta P = plus$ *and* $\delta Q = zero$,
- $\delta P = zero$ *and* $\delta Q = min$,
- $\delta P = plus$ *and* $\delta Q = plus$, $\delta P > \delta Q$,
- $\delta P = min$ *and* $\delta Q = min$, $\delta P > \delta Q$,

Note that the inequality on the derivatives in the last two situations needs to be explicitly present in the user level mathematical model. If it is derivable from other constraints, but not explicitly provided by the input scenario or a model fragment the termination will not be generated. And consider the following situation: $\delta P = plus$ and it is derivable that $\delta Q \leq zero$. Again the termination would not be generated. In GARP 2.0 this problem is solved because the internal mathematical model is referenced. In our example the only condition is that $\delta P > \delta Q$ is derivable. This condition includes all situations outlined above. Table 4.3 shows conditions and results for the regular inequality terminations.

Name:	**Relation conditions:**	**Derivative conditions:**	**Relation Results:**	**Derivative Results:**
from_equal_to_greater(P,Q)	$P = Q$	$\delta P > \delta Q$	$P > Q$	$\delta P \geq \delta Q$
from_equal_to_smaller(P,Q)	$P = Q$	$\delta P < \delta Q$	$P < Q$	$\delta P \leq \delta Q$
from_greater_to_equal(P,Q)	$P > Q$	$\delta P > \delta Q$	$P = Q$	$\delta P \geq \delta Q$
from_smaller_to_equal(P,Q)	$P < Q$	$\delta P < \delta Q$	$P = Q$	$\delta P \leq \delta Q$

Table 4.3: Regular Inequality Terminations

As noted in section 4.3.1 all inequalities concerning three quantities or more are assumed to be constants. In GARP 1.7.2 the same assumption is made for the weaker constraining inequalities $\{\geq, \leq\}$. GARP 2.0 has the possibility to generate terminations for binary weak inequalities. This feature is optional and by default it is not active because most models use these inequalities as constraints. The terminations are shown in table 4.4. Note that in one direction of change two terminations are possible. And as can be seen the terminations defined here move from a weak to a specific relation. Another approach could have been only to have terminations from $\geq$ to $\leq$ and vice versa. This approach was not chosen for it adds little extra reasoning capability and stronger inferences can be made. If, for example, $P \geq Q$ is known, then either $P > Q$ or $P = Q$ holds. If $\delta P > \delta Q$ is also known, the latter relation immediately[32] terminates into $P > Q$ and in the former case $P > Q$ already holds.

[32] See section 4.5.3 on epsilon ordering.

Name:	Relation conditions:	Derivative conditions:	Relation Results:	Derivative Results:
from_greater_or_equal_to_greater(P,Q)	$P \geq Q$	$\delta P > \delta Q$	$P > Q$	$\delta P \geq \delta Q$
from_greater_or_equal_to_equal(P,Q)	$P \geq Q$	$\delta P < \delta Q$	$P = Q$	$\delta P \leq \delta Q$
from_greater_or_equal_to_smaller(P,Q)	$P \geq Q$	$\delta P < \delta Q$	$P < Q$	$\delta P \leq \delta Q$
from_smaller_or_equal_to_greater (P,Q)	$P \leq Q$	$\delta P > \delta Q$	$P > Q$	$\delta P \geq \delta Q$
from_smaller_or_equal_to_equal(P,Q)	$P \leq Q$	$\delta P > \delta Q$	$P = Q$	$\delta P \geq \delta Q$
from_smaller_or_equal_to_smaller(P,Q)	$P \leq Q$	$\delta P < \delta Q$	$P < Q$	$\delta P \leq \delta Q$

Table 4.4: Weak Inequality Terminations

4.4.3 Generating behavior in undetermined situations:

One can wonder what should happen in a simulation when derivative conditions do not hold, but neither are they contradicted. For example the inequality $P = Q$, changes to $P > Q$ if $\delta P > \delta Q$ is derivable and it does not change if $\delta P = \delta Q$. In case the relation between both derivatives is unknown, change is possible but it cannot be inferred. An example of a situation where a termination is required in such a case comes from population ecology. This simple model describes a dying population with three quantities: the number of individuals, the born rate and the death rate. The situation is shown in figure 4.7, note that the quantities are decreasing because born is smaller than dead[33]. Intuitively, a possible transition from this state is that all quantities become zero. This transition is impossible however, because it is inconsistent with the inequality between born and dead. In this case a termination (from_smaller_to_equal) of this inequality is welcome. Note that the relevant derivative condition ($\delta Born > \delta Dead$) is not in contradiction with the state.

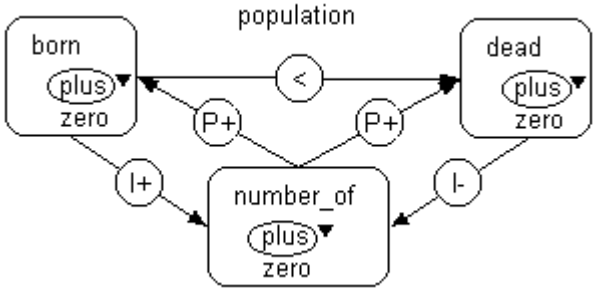


Figure 4.7: A Dying Population

GARP 2.0 features a mechanism for assuming behavior in such undetermined situations. In case the derivative conditions for a specific value or inequality termination are not met but neither are they contradicted, then they are assumed and the termination is generated. This means that for an arbitrary quantity X that is not in an extreme value, two value terminations will be generated when no information on the derivative is present. If on the other hand a weak relation: $\delta X \geq zero$ or $\delta X < zero$ is derivable, only the consistent termination is generated. And of course if strong derivative information is present no assumptions are made. The mechanism works in a similar way when generating inequality terminations. The behavior assumption mechanism is optional[34]. Separate algorithm option switches are provided for value and inequality terminations. The default setting for both switches is off.

[33] See section 4.2 on influence resolution.
[34] See section 4.1 on flexibility.

4.5 Transitions: Ordering Changes

The second step in the Find Transitions inference (figure 2.6) is the ordering procedure. The input for this step is the set of terminations (independent elementary changes) generated by the termination procedure described in section 4.4. All different combinations of these terminations form all possible transitions. If all these branches in the search space are generated and successively checked there is a great danger of combinatorial explosion[35]. On top of that some types of behavior take precedence over other types of behavior due to the point-interval nature of quantity spaces[36]. Subsequently the goals of the ordering process are the following:

- To narrow down the search space by cutting off prospectless branches.
- To give precedence to behavior happening immediately over other behavior.
- To generate all remaining branches in the search space.

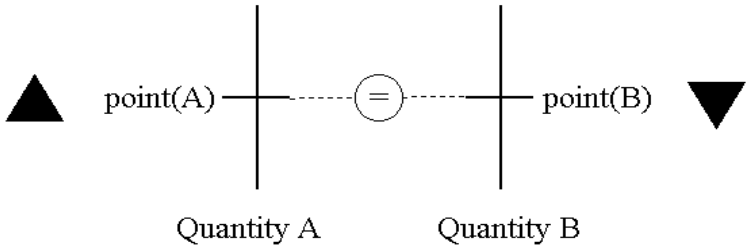
The problems associated with the rule-based procedure used in GARP 1.7.2 have been discussed in section 3.2. A short summary: The semantics of the rule format are limiting, because they only allow fully merging two terminations or removing a termination. The inequality reasoning capacities of GARP cannot be used. Negation cannot be expressed. And ordering rules can only consider two terminations at a time. Further considerations are that domain specific ordering rules are used in large simulations and a comprehensive implementation of all possible ordering concepts is needed to remove the need for such extra rules. To solve these problems GARP 2.0 makes use of special format to record constraints on combinations of terminations. Dedicated procedures implement each ordering concept and generate these constraints. In this approach ordering knowledge is no longer declaratively captured in ordering rules. Therefore a tracer can illustrate the reasoning process in GARP 2.0. Examples of such reasoning traces can be found in section 5.1.

4.5.1 Storing constraints on termination combinations:

As mentioned, a more expressive way of recording constraints on valid combinations of terminations is needed then simply merging two terminations or removing one. Consider for example the situation in figure 4.8, where two quantities are equal and at equal points, and moving in opposite directions.

[35] The search space associated with the search for valid successor states is meant here. Note that in the component centered approach (De Kleer & Brown, 1984) first all possible valid states are computed and therefore the search for valid transitions is somewhat bounded. However, combinatorial explosion is an issue in this approach as it can arise when combining all components in their different behavior modes to compute all possible valid states in the first place.

[36] See section 4.5.3 on epsilon ordering.



Relevant Terminations:

- T1: to interval above: A
- T2: to interval below: B
- T3: from equal to greater: A, B

Figure 4.8: Terminations For Two Quantities

These two quantities cannot become unequal unless at least one of them moves away from its point. And if any one of the two quantities leaves its point then they must become unequal as well. Note that these constraints cannot be captured using full merges of terminations and / or removing terminations. All three terminations produce an inconsistency when applied alone but several different combinations produce a consistent situation. For clarity, table 4.5 shows all consistent and inconsistent combinations.

Valid Combinations:	**Invalid Combinations:**
T1 & T3, *T2 & T3,* *T1 & T2 & T3*	*T1,* *T2,* *T3,* *T1 & T2*

Table 4.5: Validity Of Different Termination Combinations

To use this knowledge a representation is needed for storing such information. Multiple inferences are made supplying information using different ordering concepts, so also a method is needed to update the information. And finally a method is required that, given all this information, generates all possible combinations of terminations.

The following representation is used in GARP 2.0: All terminations are indexed at the beginning of the ordering procedure, and every inference applying an ordering concept produces a set of valid and invalid combinations of indexes. Table 4.5 could be such a set. All these distinct sets are kept intact, and are stored in a large table. Updating the information with a new set of constraints is nothing more then storing this set in the table. Since each inference in this process considers a different aspect of the situation, one can imagine that contradictory entries are present in different sets. For example, the situation sketched above considers simple algebraic consistency. But another inference might evaluate terminations *T1* and *T2* in the context of a value correspondence between *A* and *B*. In this situation, *T1* and *T2*

cannot happen separately. The following constraints are inferred: valid ({*T1*, *T2*}), invalid({*T1*}, {*T2*}). As can be seen, the combination; {*T1*, *T2*}, is valid in one set and invalid in the other. Which constraint is true? Closer inspection answers this question. This combination corresponds to the event that both quantities change value. As can be seen in figure 4.8 this event leads to an inconsistent situation if not combined with *T3*. Correspondences cannot change these algebraic facts. The inference on correspondences only shows us *T1* and *T2* cannot occur separately. In general, this approach requires that all inferences supply definite knowledge on inconsistent combinations. That means that if a set of constraints indicates a combination to be inconsistent, then no other set of constraints can change this verdict. On the other hand, if a set of constraints considers a combination to be consistent, any other set of constraints can prove otherwise. So the constraint of inferences being certain on invalid combinations makes their results independent information sources on ruling out combinations. But not until all information has been considered is a combination surely consistent[37].

After finishing all ordering inferences the set of all possible transitions is generated. To do this the crossproduct of the set of terminations is taken. For every element in this crossproduct the corresponding index combination is evaluated using the table containing all sets of constraints. The algorithm for this basic validity check is given below:

Basic validity check:

Given:

- A table *T* containing sets *S* of index-combinations labeled consistent and inconsistent. each labeled index-combination is called an item: *I*.
- An index-combination *C* that needs to be evaluated.

1) If no more *S* in *T*, then *C* is valid.

2) Take the next set *S* from *T*. Create an empty set of active relevant items: *R*

3) For every item I_j in *S*:
   - If I_j is a subset of *C* then place I_j in *R*
     and remove all other items I_x that are a subset of I_j from *R*
   - If I_j is a not subset of *C* then ignore I_j.

4) If *R* contains an item *I* labeled inconsistent then *C* is invalid
   Else: Return to 1)

This validity check needs to be fast for the cross product obviously can be large, and for every combination, potentially the whole table has to be checked. Speed is achieved in two ways. Firstly because table consists of independent sets a final decision can be reached before processing the whole table in most cases of invalid combinations. A second technique is applied to gain even more time. Any inconsistent item that has no consistent superset item in its set of constraints is considered a fatal constraint. No superset item can ever push it out of the set of relevant items. Fatal constraints are selected from the table before evaluating the crossproduct and the basic validity check is preceded by a check of the set of fatal constraints. If any item in this set is a subset of the combination to be evaluated then it is invalid. This method of checking constraints has proven in practice to be computationally tractable.

[37] Altough a transition can still prove inconsistent during state specification, because ordering principles have only a local view of the situation.

4.5.2 Application of ordering principles:

Different ordering principles can be applied to reason about valid combinations of changes. These information sources have to be reliable. Inferring over restrictive constraints will prevent the transition procedure from discovering all possible state transitions. Reliability in this context comes down to two criteria:

- The information source must be stable.
- The information source must supply definite information on invalid combinations.

The state description supplies all information, but many elements of the state description can change over a transition. Therefore the criterion of stability requires that the information source can be assumed to remain present in any successor states. For example, an inequality between quantities can change and is therefore in general not a stable information source. An inequality between landmarks on the other hand cannot change and is therefore a stable information source. The second criterion follows from the representation for storing the information that is defined in the previous section. An information source may only classify a combination as invalid if no other added knowledge can change this verdict.

Different information sources can be used, each will be discussed in turn.

- Correspondences supply information on valid value combinations
- Landmark relations supply information on valid value and inequality combinations.
- Constant inequalities[38] supply information on valid value and inequality combinations.

A minor information source is a table on mutual exclusive terminations. The behavior assumption mechanism described in section 4.4.3 can generate changes in two directions for a quantity or inequality and these terminations cannot be combined. Likewise, the weak inequality terminations described in section 4.4.2 also comprise mutual exclusive alternatives.

Correspondences:

Value[39] correspondences are causal primitives like influences and proportionalities. Together they form the causal model of the system being described. Causal mechanisms in a system normally[40] do not change because the system shows a different behavior. Therefore it is safe to assume that correspondences satisfy the stability criterion. Directed and undirected correspondences are distinguished in GARP, but undirected correspondences can be seen as two directed ones (in opposite directions). When a directed correspondence is active; the quantity has the conditional value, then it must be satisfied for a state to be consistent; the other quantity must have the consequential value. This means that if a combination of value changes "activates" a correspondence but it does not satisfy the correspondence then the combination is definitely invalid. Therefore, the second criterion for information sources is satisfied as well.

To extract all information from this source, GARP 2.0 determines all corresponding value pairs (NB. directed) defined in the state. Each pair is evaluated separately resulting in a set of

[38] See section 4.3.1 on constants.
[39] Note that quantity correspondences can be seen as defining a value correspondence for every value pair in two quantity spaces. Both value- and quantity correspondences are referred to in this subsection by the term correspondence.
[40] Technically it is possible to construct models in wich correspondences dissapear or change over transitions but this is not a common practice.

constraints. Five relevant situations are possible for a corresponding value pair, these are pictured in figure 4.9. In each situation, different combination constraints are derived. Sometimes, a termination is completely invalid, in that case it is removed permanently from the set of terminations. In all situations different from these five situations no information can be inferred because either the correspondence cannot be activated, or the correspondence is already satisfied and no termination is present to change this.

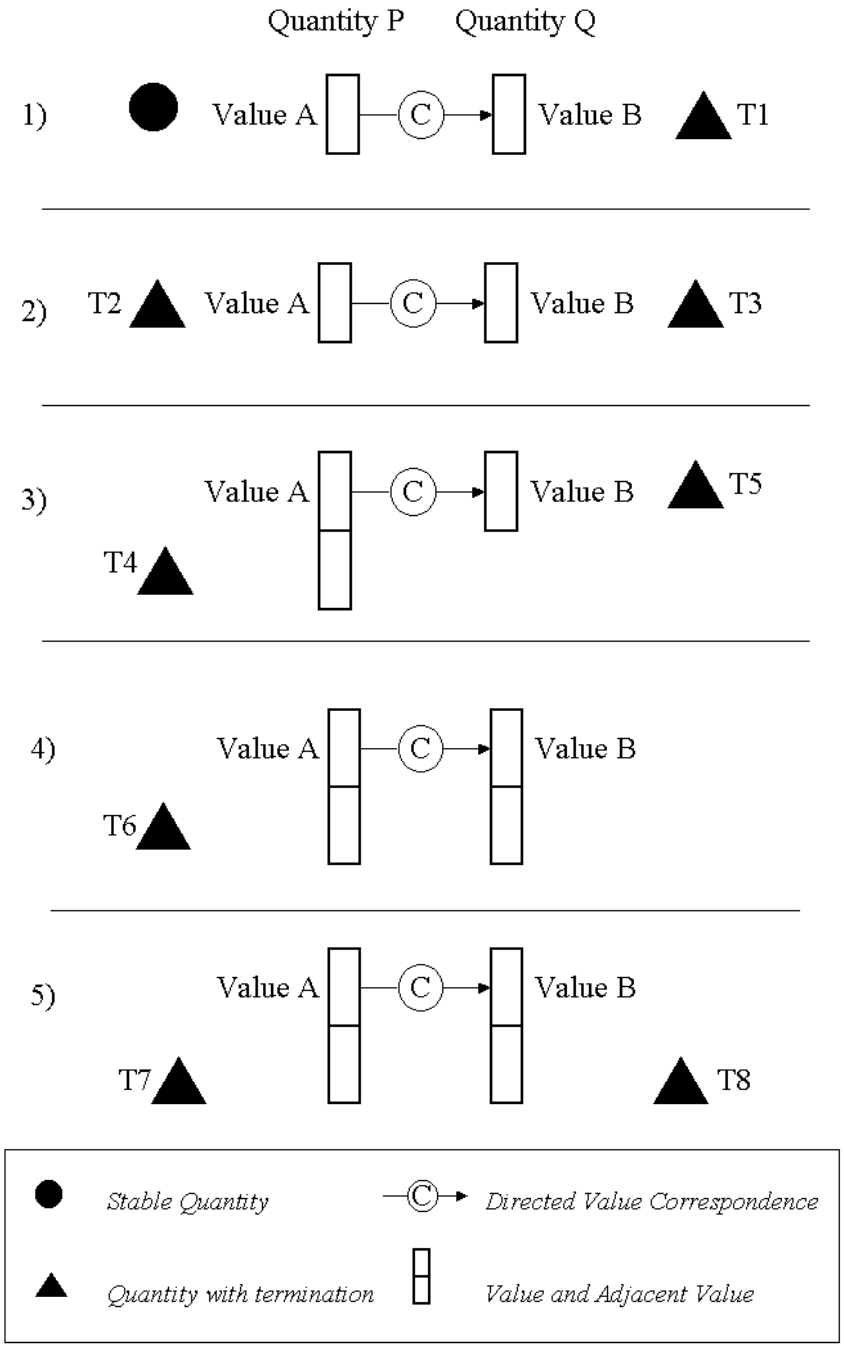


Figure 4.9: Corresponding Values And Terminations

Note that in this picture no distinction between points and intervals is made, because this does not supply extra knowledge in this context. Value terminations are indicated with a triangle and although these are all pointing upwards this does not mean they represent only changes to

higher values. The important distinction is if the quantities are leaving or arriving at the value in question. The following interpretation for the five situations is used:

- Situation 1) Quantity *P* is at value *A* and no termination is present for *P*. This means that the correspondence is active and will remain active. Note that quantity *Q* must be in value *B* in this state. Results: All value terminations for *Q*, such as *T1*, are removed. No further constraints are derived.
- Situation 2) The value correspondence is active, but now quantity *P* may leave value *A*. Any value termination *T3* for *Q* may only happen together with a termination *T2* for P, deactivating the correspondence. Results: consistent({*T2*}, {*T2*, *T3*}), inconsistent ({*T3*}).
- Situation 3) The value correspondence will become active if termination *T4* towards the conditional value *A* is applied. Quantity *Q* is already at the consequent value and may leave *B* trough some termination *T5*. These terminations may not happen together. Results: consistent({*T4*}, {*T5*}), inconsistent ({*T4*, *T5*}).
- Situation 4) The value correspondence will become active if termination *T6* towards the conditional value *A* is applied. Quantity *Q* is not at value *B* and has no termination towards value *B*. Results: Remove termination *T6*, no further constraints derived.
- Situation 5) The value correspondence will become active if termination *T6* towards the conditional value *A* is applied. Quantity *Q* is not at value *B* but has termination *T7* towards value *B*. Termination *T6* may only happen together with *T7*. Results: consistent({*T7*}, {*T6*, *T7*}), inconsistent ({*T6*}).

Note that situation 4 is an example of a 'rule' requiring negation as mentioned in sections 3.2 and 4.5 on the problems of the rule format in GARP 1.7.2. Negation is needed here to express there is no termination towards value *B* in the set of terminations. Only this fact can justify the immediate removal of *T6*. In GARP 1.7.2 undirected value correspondences were used to derive full merges between terminations. Figure 4.10 illustrates such a situation. Both terminations *T1* and *T2* will have to happen together. So merging these terminations is justified here. Merging terminations is beneficial because it reduces the size of the terminations set and therefore shortens the time needed to compute the final crossproduct of terminations[41]. By considering only one directed corresponding value pair at a time GARP 2.0 cannot derive full merges. The constraints derived from evaluating both value pairs separately implement the same principle though. In the final crossproduct of terminations both terminations will only appear together. Furthermore, because of the behavior assumption mechanism described in section 4.4.3, it is possible that two value terminations are present for a quantity. In this case no merging of terminations is justified and constraints are needed again. Therefore GARP 2.0 does not check more then one corresponding value pair at a time.

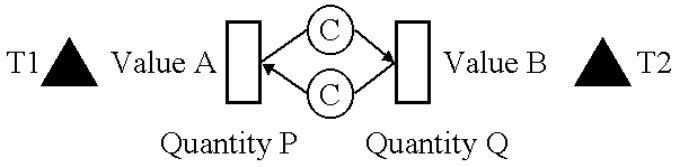


Figure 4.10: Undirected Value Correspondence

[41] See section 4.5.1 on storing constraints on termination combinations.

Landmark relations:

As described in section 4.3.2 inequalities between landmarks are constant throughout a simulation and algebraic reasoning can be used to determine inconsistent situations and consequently inconsistent combinations of terminations. This means both criteria for an ordering information source are satisfied[42].

Binary landmark relations provide a context of two quantities. In such a context many different situations can occur. Each quantity has a current value and maybe an inequality between the quantity values is present. One or more inequality terminations can be generated as well as on or more value terminations. An enumeration of generic situations in which combination constraints can be derived is not feasible. GARP 2.0 uses a procedural approach instead and its inequality reasoning capabilities provide the generic knowledge on consistent and inconsistent situations. To provide the user with insight in the reasoning process a tracer is provided. The user can be expected to understand the simple algebraic inferences made, but if needed a trace of the inequality reasoning process can be generated as well. Another example of the kind of reasoning performed by this procedure was given in the previous section (fig 4.8). A sample reasoning trace for this situation is given below:

```
determining combination constraints
for context:
    a1, b1
    current values:
    [point(a1), point(b1)]
    terminations:
   [[from_equal_to_greater(a1, b1)],
    [to_interval_above(a1)],
    [to_interval_below(b1)]]
    constant:
    [equal(point(a1), point(b1))]
testing combination:
    from_equal_to_greater(a1, b1)
    - inconsistent combination.
testing combination:
    from_equal_to_greater(a1, b1)
    to_interval_above(a1)
    - consistent combination.
testing combination:
    from_equal_to_greater(a1, b1)
    to_interval_above(a1)
    to_interval_below(b1)
    - consistent combination.
testing combination:
    from_equal_to_greater(a1, b1)
    to_interval_below(b1)
    - consistent combination.
testing combination:
    to_interval_above(a1)
    - inconsistent combination.
testing combination:
    to_interval_above(a1)
    to_interval_below(b1)
    - inconsistent combination.
testing combination:
    to_interval_below(b1)
    - inconsistent combination.
```

The following procedure is used to determinate constraints on termination combinations for a binary quantity context:

Given:

- Two quantities with their current values
- The indexed set of relevant value- and inequality terminations, *T*
- The set of relevant landmark relations, *R*

An inequality termination is relevant if the inequality is between both quantities in the context. A landmark relation is relevant if it concerns either current values or values resulting from value terminations.

[42] See section 4.5.1 on storing constraints on termination combinations.

1) Determine the crossproduct of *T*, resulting in a set *C*, of all possible combinations of terminations. Label all combinations with their index-combination.
2) Take a compound termination *I* from *C,* supply the inequality reasoning procedure with:
   - All results of *I*,
   - *R,*
   - If no inequality termination is part of *I*: the original inequality.
   - If no value termination for a quantity is part of *I*: the current value of the quantity.
3) Record the result of the inequality reasoning procedure:
   If contradiction is derived, record the index combination of *I* with the label inconsistent.
   If no contradiction is derived, record the index combination of *I* with the label consistent.
4) Return to 2) until *C* is empty.
5) Return the set of constraints.

Note that in this procedure the original inequality is asserted in all combinations that do not contain an inequality termination. In case an inequality termination is present it will either be applied or the original inequality may be assumed to remain present. In fact, the set of active model fragments could change over a transition and this could lead to the change of this inequality also. To depend on such a transition is poor modeling though. In GARP an inequality should change through a termination. This provides an explicit mechanism and explanation for the change.

Because the sets of terminations are limited in size for a two-quantity context the complexity of determining consistency for every possible combination is not problematic. For extra efficiency, the knowledge present in the combinations table at this point is used. The possible combinations are generated by taking a crossproduct whilst checking fatal combination constraints[43]. Note that complete sets of constraints cannot be used at this point. They are generated using a particular subset of terminations and their information is only valid for the same set or supersets. Consider for example the set of constraints in table 4.6 corresponding to the situation shown in figure 4.8 in the previous section.

Valid Combinations:	**Invalid Combinations:**
T1 & T3, *T2 & T3,* *T1 & T2 & T3*	*T1,* *T2,* *T3,* *T1 & T2*

Table 4.6: Validity Of Different Termination Combinations

Now, if another quantity context would consider for example the terminations *T1*, *T5* and *T6*, then all combinations that include *T1* would not be tested. As can be seen *T1* is an invalid combination in this set of constraints and superseding valid combinations all require termination *T3*, which is not associated with the context in question. Things can go wrong in this case: In the final crossproduct, the combination {*T1*, *T3*, *T5*} may be generated and may later prove inconsistent because of a reason that could have been detected at this point. This scenario is not favorable and therefore only fatal combinations are used at this point. They have information value exactly for those combinations to which they apply.

[43] See section 4.5.1 on storing constraints on termination constraints.

Another gain in efficiency is made by only considering those quantity contexts that are likely to produce results. Two requirements have to be satisfied before a context is subjected to the routine described above. Firstly at least one relevant landmark relation has to be present for the quantity context[44]. Secondly at least one relevant inequality termination has to be present. These requirements make sure that at least two relations link both quantities. With only one link between quantities, contradiction can never be derived.

Constant inequalities:

As mentioned in section 4.3.1 inequalities can be labeled as constants and inequalities involving more than two quantities are assumed to be constants in GARP 2.0. These inequalities are used in a similar way as landmark relations to determine invalid combinations of terminations using inequality reasoning. An example is the following situation involving three quantities, all with the quantity space {min, point(zero), plus}:

Given:

1) $A = B - C$ (assumed constant)
2) $B = C$
3) $A = zero,\ \delta A = plus$
4) $B = plus,\ \delta B = zero$
5) $C = plus,\ \delta C = min$

Terminations:

T1 *to_interval_above(A)*
T2 *from_equal_to_greater(B, C)*
T3 *to_point_below(C)*

Simple algebra tells us that *T1* and *T2* cannot occur separately and *T3* cannot occur without *T1* and *T2* occurring. The set of constraints shown in table 4.7 is derived, the reader is invited to check these inferences:

Consistent:	**Inconsistent:**
T1 & T2, *T1 & T2 & T3*	*T1,* *T2,* *T3,* *T2 & T3* *T1 & T3*

Table 4.7: Combination Constraints

[44] The *zero = zero* relation which is implicit in GARP also counts ahere.

The following trace of the reasoning process is given in this case:

```
 determining combination constraints
 for context:
     [a1, b1, c1]
     current values:
     [zero, plus, plus]
     terminations:
    [[from_equal_to_greater(b1, c1)],
     [to_interval_above(a1)],
     [to_point_below(c1)]]
     constant:
     [equal(a1, min(b1, c1))]
testing combination:
     from_equal_to_greater(b1, c1)
     - inconsistent combination.
 testing combination:
     from_equal_to_greater(b1, c1)
     to_interval_above(a1)
     - consistent combination.
 testing combination:
     from_equal_to_greater(b1, c1)
     to_interval_above(a1)
     to_point_below(c1)
     - consistent combination.
testing combination:
     from_equal_to_greater(b1, c1)
     to_point_below(c1)
     - inconsistent combination.
testing combination:
     to_interval_above(a1)
     - inconsistent combination.
testing combination:
     to_interval_above(a1)
     to_point_below(c1)
     - inconsistent combination.
testing combination:
     to_point_below(c1)
     - inconsistent combination.
```

Similar to the procedure considering landmark relations, this procedure generates a quantity context with relevant terminations and relevant landmark relations for every constant inequality. And to evaluate these quantity contexts the same algorithm is used. Requirements for a quantity context to be evaluated are that it has not been evaluated by the previous procedure and that at least one termination is present. The application of this procedure is optional because the assumption of 'larger' inequalities being constant may be invalid in some models. The default setting is that the procedure is active.

4.5.3 Epsilon ordering:

The epsilon-ordering rule (De Kleer & Brown, 1984) can be used to determine if some terminations take precedence over others. The principle behind this rule is that if a quantity *Q* is changing from a point *P* to an interval *I* this will happen immediately. This is because it is moving and therefore it can only be at a point for an instant. When quantity *Q* is changing from interval *I* to point *P*, this change will not be immediate. It will always take some amount of time because we are dealing with continuous functions and therefore there is always some ε between *Q* and *P* ($Q > \varepsilon > P$). The same argument can be given for changes from equality to inequality versus changes in the other direction[45]. All immediate terminations take precedence over non-immediate terminations.

In GARP 1.7.2 epsilon ordering is done after applying other ordering concepts, but before determining all possible terminations combinations. The rule based procedure compares 2 terminations at a time, removing non-immediate terminations one by one if an immediate termination is present. In GARP 2.0 epsilon ordering is done after determining all possible combinations because in this procedure all combination constraints[46] are applied. If epsilon ordering is done before the application of these constraints a problematic situation can occur. It is possible that all immediate terminations that take precedence over others are later

[45] For the equality A = B define a quantity Q with the sign quantity space {min, point(zero), plus}. Define Q as Q = A – B, and therefore Q = zero when A = B. If a change occurs to to A > B or A< B then the value of Q changes to Q = plus and Q = min respectively. This a change from a point to an interval, thus it is an immediate change. A change in the opposite direction is not immediate for it is a change from an interval to a point.

[46] See section 4.5.1 on storing constraints on termination combinations.

removed because they are found to be inconsistent with the combination constraints. In this case an empty set of possible transitions is generated, while there are valid non-immediate terminations to form possible transitions. The following procedure is used to apply epsilon ordering: A possible transition (combination of terminations) is immediate if it contains only immediate terminations. If the set of possible transitions contains at least one immediate possible transition then all non-immediate possible transitions are removed.

In GARP 2.0 some new inequality terminations[47] are introduced. Their epsilon types are given in table 4.8. The only termination that is known to be immediate is when a '≥' relation is changing to '>'. Either it already is '>' or it must have been '=' in which case it is a change from equality. Therefore the termination is immediate. The same argument is valid for the '≤' relation changing to '<'. Terminations in the other direction are non-immediate. The change to equality is non-immediate for the normal reasons and the change to inequality must also be non-immediate because otherwise it would always take precedence over the former type.

Name:	**Epsilon Type:**
from_greater_or_equal_to_greater(P,Q)	*immediate*
from_greater_or_equal_to_equal(P,Q)	*non-immediate*
from_greater_or_equal_to_smaller(P,Q)	*non-immediate*
from_smaller_or_equal_to_smaller(P,Q)	*immediate*
from_smaller_or_equal_to_equal(P,Q)	*non-immediate*
from_smaller_or_equal_to_greater (P,Q)	*non-immediate*

Table 4.8: Epsilon Types For '≥' Transitions

4.6 Transitions: Applying Changes

The third step in the Find Transitions inference (figure 2.6) is the Close procedure. This procedure has two tasks. Firstly it finishes the compound terminations assembled by the ordering procedure into transition-scenarios by applying continuity rules[48]. Secondly it determines successor states for these transition scenarios. The principles of the continuity regime are/have been described in section 4.3.3 and this regime is applied to all quantities not appearing in the transition results. For these quantities the continuity regime is applied upon generating the terminations that form the transitions.

As mentioned in section 3.2, in some situations the application of transition scenarios is problematic in GARP 1.7.2. This is done by the Determine States inference (figure 2.5). In a transition scenario all quantity values are determined, but derivatives have some degree of freedom and this can lead to branching: multiple successor states are found. Another theoretical possibility for branching is that a new active process causes a new quantity to be introduced and this quantity can take on different values. Thus, one transition scenario can have multiple successor states. The Subsumption procedure, part of the Determine States inference, compares the transition scenario with existing states and determines if these are valid successors. If existing states are found, these are returned as successor states. The transition scenario is never fully specified in this case. Doing so would lead to the generation of states already existing in the simulation. The indicated problem occurs when only a subset of all possible successor states is previously generated in the simulation. These existing states are found as successors and the other possible successor states are not generated in this step

[47] See section 4.4.2 on inequality terminations.
[48] See section 4.3.3 on continuity.

and therefore not found as successors. This means some valid behavior is not captured in the simulation. Making the transition scenario's deterministic with respect to their successor states cannot solve this problem. Firstly generating scenario's for all possible derivative combinations is undesirable because it enlarges the branching factor in state graphs. Normally, the specification of an input-scenario can lead to undetermined derivatives and these 'summarize' multiple underlying possibilities. Secondly this would not be sufficient because the introduction of new model fragments and quantities cannot be foreseen and this process can also cause branching.

GARP 2.0 features a new Determine States inference to solve this problem. Every transition scenario is always fully specified and the resulting states are matched to the existing states before being saved. If a match is found, this existing state is a successor state. If no match is found the specified state is saved as the successor state. The inference is shown in figure 4.11. A drawback of this solution seems to be that it spends time on specifying already existing states. But, as is shown in section 5.4, in practice this proves not to be problematic. Matching two states is a lighter inference then comparing an input scenario to a state and in case a scenario leads to an inconsistent state, time is wasted by the subsumption procedure to look for existing states, whereas the Specify-Match procedure derives the inconsistency straight away. An algorithm option switch is present in Garp 2.0 to choose between the procedure involving subsumption and the procedure involving complete specification and state matching.

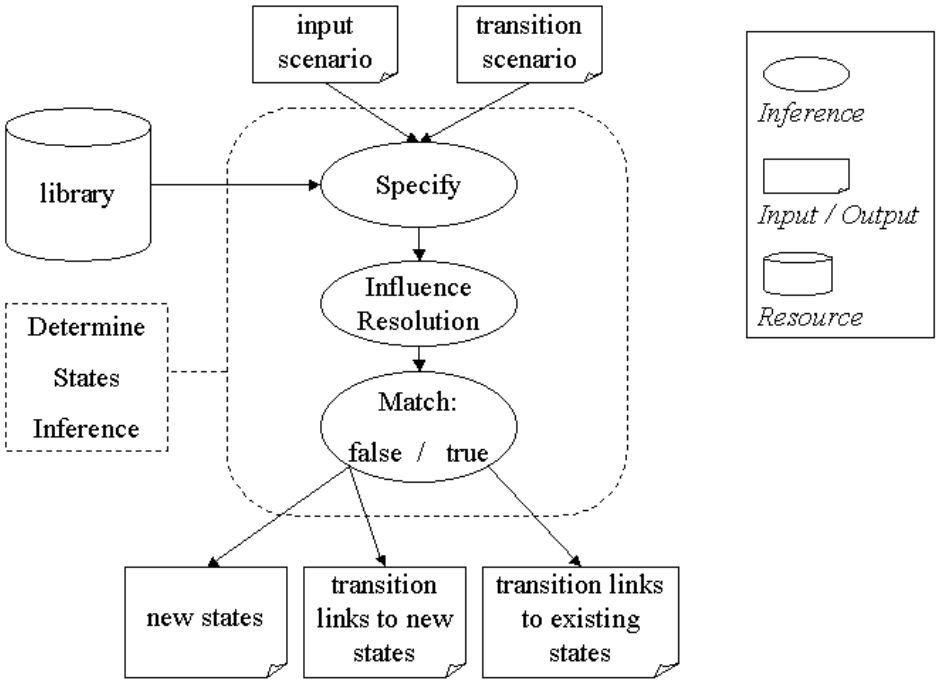


Figure 4.11: Altered Determine States Inference With State Matching

The matching process has the following characteristics:

- the set of system elements must be equal
- the set of active model fragments must be equal
- the set of quantities must be equal
- all values must be equal

- all derivatives must be equal
- the dependencies must match

These requirements are tested in this order going from straightforward checks to a more complicated matching procedure for relations. An unspecified value, indicated by a '?' can only match with another unspecified value. The same is true for derivatives. If this kind of matching would be allowed, problems can occur. For example, a newly generated state with a certain specified derivative could match on a state with an unspecified derivative for this quantity. In this case no behavior for this quantity can be inferred in this state, while in the newly generated version of this state inferring behavior was possible. Note that in the subsumption procedure these kinds of matches are allowed. As mentioned, the matching of dependencies is the most complicated process. Firstly, most dependencies come from model fragments and because the sets of model fragments are known to be equal at this point, all these relations must be equal as well. Other dependencies either come from the input scenario or they are derived landmark relations[49]. Landmark relations are a special case and will be discussed in the next paragraph. Relations in the input scenario will always re-appear in transition scenario's to new states unless they are either explicitly changing through a termination or they are inequalities between derivatives. These can also change because of the continuity regime. This leaves us two categories of relations that can differ between states, given an equal set of active model fragments: binary inequalities and inequalities on derivatives. Relations from the first category are extracted from both state descriptions. The resulting sets are compared: Every relation must be present in the other set or derivable in the internal mathematical model[50] of the other state. Derivative relations are the second category to be compared. They are also extracted from both state descriptions. Included in these sets are the continuity constraints, which are filtered from both internal mathematical models. The resulting sets are required to be mathematically consistent. The inequality reasoning procedure is used for this purpose.

The landmark relations of both states have to make an exact match. The reason for this is that they specify the possible world being modeled as explained in section 4.3.2 on landmark relations. The only change in the description of the possible world may come from deriving new constraints in the course of a simulation branch. When the matching process only requires landmark relations from two states to be consistent the specification of the possible world can be weakened or strengthened in the course of a behavior path without justification. This problematic situation is illustrated in figure 4.12. The input scenario leads to state 1 that is indefinite about modeling possible world 1 or 2. The transitions to state 2 and 3 represent behavior occurring in possible world 1 and possible world 2 respectively and in these states a landmark relation is correctly derived specifying the modeled possible world. Now states 4 and 5 are incorrectly allowed to make a transition to the lesser-specified state 1. As a result the state graph no longer represents valid behavior occurring in the real world. A possible behavior path is (1, 3, 5, 1, 2, 4, etc). which involves a jump to a different possible world. Consider for example the u-tube system. State 1 may model a u-tube with tubes of equal height or a u-tube where the left tube is lower (figure 4.5 left and middle). In states 2 and 3 the relations *maximum(left) = maximum(right)* and *maximum(left) < maximum(right)*, may be derived respectively. Now each state description is clear about which u-tube is modelled. These relations are kept in the transitions to the states 4 and 5. Normally the u-tube system does not show circular behavior, but we can imagine that the transitions from states 4 and 5

[49] See section 4.3.2 on landmark relations.
[50] See section 2.4 on inequality reasoning

back to state 1 represent the event of removing an amount of water from one tube and adding it to the other. This transition brings us back to the starting state. This next state however, should also conservate the knowledge of the relation between both maximum heights. In this problematic example, the transitions to the existing state 1 do not conservate this knowledge. And now, when following a behavior path a u-tube with equal tubes can change into one with unequal tubes and vice versa.

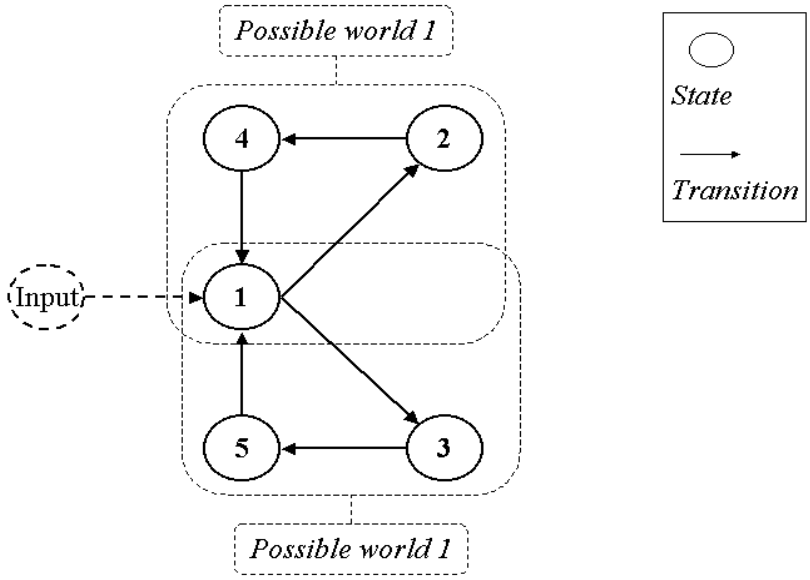


Figure 4.12: State Graph With Overlapping Possible Worlds

The correct state graph for this hypothetical simulation is shown in figure 4.13. States 4 and 5 do not match to the lesser-specified state 1. Instead new states 6 and 7 are generated and the possible behavior paths do not cross possible world boundaries. States 1, 6 and 7 will share the same values, derivatives and model fragments but will have different sets of landmark relations in their state description.

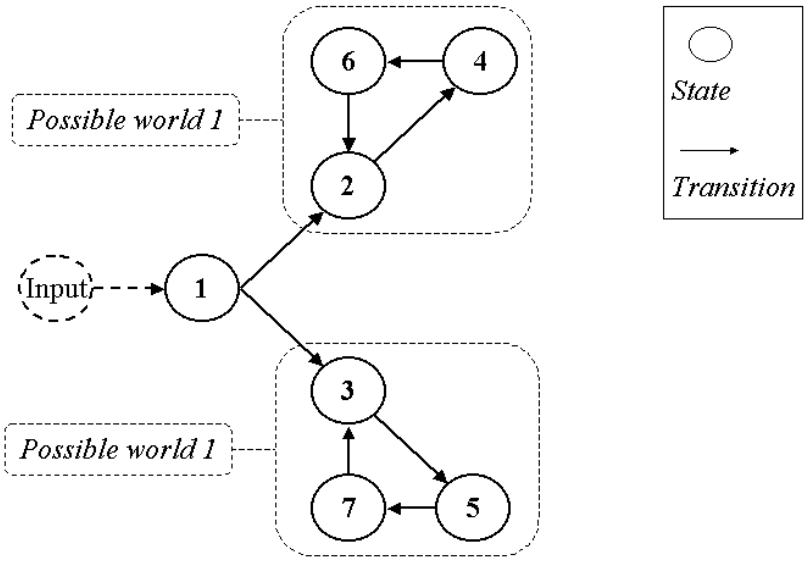


Figure 4.13: State Graph With Separate Possible Worlds

4.7 Exogenous Quantities

As explained in section 3.3 a way to deal with exogenous quantities is needed. A set of different types of behavior an exogenous quantity can exhibit is required as well as a mechanism implementing these behaviors. GARP 2.0 recognizes five behavior types ranging from simple to complex behavior:

- Steady: The quantity remains in its initial value.
- Increasing: The quantity moves up from its initial value to the maximum value of the quantity space. In case this value is a point and quantity space constraints[51] are active it will stop there.
- Decreasing: The quantity moves down from its initial value to the minimum value of the quantity space. In case this value is a point and quantity space constraints are active it will stop there.
- Sinus: The quantity moves up or down it's quantity space until it reaches the maximum or minimum value. There it stops and starts it's way down or up respectively. This behavior continues in an infinite loop.
- Free: The quantity may move up and down freely across its quantity space. It may stop and reverse direction in any value. It may also stop and continue in the same direction.

These patterns cover a large range of behavior, and a quantity behaving in a more complex way probably deserves a model of its own. In general, simplicity is required in this area not to generate enormous intractable and incomprehensible state graphs (Travé-Massuyès et al., 2003). Therefore the *Free* and *Sinus* types should be used with care and only in smaller simulations. Exogenous quantities can be defined in a dedicated model fragment. Keyword labels are used to indicate such a model fragment. This mechanism is described in section 4.8.

The initial value of an exogenous quantity has to be specified using the regular modeling techniques. The implementation of the behavior patterns is done by controlling the derivatives of the exogenous quantities. The derivative of an exogenous quantity is initially set in the Determine States inference (figure 2.5 & 4.11). This is done just before influence resolution takes place. The rules for determining exogenous derivatives are shown in table 4.9. Note that the rules for the Sinus and Free patterns are not deterministic and will therefore produce branching: Multiple states are possible.

Type:	**Value:**	**Result derivative:**
Steady	-	*zero*
Increasing	*< maximum value*	*plus*
Increasing	*maximum interval*	*plus*
Increasing	*maximum point*	*zero*
Decreasing	*> minimum value*	*min*
Decreasing	*minimum interval*	*min*
Decreasing	*minimum point*	*zero*
Sinus	*maximum value*	*plus, zero or min*
Sinus	*> minimum value & < maximum value*	*min or plus*
Sinus	*minimum value*	*plus, zero or min*
Free	-	*plus, zero or min*

Table 4.9: Rules For Setting Exogenous Derivatives

[51] See section 4.11 on quantity space constraints.

After the derivative is set the exogenous quantity joins in the regular transition regime. Thus its value can change by means of normal transitions. In the next state the derivatives of the steady, increasing and decreasing exogenous quantities are again set using the rules in table 4.9. The sinus and free exogenous quantities however have a mechanism to control their movement. Unlike the derivatives of other quantities, which are given freedom to move by the continuity regime, these exogenous derivatives are kept stable by default. Special exogenous terminations are generated for these quantities to explicitly change their derivatives. These are shown in table 4.10. Since the behavior of exogenous quantities is independent these terminations have no conditions concerning the state of the system. There are only conditions that ensure the correct behavior pattern is followed.

Name:	**Value conditions for Sinus type[52]:**	**Derivative conditions:**	**Derivative Results:**
exogenous_up_to_stable(Q)	*Q in maximum value*	*$\delta Q > zero$*	*$\delta Q = zero$*
exogenous_stable_to_down(Q)	*Q in maximum value*	*$\delta Q = zero$*	*$\delta Q < zero$*
exogenous_down_to_stable(Q)	*Q in minimum value*	*$\delta Q < zero$*	*$\delta Q = zero$*
exogenous_stable_to_up(Q)	*Q in minimum value*	*$\delta Q = zero$*	*$\delta Q > zero$*

Table 4.10: Exogenous Terminations

By using the derivatives for implementing exogenous behavior these quantities can be treated as normal quantities when changing value and they can be treated normally in the complete reasoning process. This an advantage over a dedicated mechanism controlling all exogenous behavior including value changes. The ordering process described in section 4.5 distinguishes the special exogenous terminations and these are not considered the regular inferences determining combination constraints. A specific ordering procedure is present for these terminations. Firstly combination constraints are generated for mutually exclusive terminations: A free exogenous quantity cannot start to move up and down simultaneously. Secondly, as described in section 4.9, GARP 2.0 features derivative correspondences. These are used in determining combination constraints on exogenous terminations of different exogenous quantities. This inference is equal to the one using value correspondences to order value terminations described in section 4.5.2. The epsilon type of the terminations that determine the derivatives of exogenous quantities is decided to be non-immediate. These terminations do not have a specific reason for happening and therefore they should not take precedence over others.

4.8 Keyword Labels

To implement the notions of constants and exogenous variables, described in sections 4.3.1 and 4.7 respectively, a labeling system is needed. The assumption-functionality in GARP is used for this purpose. Upon encountering a model fragment with one of these keyword assumptions in its conditions, GARP 2.0 applies the associated procedure implementing the correct functionality. An isa tree of the all keyword assumptions is shown in figure 4.14. As can be seen, a third function is incorporated in GARP 2.0: The keyword assumption *generate_all_values* can be used to make a quantity take on all values in its quantity space.

[52] The sinus pattern only changes direction in its extreme values, the free pattern can change direction in any value.

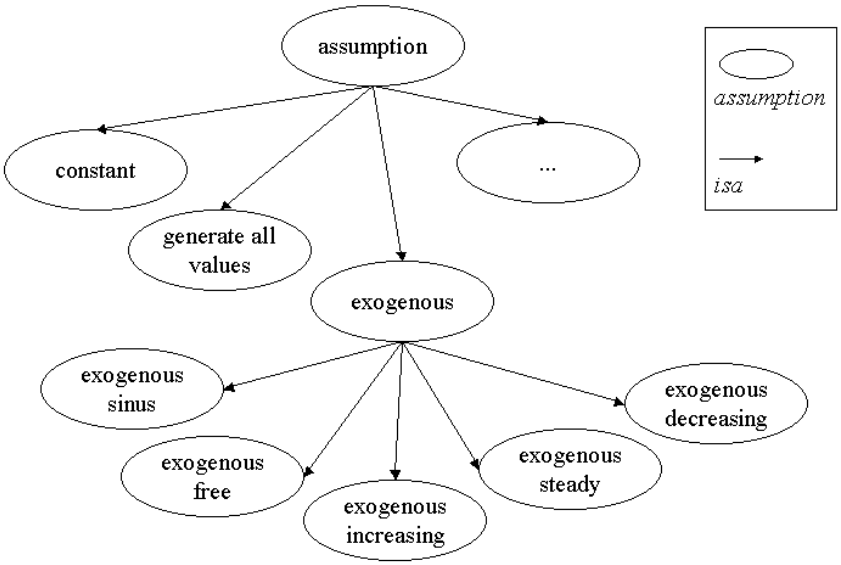


Figure 4.14: Keyword Assumptions In GARP 2.0

A modelfragment should only use one keyword assumption. Correct functionality when placing more then one keyword assumption in a model fragment is not guaranteed. If the *constant* keyword assumption is found in the conditions of the model fragment, then all value- and inequality information present in the consequences is considered constant. These values will not change, neither will the inequalities change, even if their derivatives indicate this. Correct use of the *exogenous* keywords is as follows: A model fragment should have the *exogenous* or a subtype of the *exogenous* keyword assumption present in the conditions. In the former case the input-scenario should supply a specific subtype. The consequences of the model fragment should introduce exactly one quantity. This quantity will take on the specified exogenous behavior pattern. The situation is a bit different with the *generate_all_values* function. Again the keyword assumption should be present in the conditions, but the quantity in question (again; exactly one) should also be present in the conditions. The quantity should therefore be introduced earlier on by the input-scenario or another model fragment. The engine generates specific model fragments for every value in the quantity space and these model fragments make use of the mechanism present in GARP that can assume different value and inequality conditions, thus generating all values for the quantity.

4.9 Correspondence Primitives

Section 3.4 indicates that extra correspondence types are needed in GARP. As shown below, there are 2 basic correspondence primitives in GARP 1.7.2. To aid model building, correspondence types are present that define mappings for whole quantity spaces at once.

Primitives:

- *Value correspondence*
- *Directed value correspondence*

All types:

- *Value correspondence*
- *Directed value correspondence*
- *Quantity correspondence*
- *Directed quantity correspondence*

The set of basic correspondence primitives is extended in GARP 2.0 by adding the notion of derivative correspondence. This derivative correspondence provides extra possibilities for abstract function description. The semantics of the derivative correspondences are analogous to those of the value correspondences: If two quantities have a pair of corresponding derivative-values[53] and one quantity has this particular derivative-value, then the other quantity must also have the corresponding derivative-value. And vice versa in the undirected case. This results in the four basic primitives shown below.

- *Value correspondence*
- *Directed value correspondence*
- *Derivative correspondence*
- *Directed derivative correspondence*

To aid model building some extra types are defined besides the quantity space mapping already present in GARP 1.7.2. A mirrored quantity space mapping and a full correspondence where both values and derivatives correspond for their full quantity spaces. Mirrored quantity spaces need an equal amount of points and intervals and a point-point and interval-interval mapping should be obtained. Figure 4.15 shows valid and invalid mappings.

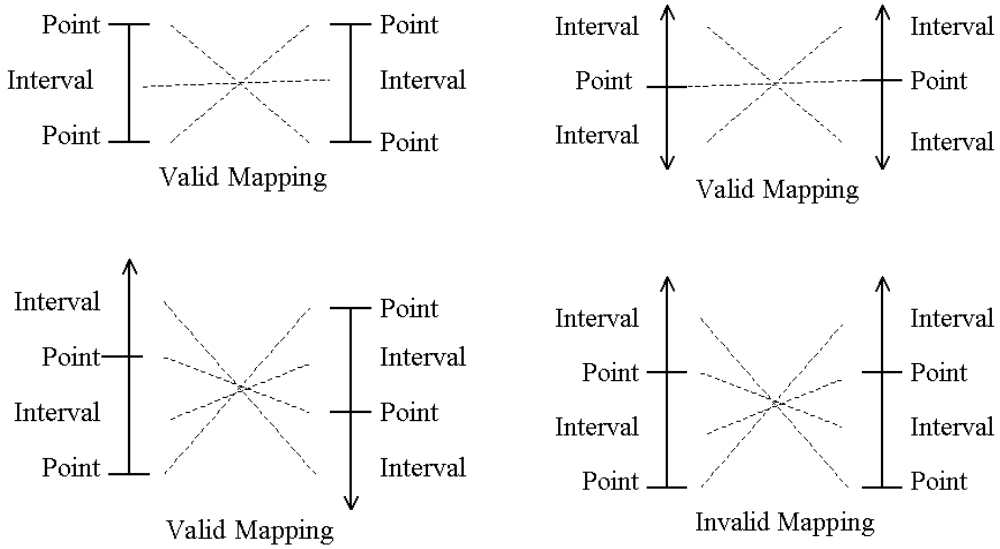


Figure 4.15: Value Mappings

The following set of correspondence types is operational in GARP 2.0:

Values:

- *Value correspondence*
- *Directed value correspondence*
- *Quantity correspondence*
- *Directed quantity correspondence*
- *Mirrored quantity correspondence*
- *Directed mirrored quantity correspondence*

[53] Note that the derivative of a quantity can have a *value* on the derivative quantity space {min, point(zero), plus}.

Derivatives:

- *Derivative correspondence*
- *Directed derivative correspondence*
- *Derivative-quantity correspondence*
- *Directed derivative-quantity correspondence*
- *Mirrored derivative-quantity correspondence*
- *Directed mirrored derivative-quantity correspondence*

Values and Derivatives:

- *Full correspondence*
- *Directed full correspondence*

4.10 Improving Inequality Reasoning

4.10.1 Performance:

The inequality reasoning procedure in GARP 1.7.2 incorporates a technique for reducing complexity named *Analyse Simple Equality*. Recall that pointers are used in the relations in the internal mathematical model[54]. When encountering a simple binary equality $A = B$ the pointer for B is replaced everywhere in the set of relations by the pointer associated with A. Now a number of relations will become redundant. For example, if $A > C$ and $B > C$ were known then after the substitution one relation suffices. Relations of the form $X = X$ are also discarded. This way the amount of relations in the set is reduced and this results in a lesser computational complexity in computing the closure of the mathematical model.

This idea a stretched a little further in GARP 2.0 by adding a technique named *Analyse Zero Equality*. Recall that constraints are placed on the inferences made by the inequality reasoning procedure and that one of those constraints is that no transitivity rules are applied with zero as the intermediate quantity. However, many simple equalities of the form $A = B$, can be derived using zero. Consider the following example:

$X = zero,\ \ Y = zero,\ \ \rightarrow X = Y$

And these simple equalities can be used in an *Analyse Simple Equality* procedure, reducing the number of relations even more. This main idea is used in the analyse zero equality technique which is given below:

Given:

- A set S of relations using a pointer representation, with pointers from $[\ 1, \infty\ )$.
- A mapping M from quantity values, derivatives and landmarks to pointers.

1) Construct the set Z of pointers P that are used in relations in S and are equal to *zero*.
2) For every relation R in S:
   a) Replace the pointers in R that are present in Z by the special purpose pointer *0*.
   b) If R involves an addition, remove any redundant *0* pointers in these additions: $X = Y + 0 \rightarrow X = Y$.
   c) If R has the form $X = X$ then remove R from S.
   d) If R has the form X > X then stop and return contradiction.

[54] See section 2.4 on inequality reasoning.

3) Add the relation *zero = 0* to *S*.

4) Remove any double elements in *S*

5) Replace all pointers in *M* that are present in *Z* by *0*.

6) Return *M* and *S*.

Figure 4.16 illustrates how a set of relations is reduced in size by the Analyse Zero Equality technique. The intermediate result after step 2a in the algorithm is shown. Next, the algorithm will remove double relations*. The relation marked ** involves an addition and will be processed. The relation marked *** will be removed after processing because it is a tautology. Note that if this relation would have had the '>' operator instead of the '=' operator then contradiction would have been derived in step 2d.

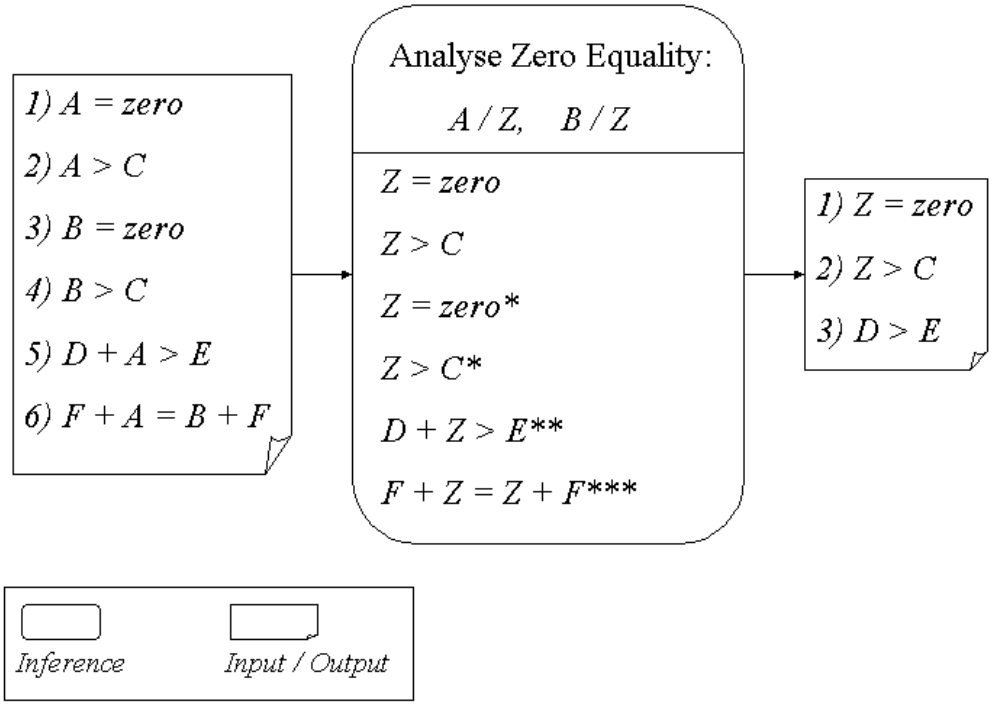


Figure 4.16: Analyse Zero Equality

The *Analyse Zero Equality* technique is applied on the internal mathematical model after computing the full closure of the model. For efficiency reasons it is only applied in those cases where at least one relation of the type *X = zero* is added or newly derived.

One problem arises because of the *Analyse Simple Equality*- and the *Analyse Zero Equality* technique. By substituting the pointers used to represent quantities[55] in relations, the unique quantity / pointer relation is lost. Therefore translation from internal- to normal representation is no longer unambiguous. This translation is needed when providing a trace of the reasoning process. This problem is solved in GARP 2.0 by using variables in the displayed relations and offering a list of possible substitutions open to interpretation by the user. Translation from normal- to internal representation is still deterministic and questions of derivability needed in the reasoning process are not affected.

[55] As in section 2.4 on inequality reasoning the word *quantity* refers to *values*, *derivatives* and *landmarks* in general in this context. When performing internal inequality computations the distinction is irrelevant.

4.10.2 Inference capacity:

The inequality reasoning procedure in GARP 1.7.2 has two lacks in inference capacity. Firstly the relations: $A \leq B$ and $A \geq B$ can be combined to yield $A = B$. This inference is missing because it is not a transitive inference. Simmons (1986) uses graph search to find the most constrained relation between two quantities. Relations used in the quantity lattice of Simmons are $\{\geq, \leq, \neq\}$ and $\{>, =, <\}$, the former set being less constrained. These less constrained relations can be combined in pairs to find the more constrained relation implied. In GARP the less constrained relations are only $\{\geq, \leq\}$, which means there is only one combination of less constrained relations possible. GARP 2.0 has a rule to make this inference. Secondly, algebraic simplification of relations is performed on newly derived relations but not on relations added to the internal mathematical model in GARP 1.7.2. The principle of algebraic simplification is given below:

$$A + C \; rel \; B + C \quad \rightarrow \quad A \; rel \; B \qquad\qquad rel \in \{\geq, \leq, >, =, <\}$$

This can be problematic because sometimes this causes derivable relations not to be found. For example the following relations[56]:

$P + Q + R = X + Y,$
$Q = X,$
$R = Y,$

GARP applies the analyse simple equality technique[57] to this set of equations, yielding:

$P + Q + R = Q + R,$

Using algebraic simplification, the following result is obtained:

$P = zero,$

Without algebraic simplification this result is not found. Therefore GARP 2.0 does apply algebraic simplification on relations when they are added to the internal mathematical model.

4.11 Quantity Space Constraints

If the quantity space of a quantity has a landmark as its extreme value the quantity should not be allowed to keep on moving upon reaching this landmark. Take for example the leaking bucket system mentioned in sections 2.2, 3.1 and 4.2. The top extreme value of the quantity *water level* is the landmark *max* in this model. Now if it reaches this maximum value the *water level* cannot increase anymore so its derivative must be *zero* or *min*. Given the discussion of epsilon ordering in section 4.5.3 this is all the more true: If a quantity is in a point and moving it should immediately change its value to the next interval, yet there is no next interval in this case so this transition can never occur. Consequently this situation is illegal.

In GARP 1.7.2 this functionality was already implemented, but due to a bug the generated constraints did not work but for the value *zero*. In GARP 2.0 this bug is fixed and quantity space constraints are active in quantity spaces with landmarks at extreme values. An example

[56] This example is drawn from the 'balance equation' used in influence resolution calculus described in section 4.2. It corresponds to the case where a quantity has four influences, two negative, two positive, which are in balance.
[57] See previous section.

of how this influences model construction is the construction of a model of an overflowing U-tube. A model is presented in section 5.2 and this issue is discussed further in that section. For compatibility and modeling flexibility an algorithm option switch[58] is present to disable the quantity space constraints.

The next section of this thesis presents an evaluation of GARP 2.0 and the solutions described in the previous sections.

[58] See section 4.1 on flexibility.

5. Evaluation

To evaluate GARP 2.0 two ecological models and some different models of the U-tube system were used. Simulations were judged on completeness and soundness of the simulated behavior given the model used. Main goal has been to test GARP 2.0 and illustrate its functionality therefore these models do not intend to be perfect representations of the described phenomena. Performance tests were conducted measuring the total time spent on computation and the number of dead-end transitions generated.

5.1 Ecology: Single Population Dynamics:

The behavior of a single population plays an important role in the Cerrado Succession Hypothesis model (Salles & Bredeweg, 1997; Salles & Bredeweg, 2003). A very simple model of a single population is presented first to introduce the subject and outline some of the problems in this domain. Figure 5.1 shows the quantities and basic dependencies in the model. The *number of individuals* in the population represents the population size and has the quantity space {point(zero), low, point(normal), high, point(max)}. The *number of individuals* is influenced by four quantities: the *born, dead, emigrated* and *immigrated* individuals. These quantities have the simple quantity space {point(zero), plus}. *Immigrated* is modeled as an increasing exogenous quantity and has the value *plus*. In this model the following relation between *Number_of* and the other quantities is assumed: When it is higher than *zero*, *Born*, *Dead* and *Emigrated* all have the value *plus*. When *Number_of* is *zero*, all are *zero* as well. Changes in *Number_of* propagate to these three quantities.

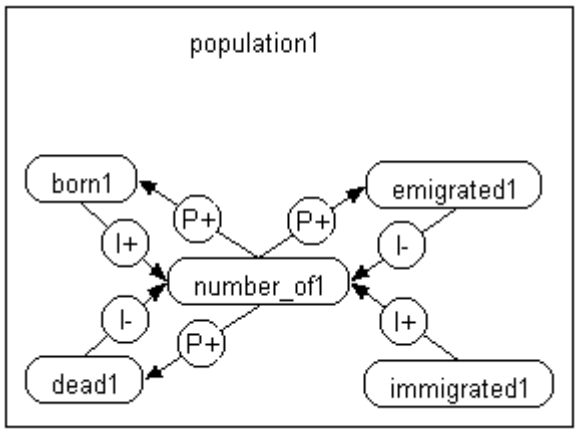


Figure 5.1: Single Population Dependencies

A state diagram[59] and value diagram of this simulation are shown in figures 5.2 and 5.3 respectively.

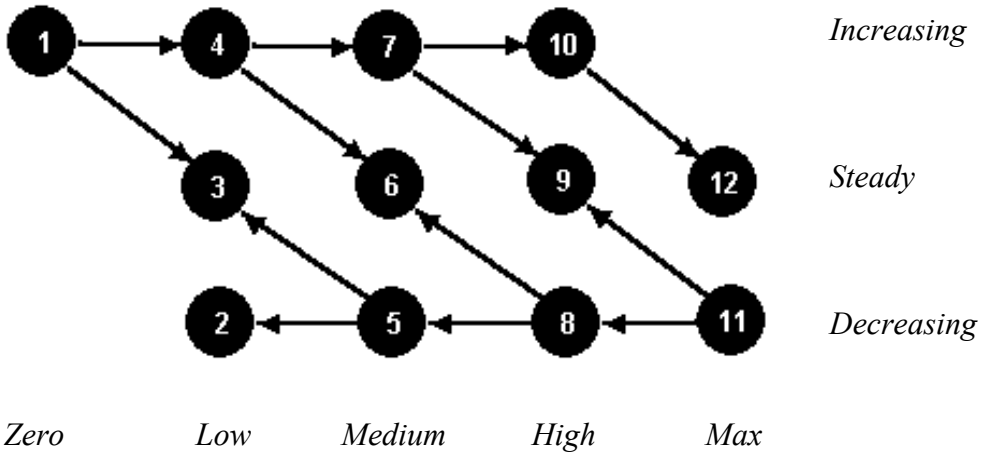


Figure 5.2: State Diagram Single Population

First observation is that all possible states are found: With *Immigration* fixed at value *plus*, the 'increasing' state 1 is the only possibility when *Number_of* and all other quantities are *zero*. In the value *max* a state with increasing *Number_of* is impossible because this is the uppermost point in its quantity space[60]. All other value and derivative combinations are present for *Number_of*. Because other quantity values and derivatives are deterministically defined in this model by those of *Number_of* all possibilities are covered.

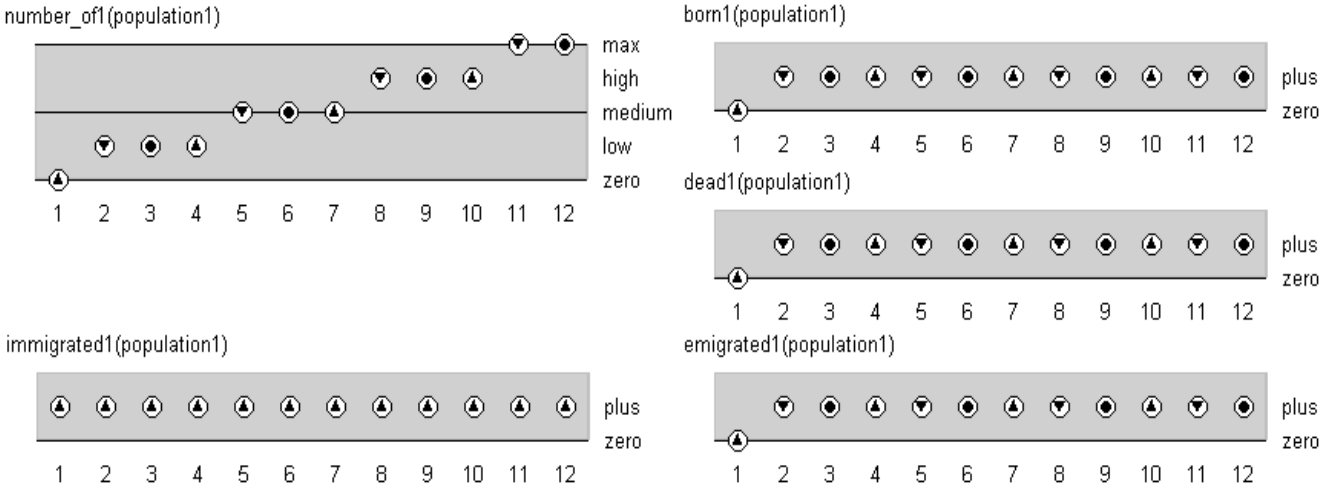


Figure 5.3: Value Diagram Single Population

[59] In all following state diagrams on single population dynamics states are ordered in two dimensions: Horizontally from left to right the value of Number_of starting with *zero* and ending with *max*. Vertically from below to above the derivative of Number_of starting with *min/decreasing* and ending with *plus/increasing*. Given the assumptions described above Number_of determines all other values. For example, in state 3 Number_of is *low* and *steady*. This means Born, Dead and Emigrated are *plus* and *steady*. Exception is Immigrated which is exogenous and defined to be *plus* and *increasing* always. This ordering of states allows an easy interpretation of the diagram without much need to refer to the value diagram.

[60] See section 4.11 on quantity space constraints.

The second observation is that the transitions follow a very regular pattern. They are based on very simple behaviors: *Number_of* changes its value because it is increasing or decreasing. It can either continue to move or stop in the next state. All transitions are shown in figure 5.4. Note that the transitions from state 1 also include a value change for the other quantities and that the transition from state 2 from *low* to *zero* does not succeed. The only valid state with *Number_of* at *zero* is state 1 and continuity does not allow this jump from 'decreasing to 'increasing'.

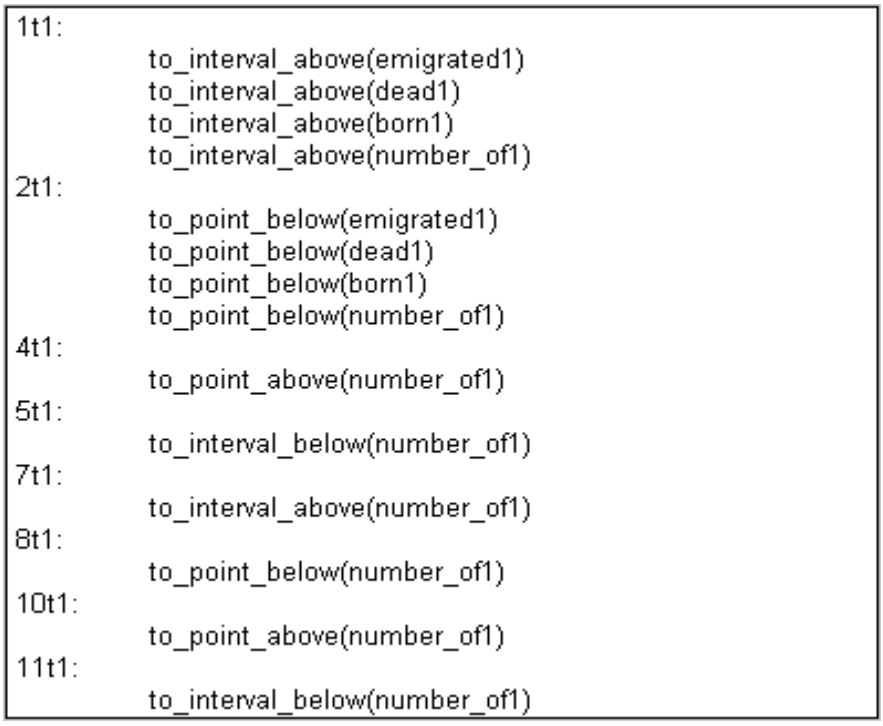


Figure 5.4: Transitions Single Population

Note that no 'vertical' transitions (only a change in derivatives) are found. This is correct, in GARP a change of a derivative always coincides with some other behavior. Inferring derivative behavior separately is impossible for second order derivative information is not present. Also no transitions exist from the 'equilibrium' states 3, 6, 9 and 12. Again this is technically correct, all quantities are *steady* except for *Immigration*, which is *plus* but has no higher value to go to. However, these states are intuitively and semantically inconsistent. How can the situation be steady if one influencing flow is growing and none is decreasing to compensate? The next extended model suggests a solution to this problem[61]. This model makes use of the more elaborate reasoning capabilities of GARP 2.0.

In this extended model, summations of the positive and negative influences, *Inflow* and *Outflow* respectively, are defined. An inequality between *Inflow* and *Outflow* describes the relation between all influencing quantities. The problematic situation described above is now interpreted as follows: It describes the momentary situation where *Inflow* becomes equal to *Outflow* while increasing. Because it keeps increasing, this situation should immediately transform to the situation where *Inflow* is larger than *Outflow*. This behavior is captured in

[61] Note that this situation where immigration is fixed at plus and increasing is quite artificial. However, the same problematic pattern of an impossible equilibrium state occurs when combining more populations that influence each others born and death rate. (E.g. a predator-prey situation)

this model because the equality between *Inflow* and *Outflow* changes to inequality. And this change to inequality is immediate and takes precedence over other changes[62]. Figure 5.5 shows the dependencies, the values and derivatives in state 6 of the resulting simulation. Note how the value and derivative of *Inflow* and *Outflow* are defined.

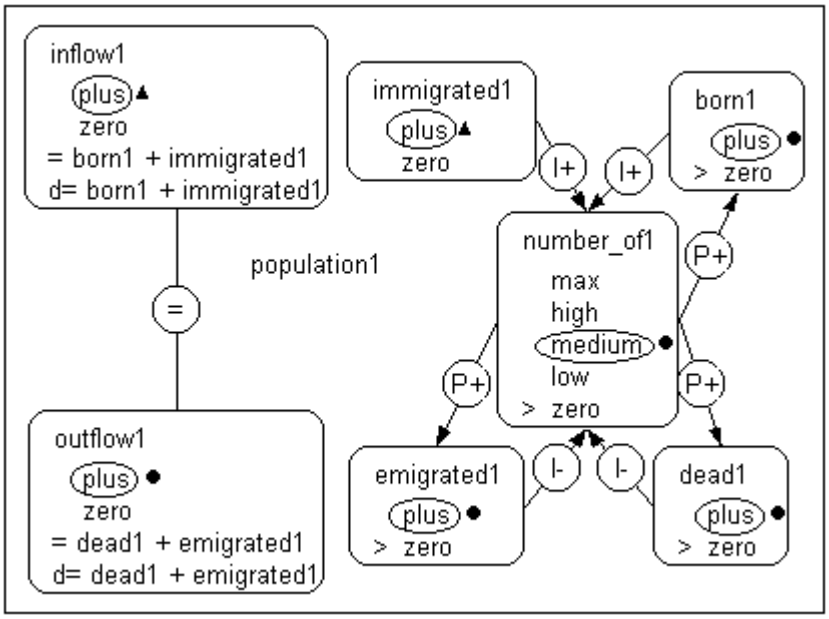


Figure 5.5: Extended Single Population Dependencies For State 6

A full state diagram of this extended model is shown in figure 5.6. As can be seen the states are the same as in the previous model (figure 5.2), yet the pattern of transitions is completely different. Intuitively, some transitions are missing in this simulation. However, transitions will be discussed below and shown to be correct. A value diagram for the most important quantities[63] is shown in figure 5.7. The derivative of *Inflow* is undetermined in states 9 to 12. This is correct since it is the summation of *immigration* (derivative *plus*) and *born* (derivative *min*).

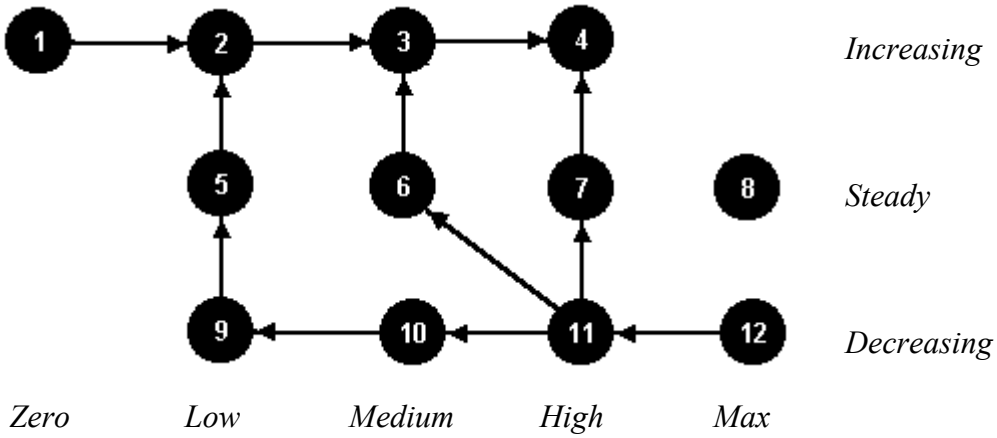


Figure 5.6: State Diagram Extended Single Population

[62] See section 4.5.3 on epsilon ordering.
[63] All other quantity values and derivatives can be inferred quite easily by the reader given the previous model.

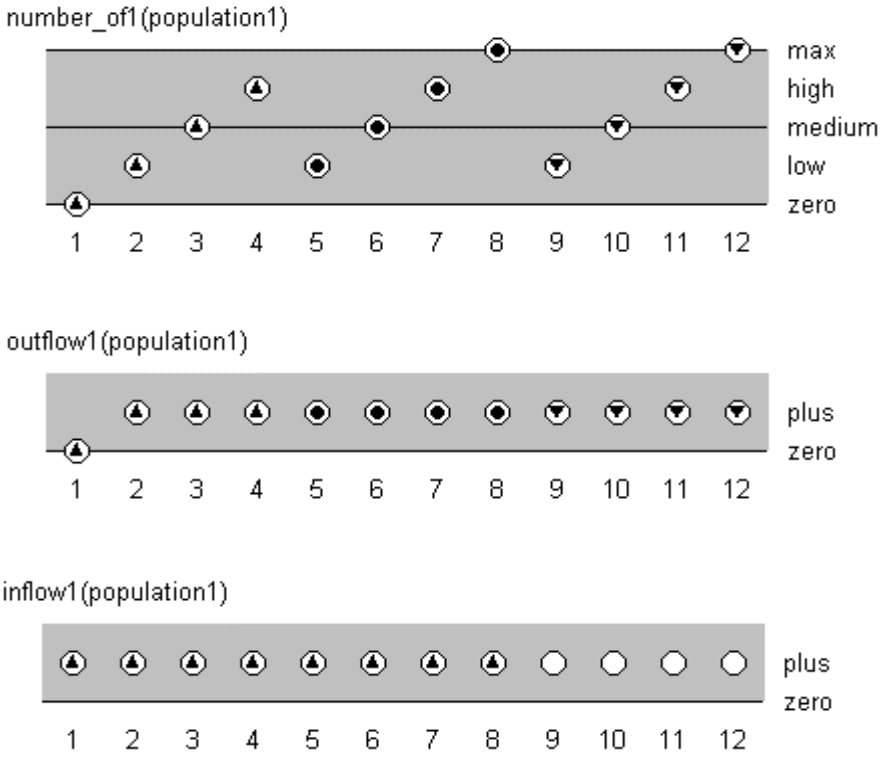


Figure 5.7: Value Diagram Extended Single Population

As this extended model was made to deal with the problem of the 'illegal' equilibrium states (states 3, 6, 9 and 12 in figure 5.2), lets first examine transitions for these states in this extendended model (states 5, 6, 7 and 8 in figure 5.6). All these states generate the immediate transition where *Inflow* becomes greater than *Outflow*. As an example the transition for state 7 is shown in figure 5.8. In state 8 this transition does not succeed because the target state, *Number_of* is *max* and increasing, is invalid. Note that the derivatives of *Inflow* and *Outflow* indicate all these transitions and that the derivative of *Number_of* is *zero* in these states, which means it cannot change value so no other transitions are possible.

```
7t1:
        from_equal_to_greater(inflow1, outflow1)
11t1:
        to_point_below(number_of1)
11t2:
        to_point_below(number_of1)
        from_smaller_to_equal(inflow1, outflow1)
11t3:
        from_smaller_to_equal(inflow1, outflow1)
10t1:
        to_interval_below(number_of1)
```

Figure 5.8: Some Transitions Extended Single Population

Now for the transitions of the other states: vertical transitions correspond to a change in the inequality between *Inflow* and *Outflow* and horizontal transitions correspond to a change in the value of *Number_of*. Note that the inequality changes are not commanded by the derivatives of *Inflow* and *Outflow*, but they are generated using the behavior assumption mechanism described in section 4.4.3. The transitions for state 11 illustrate the possibilities all

these other states have: move vertically, horizontally or both. States 1, 3, 10 and 12 show no vertical transitions because the value transitions (*to interval above/below*) take precedence over the inequality transitions (*from smaller/greater to equal*)[64]. For states 1, 2, 3 and 4 the vertical transitions are illegal: To generate the change of the inequality between *Inflow* and *Outflow* it has to be assumed that $\delta Outflow > \delta Inflow$, while in the target states it is known that $\delta Outflow < \delta Inflow$. A transition here would represent discontinuous behavior. State 9 cannot make a transition to a state where *Number_of* is *zero*, because only state 1 is valid in this category and a transition to this state requires a discontinuous jump for both the *Inflow-Outflow* inequality and many derivatives. Lastly state 4 cannot make the transition to the *max-increasing* state because this state is illegal and therefore not present.

Why did the first model generate transitions illegal in the extended model? Most important is that the influences were ambiguous in this model, the underlying relation between positive and negative influences being unknown[65] and therefore free to change discontinuously. This leaves us to conclude that the state diagram is theoretically correct. As noted in the discussion of the previous model, the modeled situation is quite artificial but it does succeed in illustrating important principles and should be judged accordingly. It shows that the problematic states of figure 5.2 represent valid behavior. They provide a path from decreasing to increasing states in the extended simulation (figure 5.6) and vice versa in less artificial simulations. In older models a constraint was placed on derivatives[66], completely ruling out these states. The approach taken in this extended model is only possible in GARP 2.0, because the inequality between Inflow and Outflow provides enough information to focus the influence resolution. To complete this new solution momentary states should be clearly discernable from states having temporal extent. A graphical rendition of this distinction could for example make use of circles and squares as done by (De Kleer & Brown, 1984).

To show how the determination of valid behavior combinations was improved in GARP 2.0 the transition process for state 11 is examined. Firstly the single changes or terminations are determined. They are shown in figure 5.9. The first five terminations do not represent valid behavior as long as Number_of is greater then zero. In the Cerrado Succession Hypothesis model, using GARP1.7.2, domain specific ordering rules were needed to remove such terminations and avoid combinatorial explosion.

```
11t1:
        to_point_below(outflow1)
11t2:
        to_point_below(inflow1)
11t3:
        to_point_below(emigrated1)
11t4:
        to_point_below(dead1)
11t5:
        to_point_below(born1)
11t6:
        to_point_below(number_of1)
11t7:
        from_smaller_to_equal(inflow1, outflow1)
```

Figure 5.9: Changes For State 11

[64] See section 4.5.3 on epsilon ordering.
[65] As noted in section 4.2.1 an inequality discribing this relation could be generated automatically once a mechanism is present for determining behavior of non-binairy inequalities.
[66] dNumber_of = (dBorn + dImmigrated) – (dDead + dEmigrated)

GARP 2.0 uses domain independent concepts to achieve this goal. Firstly: *Emigrated*, *Born* and *Dead* correspond to *Number_of* in the value *zero*. *Number_of* is not in *zero* in this state, nor can it reach zero with any of the terminations present. Therefore changes *11t3*, *11t4* and *11t5* are removed. A partial trace of the reasoning process involved is given:

```
to_point_below(emigrated1) may not occur because of activating
correspondence, corresponding value not present and no
corresponding termination.
removing:
    to_point_below(emigrated1)
```

Secondly, *Inflow* and *Outflow* are summations of *Born* and *Immigrated*, *Dead* and *Emigrated* respectively. Since these all remain plus, *Inflow* and *Outflow* should also remain plus. Again a partial trace of the reasoning process is given:

```
determining combination constraints for context:
    [inflow1, born1, immigrated1]
    current values:
    [plus, plus, plus]
    terminations:
    [[to_point_below(inflow1)]]
    constant:
    [equal(inflow1, plus(born1, immigrated1))]
testing combination:
    to_point_below(inflow1)
    - inconsistent combination.
```

The complete process results in only the three transitions for state 11. These are shown in figure 5.8.

Lastly figure 5.10 shows the simulation of the extended single population model with Immigration at the values *zero* and *plus* and again modeled as exogenous and increasing. The upper half of the diagram contains all states where *Immigration* is *plus*, the lower half contains all states where *Immigration* is *zero*. States in each half are again ordered on the value and derivative of *Number_of*. Since *Immigration* is increasing it can change from *zero* to *plus* at any point. Therefore transitions from the lower half to the upper half are omnipresent and often take precedence over other possible transitions. The upper half of the diagram contains the same structure as figure 5.6. Although a simple model is used as well as a simple exogenous behavior pattern, the resulting simulation is quite complex. This shows that the use of exogenous behavior patterns easily results in state graphs, which are correct, but very large and hard to interpret.

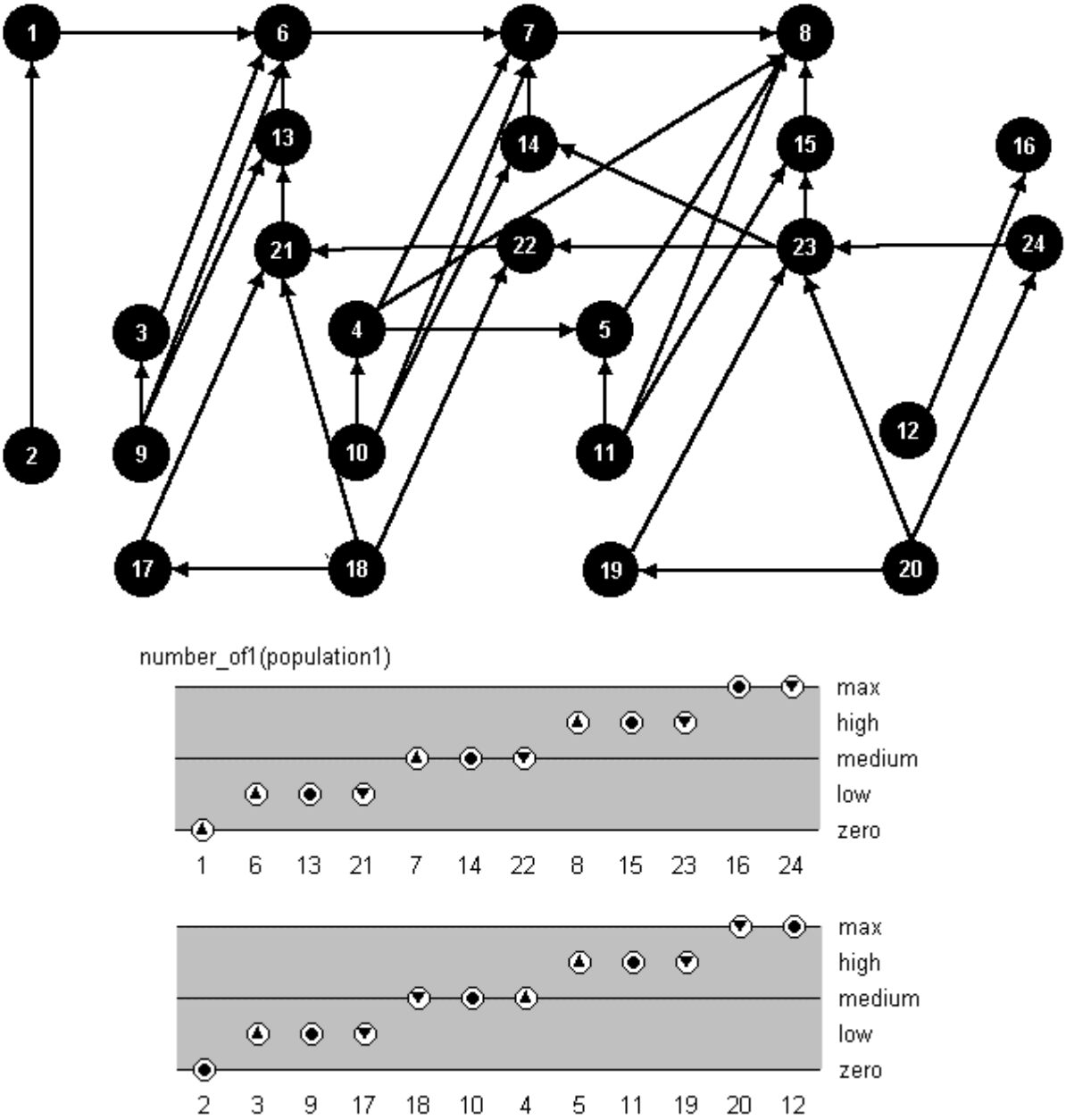


Figure 5.10: Extended Single Population With Increasing Immigration.

5.2 Physics: U-Tube modeled in Process Style:

The U-tube, or two-tank system is often modeled in Qualitative Reasoning (Weld & De Kleer, 1990). A process-centered approach (Forbus, 1984) is followed in the model presented here. Processes play the central role in a simulation of behavior by causing changes in the system. In the original Qualitative Process Theory quantities associated with processes are introduced when the process becomes active and are removed when the process stops. In GARP 1.7.2 quantities always remain in the simulation once introduced. In GARP 2.0, as an experiment, an option is built in that removes quantities associated with non-existing processes[67]. An overview of the u-tube simulation is as follows: Two containers are filled with a liquid and

[67] Implementation of this mechanism is straightforward. Any quantities that are not present in an active model fragment at the end of the state specification are removed. Application of this mechanism is optional, by default it is switched off (See section 4.1 on flexibility).

connected at the bottom by a pipe. Unequal liquid levels in the containers will result in unequal pressures at the bottoms of the containers. These unequal pressures trigger a liquid flow (a process) that influences the amounts of liquid in both containers. When the levels and pressures become equal the flow-process stops and equilibrium is reached. If the heights of the containers are unequal it is possible that the water level reaches the maximum of the lower container and it overflows.

The first question is how to model the overflow process. What situation triggers this process? Overflow should be modeled for a tank independently of the rest of the system it is connected to. This means the existence of the *flow* in the pipe cannot be a trigger for an overflow situation. A suitable quantity space for the *level*, *amount* and *pressure* in each container is {point(zero), plus, point(max)}. In GARP 1.7.2 the quantity space imposed no constraints on derivatives[68]., therefore overflow could be modeled in such a way that it occurred when the *level* in a container was *max* and rising. In GARP 2.0 this situation is illegal and therefore this is approach impossible. Intuitively, overflow occurs when the water level is above the rim of the container. The model presented here follows this intuition. The quantity space used is: {point(zero), plus, point(rim), bubble}. The uppermost value in this quantity space represents the amount of water, which is just above the rim of the container. A situation that is possible because of the viscosity of liquids. The assumption is that overflow occurs whenever the water level is above the rim. Figure 5.11 illustrates the dependencies[69] in the situation where the right container is overflowing.

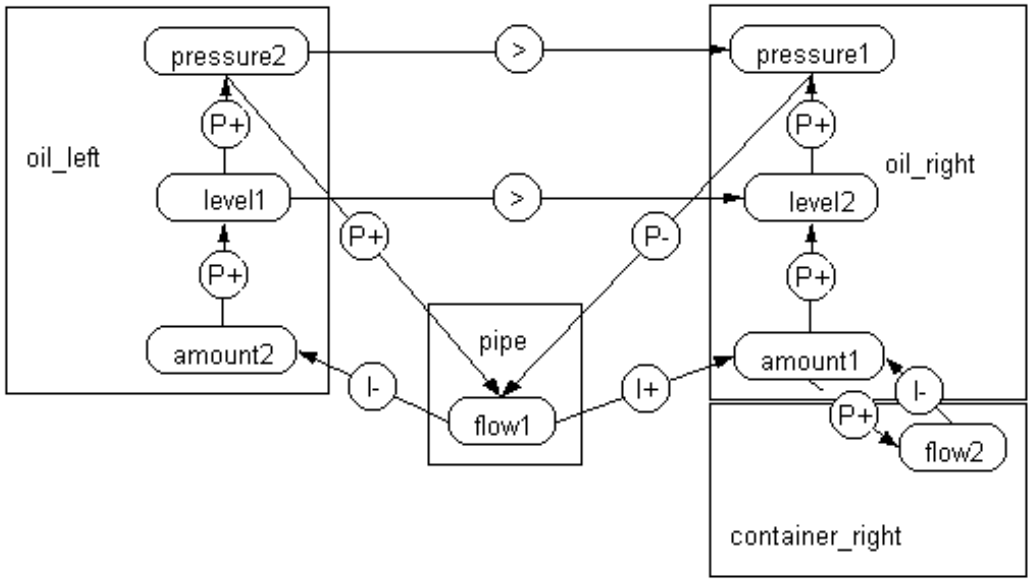


Figure 5.11: Dependencies Overflowing U-Tube.

An inequality between the two flows can be present describing if the overflow is just starting (<), flowing steady (=), or ending (>). In the latter case the water level will go down to the rim when equilibrium is reached. These inequalities can change and provide the behavior from a starting overflow to an ending overflow. As a simplifying assumption, the case where the flow in the pipe is larger then the flow out of the container (overflow) is not included in this

[68] See section 4.11 on quantity space constraints.
[69] Quantity correspondences are also present between *amount* and *level* and between *level* and *pressure* in both containers, but these are not shown here.

model. Note that without the influence resolution procedure[70] present in GARP 2.0 this modeling approach could not have been taken. The simulation for two half-filled tanks, with unequal levels is shown in figure 5.12. Three behaviors are possible. Firstly, the equilibrium can be reached with both levels under the rim (path 1-4). Both levels move towards eachother and become equal. Secondly, the equilibrium can be reached with one level exactly at the rim (path 1-3). The levels become equal at the same time the right level reaches the rim of the container. Note that this means the right container is lower than the right container. Thirdly, the equilibrium can be reached after one container overflows (path 1-2-5-6-3). In this case the right level reaches the rim of the container before the levels become equal. Again the right container is lower than the right container. The levels keep moving and the right liquid forms a bubble above the rim of the container and overflows. First the overflow and the flow in the pipe are equal resulting in a steady overflow. Later, the overflow is greater than the the flow in the pipe for a moment when the right level drops back to the rim and equilibrium is reached. The active model fragments in each state are shown in table 5.1.

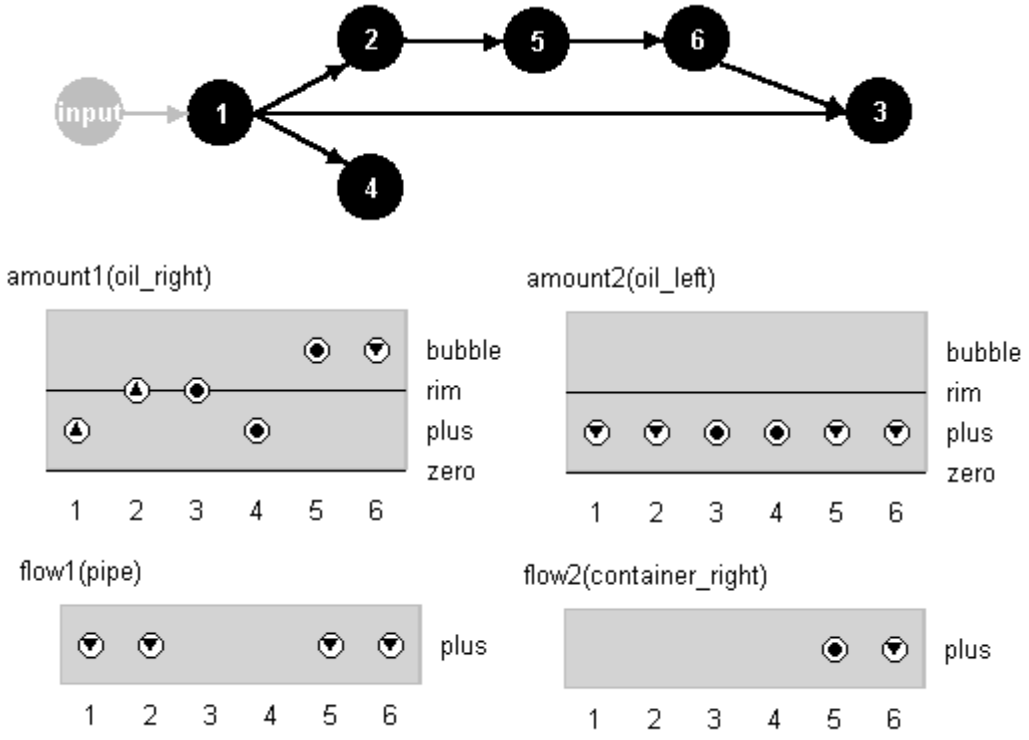


Figure 5.12: State Diagram & Value Diagram U-Tube

As can be seen, the quantities *flow 1* and *flow 2* representing the flow in the pipe and the overflow respectively are introduced when the processes are active and removed when the processes are inactive. Their quantity space is simply {plus} for this reason[71]. One advantage of this approach is that in the equilibrium situation the flow does not have to be modeled. In a model where quantities are not removed this has to be done because otherwise they can cause unwanted branching since they are unconstrained. For example, the liquid flow model fragment could define a *flow* with quantity space {point(zero), plus} and the constraint *flow* = *pressure-L – pressure-R*. If the *pressures* would become equal, the *flow* should become *zero*. But if the pressures would be equal there would be no more liquid flow process and therefore

[70] See section 4.2 on influence resolution

[71] This quantity space has no landmarks so the sign of this flow is undetermined. The inequality Flow > Zero is present in this model to solve this problem.

this inequality wouldn't be in the model anymore. Now two situations, one where *flow* is *zero* and one where *flow* is still *plus*, would be valid! Since in our model the *flow* quantity is removed in the equilibrium states there is no need to model inactive processes. The closed world assumption[72] is applied in the equilibrium states to determine that the *amounts* are steady in case no process is influencing them.

Model Fragment:	***Active in states:***
Contained Liquid	all (2x)
Liquid Flow	1, 2, 5, 6
OverFlow	5, 6
Two Flows	5, 6
Two Flows: Equal	5
Two Flows: Overflow Greater	6

Table 5.1: Model Fragments U-Tube

The transitions in this simulation are all correct, but as will be shown unnecessary dead end transitions are generated. Transitions for states 2 and 5 are shown in figure 5.13, no other transitions are generated for these states.

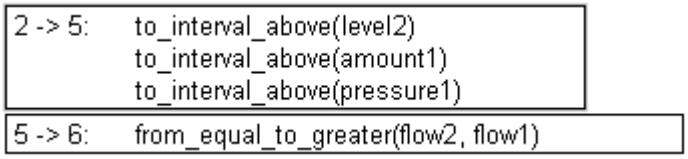


Figure 5.13: U-Tube Transitions For States 2 & 5

For states 1 and 6 more transitions are generated. The transitions for state 1 will be discussed here and are shown in figure 5.14. Transition *1t1* leads to state 2. Transitions *1t2* and *1t3* both lead to state 3. Clearly transition *1t3* is the correct one of these two. Transition *1t2* succeeds because the inequality between the pressures was only assumed in state 1 and this assumption was dropped over the transition. Note that this is a mechanism by which means an inequality can change without an explicit termination. The equivalent transition for the other inequality, *1t4*, does not succeed because this termination was given in the input scenario and therefore it can only change through a termination. The fact that transition *1t2* wrongly succeeds is not problematic because it leads to the same state as transition 1t3. Therefore it does not lead to a different or incorrect simulation. The same pattern holds for transitions *1t5*, *1t6* and *1t7*, of which the first two lead to state 4.

[72] See section 4.2.2 on the closed world assumption.

```
1t1:
        to_point_above(pressure1)
        to_point_above(amount1)
        to_point_above(level2)
1t2:
        to_point_above(pressure1)
        to_point_above(amount1)
        to_point_above(level2)
        from_greater_to_equal(level1, level2)
1t3:
        to_point_above(pressure1)
        to_point_above(amount1)
        to_point_above(level2)
        from_greater_to_equal(level1, level2)
        from_greater_to_equal(pressure2, pressure1)
1t4:
        to_point_above(pressure1)
        to_point_above(amount1)
        to_point_above(level2)
        from_greater_to_equal(pressure2, pressure1)
1t5:
        from_greater_to_equal(level1, level2)
1t6:
        from_greater_to_equal(level1, level2)
        from_greater_to_equal(pressure2, pressure1)
1t7:
        from_greater_to_equal(pressure2, pressure1)
```

Figure 5.14: U-Tube Transitions For State 1

Apart from the fact that transitions *1t2* and *1t5* should not succeed, one can wonder why the ordering procedure[73] does not filter these impossible changes. This is because it requires considering a quantity context of four quantities that are connected by two changing inequalities and constant inequalities. The dependencies of this quantity context are shown in figure 5.15. There is no dependency incorporating all four quantities and therefore this context can only be generated by combining contexts. Such a mechanism is not present in GARP 2.0.

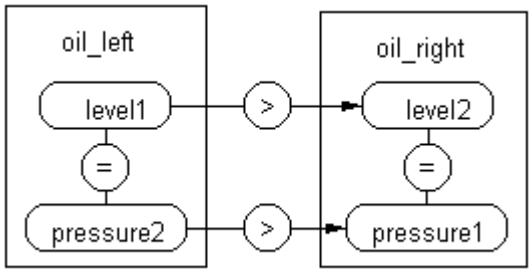


Figure 5.15: Combined Quantity Context

An interesting point can be made about the changing relation between the *overflow* and the *flow* in the pipe. As shown, this inequality changes from equal to greater from state 5 to 6. And in the next state (3) equilibrium is reached. Equilibrium suggests however that both *flows* are *zero* and therefore equal. The transition to state 3 should therefore include the change of

[73] See section 4.5.2 on application of ordering principles.

this inequality from greater to equal. But continuity[74] does not allow this transition: The first transition from greater to equal assumes *δflow1* > *δflow2* and therefore in the next state *δflow1* < *δflow2* cannot be assumed. This is sort of a limit problem and second order derivative information would be needed to solve it. An intermediate state and transition could model the change in the relation between the derivatives. By removing the *flow* quantities this problem is bypassed in this model. Problems such as these are common in Qualitative Reasoning and continuity is a debated subject (Forbus, 1988).

5.3 Physics: U-Tube modeled in Component Style:

This second model of the U-tube system is made using the component-centered approach (De Kleer & Brown, 1984). The components of the U-tube are modeled in all their possible states: A container is empty, half-full or full, emptying, steady or filling. A pipe has no flow, a left-to-right flow or a right-to-left flow. Equalities between values, derivatives and landmarks of the bottom pressures in the containers and the left- and right side pressures in the pipe respectively represent the connections between the components. Constraints are given for the system as a whole. These specify the total amount of liquid in the system, which should remain constant for valid behavior. The model fragment describing a container is shown as an example in figure 5.16. The Container instance and Constant assumption are conditional in this model fragment. All other elements are consequences[75].

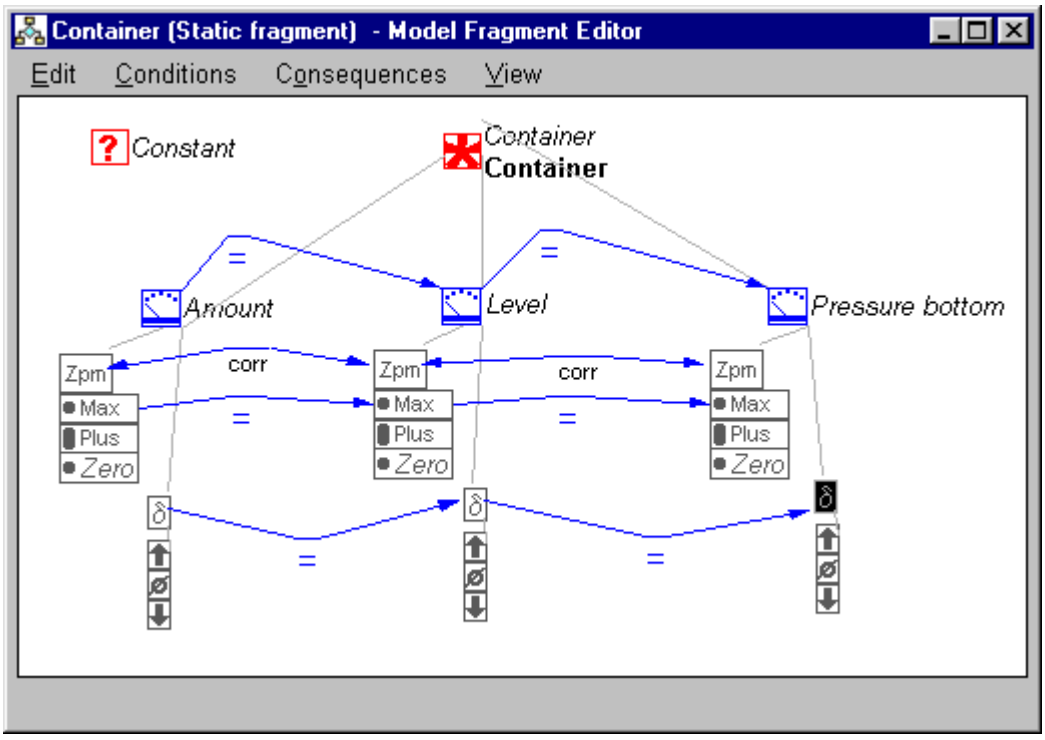


Figure 5.16: Homer Screenshot Of Container Model Fragment

This model allows the simulator to follow the computational order used in the component-centered approach. Firstly all valid states are computed directly from the input-scenario and secondly all transitions are sought. The resulting simulation is shown in figure 5.17. The input-scenario used here specifies that the containers are of equal height.

[74] See section 4.3.3 on continuity ov er transitions.
[75] Correspondences are not native to the componenent-centered approach but are added for convenience.

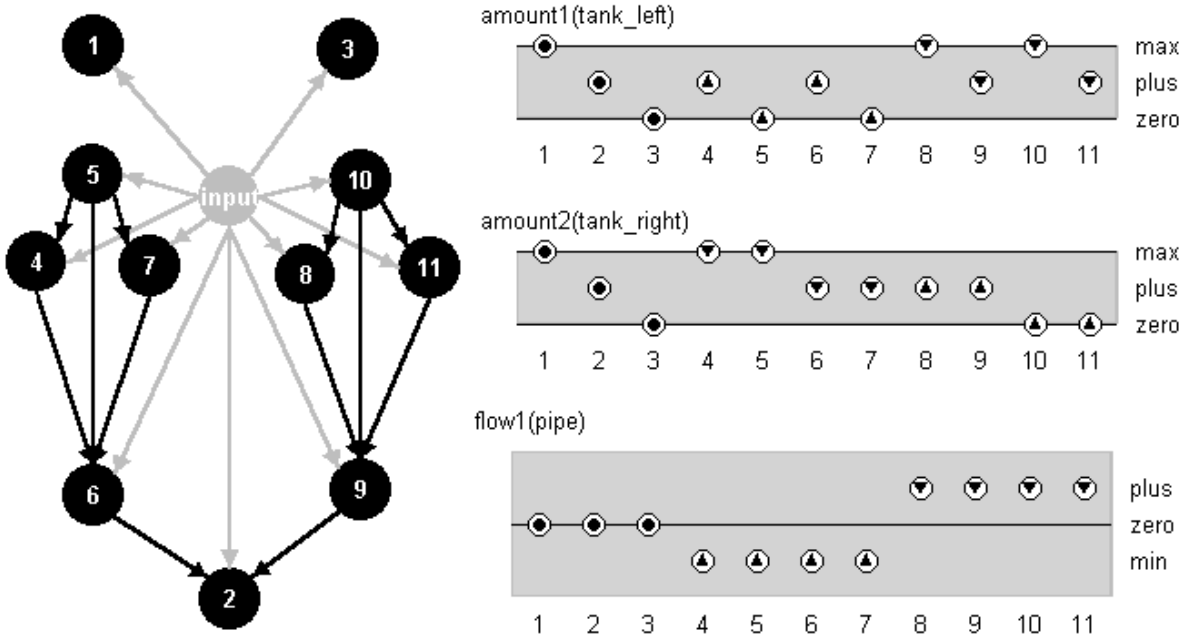


Figure 5.17: Component-Centered U-Tube Simulation

As can be seen all possible states of the specified system are found. In state 1 both containers are full and levels are therefore equal. In state 3 both containers are empty and of course again there is no flow. The other states model a partially filled u-tube either with the left side higher, the right side higher or equal levels. Because a symmetrical u-tube is modeled no steady state is possible with only one side at the maximum or zero point, neither is an overflow situation possible. There is one problem in the state descriptions generated however: Figure 5.18 shows the values of the *total amount of liquid* present in the u-tube. This quantity is defined as the sum of the amounts in the left and right container. The *max* point is defined as the sum of the *max* points of the containers. Note that no value is derived in states 1, 6 and 9. However these should be *max, plus and plus,* respectively.

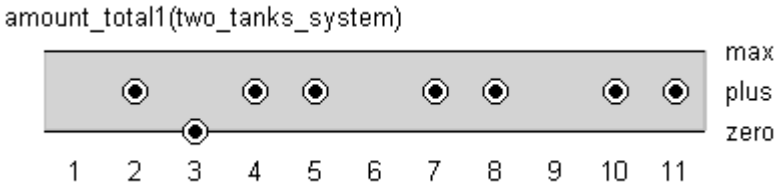


Figure 5.18: Total Amount Of Liquid In U-Tube

The trace of the internal mathematical model of state 1 shows, amongst others, the following relations and mappings:

```
Derivable relations:
  X21 >= X22
  X22 = X1 + X15
  X1 + X15 = X21
```

```
-- substitutions:
  value(amount2)/X1
  value(max(amount2))/X1
  value(amount1)/X15
  value(max(amount1))/X15
  value(max(amount_total1))/X21
  value(amount_total1)/X22
```

Obviously the stronger relation X21 = X22 can be derived that would say the *total amount* is in its maximum. This relation is not found because of the strict regime placed on the inequality reasoning procedure. As mentioned in section 2.4 constraints are placed on the combination of derived and parent relations to avoid combinatorial explosion. In general these constraints allow all relevant relations to be derived. This example shows that in some cases these constraints are too strict. Simply lessening constraints is not a very suitable solution because this results in a very serious increase in time spent on inequality reasoning.

Transitions found in this simulation are almost all correct, none are missing. States 1 and 3 are separate from the rest because they model a system with a different *total amount* of liquid. Some examples of the transitions occurring are shown in figure 5.19:

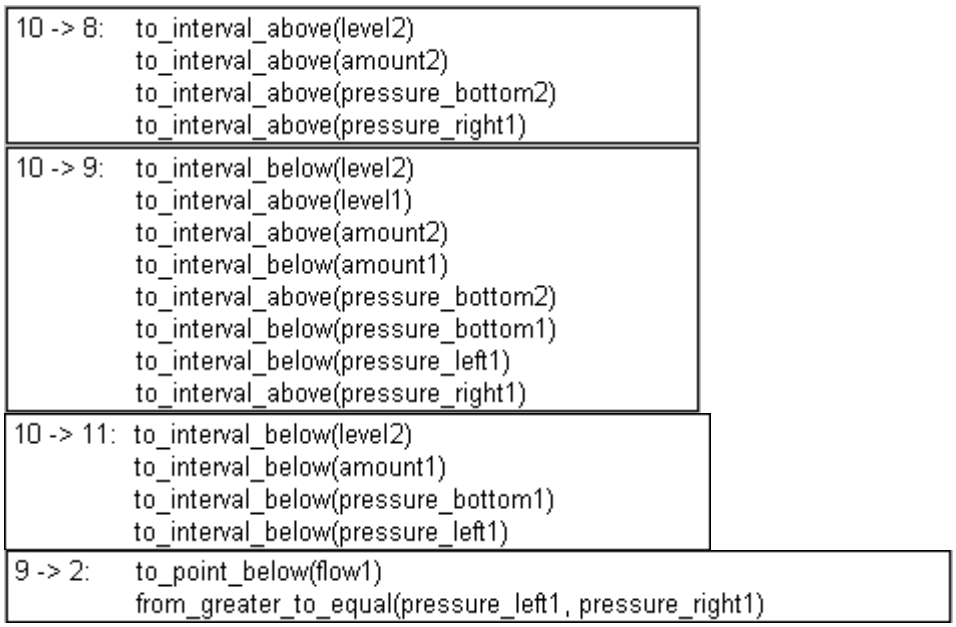


Figure 5.19: Transitions In Component Centered U-Tube

These patterns of transition are equal in the other half of the simulation. Note that the transition from state 10 to state 9 is the compound transition of those to states 8 and 11. These last two transitions are impossible in reality, they imply an in- and decrease of the *total amount* respectively. At the moment the full container leaves its maximum, the other must not be empty anymore. Therefore the transition to state 9 is the only valid one. The reason GARP 2.0 allows these is that the *total amount of liquid* is in the interval *plus* in all four states and therefore it does not seem to move illegally. Two possible solutions come to mind. Firstly it is derivable in state 10 that the *total amount of liquid* is equal to the maximum point of the left container. In state 8 and 11 it is greater and smaller than this point respectively. These relations are not used now in the transition procedure. Another possible observation lies in the observation that all three transitions of the small epsilon class[76] and one is the compound version of the other two. Mathematically speaking these two transitions take place immediately, state 10 has no temporal extent. Therefore: How can one happen before the other? They should all happen together, the compound transition is the only possibility. Both these possible solutions will have to be researched in future work.

[76] See section 4.5.3 on epsilon ordering.

Some issues that came up during model construction are noteworthy. The constraints representing the connections between components include some inequalities on derivatives. In an earlier version of this model these connections were specified in the input scenario. But these inequalities fall under the continuity regime[77] in GARP 2.0. To keep these inequalities active in the simulation they were placed in a model fragment in this version of the model. Also specified in a model fragment was the derivative of the *total amount of liquid*. This is defined *zero* in the input-scenario, but is subject to the continuity regime as well. If not defined *zero* in a model fragment, new states would have an unknown derivative of the *total amount of liquid*. These states would not match the original states with a zero derivative for this quantity[78]. Constant labels[79] were used for the Container, Pipe, Constraints and Connection model fragments and proved very useful in determining valid transitions combinations as will be shown in section 5.4. In the earlier version of this model correspondences were added for this purpose (fig 5.16) , yet they are not native to the component-centered approach. Last but not least a repair was needed in the reasoning engine. This model uses the mechanism present in GARP to assume conditional inequalities in model fragments. In states 6 and 9 (and others) it has assumed the derivatives of the *amount* of liquid present in the Filling and Emptying model fragments. In the transition procedure active model fragments are reassessed on the validity of their conditions. Any assumptions made in the original state are made again if they are not contradictory with known transition results. In this case the Filling and Emptying model fragments are applied again in the transitions to state 2. This is clearly wrong, but not contradictory because these model fragments concern derivatives and the transition results specify no specific constraints on derivatives. Only continuity rules are applied. The repair consist of not allowing any assumptions on derivatives at this point, for these can never be contradicted by transition results.

5.4 Performance Measurements:

While the previous sections have emphasized semantic validity of the simulations this section addresses computational efficiency and performance of the transition algorithms and the analyse zero equality technique[80]. All models described in the previous section were measured for the time it took to compute a full simulation[81]. A second indicator for the efficiency of the new transition algorithms is the amount of dead-end transitions generated. The time spent on the ordering procedure was also measured. Three test conditions were used to evaluate the transition algorithms: Firstly the normal configuration of GARP 2.0 was used (condition *A*). Secondly the Subsumption[82] procedure was used in the transition procedure (condition *B*). Thirdly the complete transition procedure of GARP 1.7.2 was used in combination with further GARP 2.0 functionality (condition *C*). Models were not tested with the full GARP 1.7.2 simulator because these models all depend on new inference capacities present in GARP 2.0. Four versions of the component oriented u-tube model have been tested to illustrate how GARP 2.0 utilizes certain modeling constructs. The container-, constraints- and connections model fragments in this model can be labeled constant or not. Correspondences can be used

[77] See section 4.3.3 on continuity
[78] Note that in GARP 1.7.2 this matching would succeed for the Subsumption procedure is less strict then the Specify-Match procedure. See section 4.6 on applying changes.
[79] See section 4.3.1 on constants.
[80] See section 4.10.1 on performance.
[81] All measurements in this section were done using the SWI-Prolog execution profiler running on a Pentium 266 working under windows 98. Since windows offers no possibility to record the CPU-time used by a thread or process, these profiling results do not count CPU-time, but elapsed time. A method that is less precise, but which suffices for the application in this thesis.
[82] See section 4.6 on applying changes.

within the container representation or not. The results of all tests are shown in table 5.2, time is measured in seconds.

Model	*Condition*	*Total Time*	*Ordering Time*	*Valid Transitions*	*Dead-end Transitions*
1) U-tube with: Constants, Correspondences	A	27,9	7,6	12	4
	B	27,9	7,6	12	4
	C	40,4	1,7	12	68
2) U-tube with: Constants	A	26,1	6,3	12	4
	B	25,5	6,4	12	4
	C	522,0	4,7	12	1196
3) U-tube with: Correspondences	A	25,7	6,4	12	4
	B	25,8	6,4	12	4
	C	41,7	1,6	12	68
4) U-tube with: none	A	100,6	6,3	12	268
	B	100,6	6,4	12	268
	C	457,8	4,8	12	1196
5) U-tube: Process Centered	A	4,8	1,1	9	7
	B	4,7	1,0	9	7
	C	-	-	-	-
6) Single Population	A	7,5	0,8	12	6
	B	4,9	0,8	16	2
	C	-	-	-	-

Table 5.2: Performance Test Results

Two main results can be observed. Firstly the amount of dead-end transitions generated is dramatically reduced in GARP 2.0 and this results in significantly faster simulations. Correspondences and constant constraints provide a lot of information in this context. Note that correspondences are not native to the component style of modeling. The notion of constant constraints, which is new in GARP 2.0 is more natural in this approach and can be used to quite some extent as can be seen in the U-tube with constants where the algorithms from GARP 1.7.2 produce 27 times as many dead-end transitions. The second result is that the new Specify-Match procedure is comparable in speed to the subsumption procedure. One would expect the latter to be faster for it does not have to compute a full state in case of a transition to an existing state. Two factors can explain this. Firstly subsuming a partial state description in a complete state description is more expensive then matching two full state descriptions because it involves some inequality reasoning. Secondly in case of a dead-end transition the Specify-Match approach does not first waste time checking existing states like the subsumption approach does, but immediately starts specification of the state and deriving it is contradictive. A minor result is that the ordering procedure used in GARP 2.0 is slower than the ordering procedure used in GARP 1.7.2. The previous results show that this extra time is well spent however.

No results are shown in condition *C* for the process centered U-tube model and the single population model. The former requires the behavior assumption mechanism present in GARP

2.0, thus a very different simulation is generated. The latter model also produces quite a different simulation in the *C* condition. This model depends on the procedure for implementing exogenous behavior. The exogenous derivative should be kept stable but is released in the Garp 1.7.2 transition rules. In the *B* condition this model also produces slightly different behavior. Four transitions succeed that are dead-end transitions in the *A* condition. This different behavior is found because the Subsumption procedure is not as strict in checking continuity constraints as it should be. The measurements in time should still be representative however.

The analyse zero equality technique aims to lower the computational complexity of the inequality reasoning procedure. It is described in section 4.10.1 on performance. To evaluate this technique measurements were made on all models using the normal GARP 2.0 settings with and without the analyse zero equality technique. Both total time computing a simulation and time spent on inequality reasoning were recorded. As can be seen in table 5.3, larger models show a significant reduction of inequality reasoning time in the positive condition. In smaller models the analyse zero equality technique slows down the process by a small amount of time. This difference can be explained by the fact that the extra time spent on applying the analyse zero equality technique is never won back in small models. This is because small models do less inequality reasoning on smaller sets of equations that offer a low computational complexity anyhow.

Model	***Analyse Zero Equality***	***Total Time***	***Inequality Reasoning Time***
1)	yes	27,9	27,2
	no	54,9	55,1
2)	yes	26,1	26,4
	no	53,9	55,1
3)	yes	25,7	25,6
	no	74,3	75,4
4)	yes	100,6	86,8
	no	144,4	132,2
5)	yes	4,8	2,8
	no	4,3	2,2
6)	yes	7,5	3,8
	no	7,1	3,4

Table 5.3: Analyse Zero Equality Performance

This concludes the evaluation of GARP 2.0. The next section features a conclusion, a discussion of the work, and suggestions for further research.

6. Conclusion and Discussion

6.1 Conclusion:

Goal of this thesis has been to improve existing Qualitative Reasoning techniques by dealing with issues in the GARP reasoning engine. Two main subject areas were explored as well as a number of smaller issues.

The first issue concerns resolving ambiguous influences on quantities. It is shown that by assuming the effects of influences to be equal to their magnitude, inequality reasoning can be used to determine the net effect. This extra reasoning capacity provides a useful modeling tool. It also facilitates more precise and 'in depth' modeling which reduces the amount of spurious behavior in the simulation. Using transitive inequality reasoning instead of sign calculus inherently puts an extra computational load on the reasoning engine. The inequality reasoning procedure in GARP was made more efficient on larger sets of equations by adding an extra data reduction step. The inequality reasoning procedure was further improved by adding two missing inference capacities: algebraic simplification and derivation of the most constraining equality $A = B$ from two less constraining relations $A \leq B$ and $A \geq B$.

Secondly a new procedure for inferring successors of a behavioral state was constructed. The dedicated algorithms in GARP 2.0 have access to the elaborate internal mathematical model of a state instead of the user level representation. Therefore they can infer occurring changes far more precisely. Behavior can be assumed in undetermined situations. This provides a valuable modeling tool. In determining valid combinations of changes two concepts play an important role. Firstly a distinction can be made between constant and dynamic expressions. Types of expressions always playing one of these roles were identified. Inequalities can either play the role of a constant constraint or a dynamic description. A labeling mechanism solves this problem. Constant expressions remain in effect throughout the simulation and these constraints provide information on valid combinations of changes. Using this information, the 'search space' for successor states is narrowed down and combinatorial explosion is prevented. A declarative procedure is used to evaluate correspondences in this context, while the flexible inequality reasoning capacities are used to evaluate the effect of constant inequalities. Secondly an expressive and flexible method for recording derived constraints on valid combinations of changes is used. This allows the algorithm to make exhaustive use of information sources and apply relatively expensive inferences on small enough, yet crucial subsets of changes. A detailed tracer is present in GARP 2.0 to illustrate all these reasoning steps made. In qualitative reasoning, behavior of derivatives is undetermined, therefore continuity rules allow these to change a little over each state transition. In GARP 2.0 inequalities on derivatives are also given this freedom to move unless they are specifically labeled as a constant. A new algorithm is constructed to finish transitions. Successor states are now always fully computed before being matched with existing states. This guarantees finding the full spectrum of behavior for a system. This procedure has proven to be computationally as efficient as its predecessor.

Smaller issues dealt with, include the following: Five behavioral patterns ranging from steady to free movement were implemented for exogenous variables in GARP 2.0. These patterns should cover the needs of modelers. The complex behavioral patterns seem only useful in small-scale simulations as state graphs can grow too large to be interpreted. Algorithm option

switches were added to provide the user with a choice between various reasoning strategies. These range from a strict approach resulting in minimal inferences to a full branching, full assumptive approach. A default setting is present and specific settings can be associated with models. In GARP 2.0 correspondence types for derivatives have been added as well as shortcut types for often used combinations of correspondences. These new types provide extra descriptive concepts to make abstractions of functional relations. During simulation, inequalities between landmarks can be derived specifying the possible world described in that branch of simulation. These are recorded in GARP 2.0 to keep different simulation branches, describing different possible worlds, from interacting. GARP 2.0 offers the possibility of removing quantities associated with inactive processes. This is a useful option when modeling using a process-centered approach (Forbus, 1984).

6.2 Discussion and Suggestions for Further Research:

The distinction made in GARP 2.0 between constant and dynamic inequalities using labels may be considered a first step in this area. A drawback of this solution is that inequalities cannot be labeled as constants in the input scenario and a single model fragment can only model constant or dynamic inequalities and not both. In a future implementation of GARP constant and dynamic dependencies could be consistently kept apart by defining separate fields (text based interface) or classes (graphical interface) in the model fragments, scenario's and state descriptions. A more strict approach could be taken considering dynamic inequalities. These should not be allowed to change or disappear without an explicit termination of behavior. Furthermore, the assumption made in GARP 2.0 that inequalities with more then two quantities are constant is not necessarily justified. An example is that in case of an ambiguous influence resolution result such an inequality can describe the underlying relation between influences. If this relation can change as its (summed) derivatives indicate, it can be added to the simulation and provide a useful constraint and mechanism for behavior. A future version of GARP can let go of this assumption and feature behavior for such equations. The explicit notion of constants can take the place of this assumption and the above-mentioned relation can be used.

Results in this thesis show that determining valid combinations of changes is improved significantly in GARP 2.0, but there is still room for improvement if larger subsets of the possible changes are considered. A good heuristic is needed to select these subsets as costs grow with the size of the subset. More use can be made of the intermediate ordering result if sets of constraints are labeled with the minimal context of changes to which they apply.

States with no temporal extent (instants) also form an interesting subject. These states can be distinguished by the fact that a quantity is at a landmark and moving or an equality is changing into an inequality. Generated transitions for these states are immediate, but it is possible in GARP 2.0 that these transitions do not succeed in producing a valid successor state. It can be argued that such a state is illegal. Because the transitions fail, the 'instant' state becomes and endpoint in the stategraph. Endpoints intuitively have longer time duration. A similar argument can be given for quantities and inequalities that are indicated to be constant. These should not have derivatives indicating change. Thus the derivative of a constant quantity should be zero and the (summed) left- and right hand side derivatives of a constant inequality should be equal. For example, if the inequality $A + B = C - D$ is constant, then $\delta A + \delta B = \delta C - \delta D$ should hold. A related point is that since immediate changes happen instantly, it can be argued that they should all happen at once and form one transition. Mutual exclusive changes can occur when assuming behavior however and of course these should not be combined.

The derivation of landmark relations in GARP 2.0 has not shown to constrain illegal behavior yet, because no systems with circular behavior have been considered. Future models will have to prove the practical worth of this mechanism. A side effect is that more landmark relations are present in the user level state representation. In VisiGarp this may lead to overcrowded dependency diagrams. A distinction between landmark relations and other inequalities can be made allowing the user the option not to inspect landmark relations.

Transitive inequality reasoning remains a difficult subject as it involves a tradeoff between inference capacities and computational tractability. Presently inequality reasoning in GARP is reasonably fast, but when equalities involving addition and subtraction are involved sometimes a derivable relation is not found. This does not occur often however.

One of the goals of Qualitative Reasoning is to provide explanations through explicit knowledge encoding and reasoning. From the user's perspective some generic declarative knowledge was lost by discarding the rule based transition procedure present in GARP 1.7.2. The tracer present in GARP 2.0 copes with this problem, but more work needs to be done on the communication of reasoning steps to the user in a compact form. Another approach could be to design a new explicit declarative rule format now that principles and requirements for the transition step have become clearer.

Finally, as noted in section 3 assumptions are implemented in a limited fashion in GARP and a more profound treatment is still welcome. Order of Magnitude reasoning is also still missing in GARP. Several proposals for dealing with this kind of reasoning have been made in the field and possibilities for integration in GARP can be studied.

Acknowledgements:

The author would like to thank:
Bert Bredeweg,
Hermine Linnebank, &
Jos Linnebank †.

Availability of the software:

GARP, VisiGarp and HOMER are available for non-commercial use via the web:
www.swi.psy.uva.nl/projects/garp/

References

Bessa Machado V. & Bredeweg B. (2003) *Building Qualitative Models with HOMER: A Study in Usability and Support*, Proceedings of the 17th International workshop on Qualitative Reasoning, QR'03, p.39-46, Brasilia, Brazil

Bredeweg, B. (1992) *Expertise in Qualitative Prediction of Behaviour,* Ph.D. thesis, University of Amsterdam, Amsterdam, The Netherlands

Bredeweg, B. & Forbus K. D. (2003) Qualitative Modeling in Education, *AI magazine,* vol 24, no 4, p.35-46

Bouwer, A. & Bredeweg, B. (2001) VisiGarp: Graphical Representation of Qualitative Simulation Models, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, p.294-305, IOS-Press/Ohmsha, Osaka, Japan

Falkenhainer, B. & Forbus K. D. (1991) Compositional modeling: finding the right model for the job, *Artificial Intelligence*, vol 51, p.95-143

Forbus, K. D. (1984) Qualitative Process Theory, *Artificial Intelligence*, vol 24, p.85-168

Forbus, K. D. (1988) Qualitative Physics: Past, Present, and Future, *Exploring Artifcial Intelligence*, Howard Shrobe (editor), p.239-296, Morgan Kaufmann Publishers

Goddijn, F. & Bouwer, A. & Bredeweg B. (2003) *Automatically Generating Tutoring Questions for Qualitative Simulations,* Proceedings of the 17th International workshop on Qualitative Reasoning, QR'03, p.87-94, Brasilia, Brazil

Iwasaki, Y. & Simon, H. A. (1986) Causality in Device Behavior, *Artificial Intelligence*, vol 29, p.3-32

De Kleer, J. & Brown, J. S. (1984) A Qualitative Physics Based on Confluences, *Artificial Intelligence*, vol 24, p.7-83

Kuipers, B. (1986) Qualitative Simulation, *Artificial Intelligence*, vol 29, p.289-338

Mavrovouniotis M. L. & Stephanopoulos, G. (1988) Formal Order-of-magnitude Reasoning in Process Engineering, *Computer Chemical Engineering*, vol 12, p.867-880

MONET (2003) *Monet Summerschool*, monet.aber.ac.uk:8080/monet/summer_school_2003/summer_school.html, website visited July 29, 2004

QR04 (2004) *18th International Workshop on Qualitative Reasoning*, www.qrg.cs.northwestern.edu/QR04/, website visited July 29, 2004

QRSER (2003) *Qualitative Reasoning Models on Stream Ecosystem Restoration & Recovery*, www.qrser.de, website visited July 24, 2004

Raiman, R (1991) Order of magnitude reasoning, *Artificial Intelligence*, vol 51, p.11-38

Reinders, M. (1989) *Quartz,* Master thesis, University of Amsterdam, Amsterdam, The Netherlands

Rickel, J. & Porter, B. (1997) Automated modeling of complex systems to answer prediction questions*, Artificial Intelligence*, vol 93, p.201-260

Salles, P. & Bredeweg, B. (1997) *Building Qualitative Models in Ecology*, Proceedings of the 11th International workshop on Qualitative Reasoning, p.155-164, Italy

Salles, P. & Bredeweg, B. (2003) Qualitative Reasoning about Population and Community Ecology, *AI magazine*, vol 24, no 4, p.77-90

Simmons, R. (1986) *"Commonsense" Arithmetic Reasoning,* Proceedings of AAAI-86, p.118-124, Philadelphia, California.

Struss, P. & Price, C. (2003) Model-Based Systems in the Automotive Industry*, AI magazine*, vol 24, no 4, p.17-33

Travé-Massuyès, L. & Ironi, L. & Dague, P. (2003) Mathematical Foundations of Qualitative Reasoning, *AI magazine*, vol 24, no 4, p.91-106

Weld, D. & De Kleer, J. (editors) (1990) *Readings in Qualitative Reasoning about Physical Systems,* Morgan Kaufmann, San Francisco, California

Williams, B. C. & Ingham, M.D. & Chung, S. & Elliott, P & Hofbaur, M. & Sullivan, G.T. (2003) Model-Based Programming of Fault-Aware Systems, *AI magazine*, vol 24, no 4, p.61-76