Simulation preferences – Means towards usable QR engines

Bert Bredeweg and Floris Linnebank
University of Amsterdam, Informatics institute, Science Park 904, 1098 XH Amsterdam, The Netherlands
B.Bredeweg@uva.nl

Abstract

When developing QR software there are choices to be made regarding the exact reasoning of the software. Moreover, what works well in one situation, may become a problem in another, or experts may disagree on the preferred reasoning. Instead of forcing a solution and potentially end-up with a partially suboptimal tool for users or certain models, we have developed and implemented the idea of Simulation Preferences. In Garp3 and DynaLearn users can fine-tune the working of the reasoning engine to their particular insights and needs. These preferences are stored in the model, and automatically activated when re-opening the model. This paper reports on these Simulation Preferences.

Introduction

Working with domain experts on building qualitative models [6,7,8,9] inspired us to augment the problem solver underlying Garp3[1] [1] and DynaLearn[2] [2] with user definable features controlling the reasoning behaviour of the engine. These *Simulation Preferences* allow modellers to build models taking a certain reasoning perspective.

Over the years, three drivers have steered the development of simulation preferences. First, the tension between generating a fully correct simulation output, and keeping those results simple and easy understand. For instance, imposing reasoning with higher order derivatives improves the quality of the output, yet the result may become confusing for less knowledgeable users, e.g. in learning situations. Second, maintaining upward compatibility in the context of ongoing software improvements. In many 'knowledge capture' projects the development of tools happens in parallel with the construction of knowledge, often as a deliberate choice to have the reasoning actually accommodate the needs emerging from the knowledge construction effort. But, when new features become available they may significantly change or even invalidate earlier results, leading to the knowledge engineers having to (partially) redo their earlier work. However, such redoing may not be possible or undesirable by those involved, and a better solution may be to have the option to put those new features off in order to retain the earlier established results. Third, enabling step-wise development and feature discovery in collaboration with users. Both, building problem solvers and knowledge construction are iterative processes for which the 'real' requirements are often established while being submerged in the process. This means that recently developed features need to be explored and used in context to better understand their use and required functioning. Following such a maturation process some features become fully intertwined with the basic reasoning, others get abandoned, and yet others remain as choices that some users will select and others will not.

This paper discusses the simulation preferences available in both Garp3 and DynaLearn. The next section highlights the elementary reasoning of the problem solver for both these workbenches. The subsequent sections discuss the simulation preferences.

Essentials of the problem solver used in Garp3 and DynaLearn

The qualitative reason engine underlying the Garp3 and DynaLearn workbenches generates state-graphs using a depth-first state-by-state approach, essentially following the evolution of the predicted system behaviour over time. The primary reason for taking this step-wise approach is to allow users to be in control regarding the paths of behaviours to explore. This is beneficial for experts when creating a new model, but also for learners engaged in learning by modelling, as it automatically provides an intuitive guide through a complex web of possibilities [3,4].

Garp3 and DynaLearn are advanced workbenches for conceptual knowledge capture, using a multitude of problem solving functions to predict and explain system

[1] http://www.Garp3.org
[2] http://www.DynaLearn.eu

behaviour. *State* and *Transition* are key notions in this context (Fig. 1). States represent unique sets of constrains on quantity values (pairs of <magnitude, derivative>) such as: magnitude X=0, magnitude in/equality X=Y, derivative ∂X=0, and derivative in/equality ∂X=∂Y. Transitions from a state to its successor(s) reflect changes in such sets, e.g. X=0 → X>0, X=Y → X>Y, ∂X=0 → ∂X>0, and ∂X=∂Y → ∂X>∂Y (and also for 2nd and 3rd order derivatives).

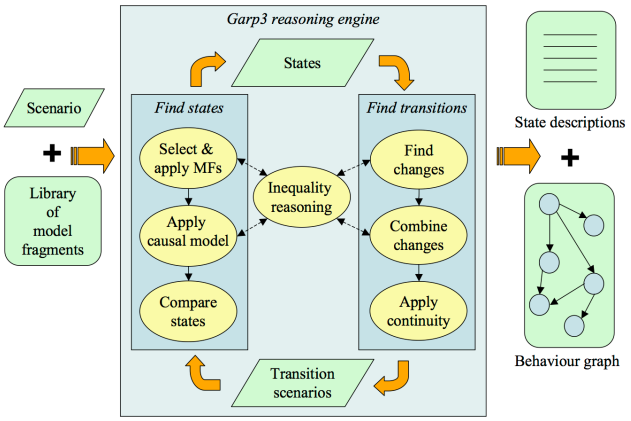

Figure 1. Garp3 reasoning engine global architecture (from [1]). The two main inferences are *Find states* and *Find transitions*. A simulation takes as input a scenario and a library of model fragments, and generates a state-graph (or behaviour-graph). A state-graph may have multiple behaviour-paths (unique successive state trajectories).

Computing states and transitions is performed by the following two top-level inferences.

Find states:

- Select and Apply MFs – Assembling the model-fragments from the library that match the scenario.
- Apply causal model – Computing the net result of the causal dependencies (typically Influences (I's) and Proportionalities (P's) [10]) and if present exogenous quantity behaviours [5].
- Compare states – For newly generated states check to see if they match an already existing state, or that they actually represent new unique behaviour.

Find transitions:

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

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

Preferences on handling unknown information

The simulation preferences addressed in this paper, are listed in 3 categories. The first category concerns different kinds of missing information (Table 1). The second category concerns choices that tighten the reasoning (Table 2). Note that restricting the reasoning may both enlarge and reduce the output (number of states and transitions between those). The third group refers to a set of miscellaneous preferences (Table 3). In this section, the first category is discussed.

Influence resolution and unknown information (#1 – #3)

The preferences #1, #2 and #3, on unknown information, address the influence resolution. When evaluating the effects of influences and proportionalities unknown magnitudes (for the I's) and derivatives (for the P's) at the start of causal chains can prohibit the evaluation of their causal effects. Preference #1 (Assume unknown influence to be zero)[3] causes the reasoning engine to set influencing unknown magnitudes and/or derivatives to 0 for all quantities that are themselves not influenced, before starting the influence resolution procedure. Users can thus decide to leave unknown information unknown, or explicitly assume it has no impact on the system behaviour.

In some cases this setting may cause conflicts with later results of the influence resolution, causing a contradiction (due to in/equality for example). In this case, the corresponding state is not generated. For example: quantity A is proportional to quantity B and C. Furthermore ∂C>0 and ∂B='unknown'. With preference #1 on, ∂B=0 will then be assumed. Now suppose that the following equality is also present ∂A=∂B. The influence resolution will calculate ∂A>0, while the equality suggests ∂A=0, a contradiction.

[3] In earlier work this was called 'Apply closed world assumption in influence resolution' [11].

When evaluating higher order effects on influences and proportionalities, preference #2 sets the 2nd order derivatives of all unaffected quantities to 0, if these 2nd and 3rd order derivatives are unknown. Preference #3 does the equivalent for 3rd order derivatives.

Table 1. Preferences for handling unknown information (T=training/E=expert)

#	Name	T	E
1	Assume unknown influence to be zero	Off	Off
2	Assume unknown 2nd order influences to be 0	Off	Off
3	Assume unknown 3rd order influences to be 0	Off	Off
4	Assume value terminations	Off	On
5	Assume inequality terminations	Off	On
6	Generate derivative terminations (based on 2nd order derivatives)	Off	On
7	Terminate ambiguous derivatives	Off	On
8	Generate terminations for ≤ & ≥	Off	Off
9	Generate all values for calculated quantities	Off	Off
10	Assume equal quantity spaces have equal points	Off	Off
11	Assume equal length intervals	Off	Off

Find terminations using indirect proof (#4 – #8)

Simulation preferences #4 through #8 affect the *Find changes* step. Notice that as a principle, the reasoning engine used in Garp3 and DynaLearn will only generate possible terminations that actually 'have a cause'. That is, for which there is information available that necessarily leads to the change. For instance, quantity X will want to change magnitude when $\partial X \neq 0$, but not when ∂X='unknown'. This is a strict interpretation, which is particular relevant in the context of education, and for providing explanations. Not having a clear cause for a change hampers the software's ability to present an argument on why the system behaviour evolved as it did. Experts on the other hand, may want to ignore this, or take a different perspective on this principled starting point. Particularly, the idea of Reductio Ad Absurdum (RAA) (proof by contradiction or indirect proof) may be taken as an alternative. If in the example, it can be proven that $\partial X \geq 0$ leads to contradiction, while $\partial X < 0$ results in a valid transition, then the latter must be true. Because in qualitative reasoning the set of possible variations of this kind is relatively small (that is, value assignments and in/equality statements), RAA may also be a very effective instrument for generating behaviours in the context of unknown information [12]. Preferences #4 and #5 allow users to deliberately exploit the RAA idea. There are however additional options, allowing users to fine-tune a particular flavour.

- A quantity X will only move if it has a known derivative, and it also holds that $\partial X \neq 0$. With preference #4 switched on, the reasoning engine will assume unequal derivatives for unknown derivatives (constrained by relations that are known) and fire terminations accordingly.
- An inequality relation such as X>Y will only terminate when it is known that the involved derivatives are unequal ($\partial X \neq \partial Y$). With preference #5 switched on, the mechanism triggered by preference #4 is also applied to quantities having in/equality relationships.
- 2nd order derivatives can cause 1st order derivatives to terminate when preference #6 is turned on. This behaviour can be necessary to correctly capture the behaviour of a system. For example, an ambiguous 1st order derivative may become non-ambiguous and cause a dead-end (contradiction in successor states) in the simulation. E.g. a state where the quantity is increasing cannot change into a decreasing state without first becoming steady (due to continuity constraints) via a separate termination. Preference #6 introduces this flexibility by generating essential 1st order changes imposed by 2nd order derivatives.
- Preference #7 causes terminations for ambiguous derivatives to be generated such that these can freely change. Note that this can be important for correct behaviour. Ambiguous derivatives are always branched and specified (states are generated for each possibility). And without free transitions between such states, when a derivative becomes unambiguously known in a successor state, this may lead to a contradiction and a dead-end state because of continuity constraints (a change from >0 directly to <0 and vice versa is not permitted).
- By design weaker relations (notably ≤ & ≥) will not be terminated by the *Find changes* step. Preference #8 causes terminations also to be generated for these relations. The relation ≥ will terminate to > in the case of the left hand side rising relative to the right hand side. Or it may terminate into = and < in the opposite case.

Assuming information relevant for inequality reasoning (#9 – #11)

Simulation preferences #9, #10 and #11 affect the *Inequality reasoning* step. Magnitudes remain unknown unless fully determined by the model. Unlike derivatives, which are branched and specified if they are ambiguous. Preference #9 activates an assumption mechanism that generates all magnitudes for ambiguous quantities. It acts in the following two situations:

- When the user specifies this behaviour for a quantity in the scenario (referred to as exogenous quantity behaviour [5], and available when adding a quantity to a scenario by selecting the option 'Generate all magnitudes').
- When a magnitude is determined by an addition or a subtraction that leaves an ambiguous result. E.g. *if* X–Y=Z, X>0, Y>0, *then* Z cannot be uniquely computed. Instead of leaving Z='unknown', this preferences will try to impose all the possible solutions (Z<0, Z=0, Z>0) and states will be computed accordingly (of course contradictory states will be abandoned).

This mechanism ensures that quantity magnitudes do not remain unknown in a sparsely specified model and that the full behaviour is generated. Note that having many extra assumable magnitudes is a computational burden. This preference is therefore turned off by default, both for Training (T) and Expert (E) (see implementation section).

Simulation preferences #10 and #11 make assumptions regarding the relationship between Quantity Spaces (QS). In Garp3 and DynaLearn QSs are ordered sets of values (alternating points and intervals) assigned to a quantity, specifying magnitudes that this quantity may take on[4]. For instance the Height of a container can be one of {0, Plus, Max} (referring to empty, partially filled, and fully filled, respectively). By definition QSs are unique, and magnitudes from different QSs (even with the same label/name) are not assumed to be equal without explicit specification. Thus, if a model includes two containers, each with a Height (H_X and H_Y) and the QS {Zero, Plus, Max} assigned to them, it cannot be assumed that $H_X(Max)=H_Y(Max)$. The only exception to this rule is 0, which is universal in a model.

Although this approach is a deliberate choice, it is sometimes confusing for beginners. And for experts it may be cumbersome when having to specify many equality statements between QS point values. Preference #10 is in this respect straightforward: it makes the reasoning engine assume equality statements between all the points that have the same position in two QSs of equal type. This simplifies the expert's modelling effort in some situations, and learners can be given a scaffold [13] by using this setting and make things initially easier to understand.

Preference #11 refines the concept discussed above. When switched on, the reasoning engine assumes all intervals within a quantity space to have equal length. This knowledge is expressed by adding the required addition and subtraction constraints.

[4] Notice that this definition of quantity space is rather different from its use in QPT [10].

Preferences on tighter reasoning

This section focuses on simulation preferences that alter the problem solving method by enforcing of relaxing certain constraints (Table 2). Tightening the reasoning typically leads to a better, but often more difficult to understand, simulation result. In addition, the computation time may significantly increase when certain constraints are enforced. It is within these dimensions that the following list of preferences has found its way into the reasoning engine.

Importance of immediate terminations (#12 – #14)

Simulation preferences #12, #13 and #14 address epsilon ordering [12]. This concept distinguishes immediate terminations (from a point, and from equality) and non-immediate terminations (to a point, and to equality). The concept is based on the idea that a point in a QS occupies no space on the line of real numbers. Therefore if a quantity is on a point and it moves it will leave the point in an infinitely small amount of time. On the other hand, a quantity moving to a point will have always some 'epsilon' amount of space between itself and the point. Therefore the transition to the point is not immediate. Because of this, immediate transitions take precedence over non-immediate transitions. However, particularly in the NaturNet-Redime[5] project there was discussion about whether 'mixed' transitions were possible [14]. One question has been whether an interval and a point could be corresponding since this correspondence could demand a 'mixed' transition. Preference #12 makes the reasoning engine work according to epsilon ordering logic, but users can decide to put it off, supporting both viewpoints held.

Ordering possible terminations is a computational costly task. To improve efficiency the epsilon ordering can be applied first in the ordering step (*Combine changes*). That is, if there are immediate terminations, these will be ordered first. The resulting set is checked, and only if it is empty the set of non-immediate terminations will be ordered. This way, valid terminations will never be removed without reason. This procedure is illustrated in Fig. 2. Preference #13 groups all immediate terminations as much as possible (mutual exclusive terminations cannot be combined, e.g. random exogenous behaviours in opposite directions for the same quantity). An extra consequence of this approach is that mixed combinations (involving immediate and non-immediate terminations) are not possible anymore. This can be seen as a positive effect since a strict interpretation of the epsilon concept would not allow these combinations either. However, for backward compatibility reasons, the strict implementation

[5] http://hcs.science.uva.nl/projects/NNR/

of the epsilon ordering can be put off using preferences #12 and #13.

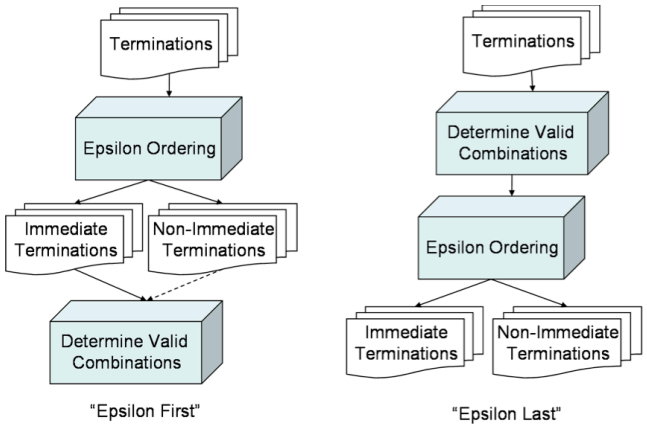

Figure 2. Epsilon ordering computational sequences

The notion of immediate transitions also has an impact on derivatives, as these may also not change to 0 in an immediate transition. The rationale behind this rule is similar to the epsilon ordering. First, it is a non-immediate event since it is a transition towards a point. Second, it is a non-immediate event since it represents the event that the balance of influences on this quantity (whose derivative is going to zero) changes from unequal to equal. This type of inequality change is of course non-immediate. Preference #14 ensures that derivatives cannot become 0 (no halting) during instantaneous transitions. When this preference is switched on, quantities will not stabilise over an immediate transition, but first keep increasing or decreasing when entering an interval, and may only stabilise in a successor state.

Higher order derivatives information (#15 – #17)

Simulation preferences #15, #16 and #17 allow for exploiting information on higher order derivates. To correctly capture the behaviour of a system, 2nd order derivatives are sometimes needed. With preference #15 turned on, 2nd order derivatives are calculated by a procedure similar to the regular influence resolution. 2nd order derivatives can only be calculated if at least one direct influence is present in the causal chain to generate the first 2nd order derivative information using the derivative of the influencing quantity. Also, all of these influencing quantities must have a determined derivative. When preference #16 is switched on, 2nd order derivatives are also part of the continuity constraints for the 1st order derivatives. However, both these preferences are default off in the training set to ease understanding of the simulation output, and for reasons of efficiency. In larger models, especially models with long chains of direct causal relations (notably I+/I-), 3rd order derivatives may become informative to system behaviour. For efficiency reasons, 3rd order derivatives are not calculated by default. Preference #17 turns this behaviour on. Note that in general, calculation and propagation of 2nd order derivatives is needed for this functionality to have an effect.

Table 2. Preferences for reasoning tightening (T=training/E=expert)

#	Name	T	E
12	Apply epsilon ordering	On	On
13	Apply epsilon merging of immediate terminations	On	On
14	Apply epsilon derivative continuity constraints	On	On
15	Calculate 2nd order derivatives	Off	On
16	Apply 2nd order derivative continuity constraints	Off	On
17	Calculate 3rd order derivatives	Off	Off
18	Use correspondence in ordering	On	On
19	Use constants in ordering	On	On
20	Allow reasoning assumptions on derivatives	On	On
21	Apply continuity on derivative inequalities	On	On
22	Constrain interaction between possible worlds (derive landmark relations)	Off	Off
23	Extra thorough inequality reasoning	On	On
24	Compare Derivatives (CD) for similar quantity pairs	Off	Off
25	Extend CD: include proportionalities	Off	Off
26	Refine CD: (26a) equal target quantity type / (26b) equal source quantity type / (26c) equal target entity type / (26d) equal source entity type / (26e) equal causal dependency sign	Off	Off

Ordering information (#18 – #19)

Simulation preferences #18 and #19 affect the grouping of possible terminations. Correspondences (between magnitudes) are a good information source for ordering terminations (*Combine changes*) in the *Find transition* step. With preference #18 switched on, terminations that do not fit the known correspondences in a state are removed thereby saving computational resources. In general, correspondences are valid throughout a simulation, but in models with correspondences changing from state to state this preference should be turned off. Preference #19 causes the reasoning engine to use relations labelled as constant (defined as such using exogenous quantity behaviour in the scenario) and relations involving addition and subtraction (these are assumed constant by the

problem solver) to order terminations in the transition process. Terminations not compatible with these constants are removed thereby gaining computational efficiency.

Continuity information (#20 – #22)

Simulation preferences #20, #21 and #22 affect the continuity enforced across successive states. When applying a transition (determining which model fragments apply to the Transition scenario), model fragments from the current state are reconsidered. Assumptions made in the previous state that are not contradictory at this point are kept. Assumptions about derivatives are not made at this point however, because these cannot be contradicted during the *Select and apply model fragment* step, since derivatives are set later in the *Apply causal model* step (using influence resolution). As such model fragments might be incorrectly retained after a transition. Still, with preference #20 switched on, the reasoning engine will make these assumptions on derivatives speeding up the reasoning process.

The continuity regime in the problem solver releases derivatives over state transitions under constraints to allow continuous change. With preference #21 switched on this regime is also applied to derivative inequalities. In the course of a simulation new inequality relations between landmarks may become derivable. When preference #22 is on, these relations are added to the state description to prevent simulation branches describing different possible worlds from interacting. For reasons of efficiency preference #22 is default set to off.

Heuristic inequality reasoning (#23)

Simulation preference #23 affects the *Inequality reasoning* step. The reasoning engines can apply heuristics to reduce the size of the search space, and increase efficiency. A very strong heuristic is to not allow inferences to be made on *all* combinations of relations but only on a subset of the relations. When preference #23 is on this heuristic is *not* used. When off, the heuristic is applied, leading to less thorough in/equality reasoning, and possible incorrect output.

Comparative analysis (#24 – #26)

Simulation preferences #24, #25 and #26 implement a kind of comparative analysis. Different from [15] the impact is computed within a single simulation (state-graph), and the analysis typically imposes order on the sequence of terminations happening. Comparative derivatives can become computational costly. However, it is only needed for particular models. Hence, preferences have been introduced. The analysis is default put off. Preference #24 can be used to activate the basic set up, which computes order between all quantities and their changes imposed by direct influences. Preference #25 broadens the scope, also including changes caused by proportionalities (see preferences #31 and #32 for implications). Preference #26 refines the scope of reasoning activated by #24 and #25 in 5 ways (each option is in fact a preference by itself), by putting constraints on the pairs that will actually be compared.

Miscellaneous preferences

Miscellaneous features are group in Table 3 and discussed in this section.

Constraints on extreme point values (#27 – #28)

Preferences #27 and #28 determine whether certain constraints will be enforced upon extreme magnitudes (values at the end of a QS). To assure that a quantity remains within the bounds of its QS some constraints are automatically added [1]. For derivatives, specific constraints are added for the highest and lowest *points* in the QS. This is because a quantity cannot be increasing in the top point of a QS or decreasing in the bottom point. For example, in the case of QS={0, Plus, Max} (see also text on preference #10 and #11) the following conditional relation is added: if H_X=Max then $\partial H_X \leq 0$. If the top of the QS is an interval, no constraint is needed, since this interval is infinitely extended. However, these derivative constraints have been made optional providing users a flexible approach to modelling. A model may be a partial description of a larger phenomenon where the QS is also part of a larger domain. E.g. overflow in a container may be modelled as an increasing amount in the top landmark. Preference #27 restricts the quantity behaviour such that increasing in the uppermost point of a QS is not possible, and decreasing in the lowermost point of a QS is not possible.

Note that preference #27 does not affect behaviour at 0, as this is a special point. Preference #28 handles 0. This option prevents derivatives from increasing or decreasing in the highest or lowest point of a QS if this point is 0. Preference #27 and #28 have also turned out to be handy instruments for model debugging.

Fastest path heuristic (#29)

Preference #29 allows for switching on the fastest path heuristic. Simulations often have multiple paths to end states. In many cases, these paths are practically very similar in the behaviour they represent: the same set of changes will occur. This option gives precedence to those transition scenarios that apply as many as possible of these terminations at once. If these transition scenarios produce a valid state then the transition scenarios that are ‘subsets’ are discarded. It is a heuristic because some end-states may

not be found. The added value is largely on communication. When a model is fully developed, applying the fastest path heuristic may significantly simplify the state-graph, making the results easier to share and discuss.

Table 3. Miscellaneous preferences for reasoning adjustment (T=training/E=expert)

#	Name	T	E
27	Apply quantity space constraints on extreme values	Off	On
28	Apply quantity space constraints on zero as extreme value	On	On
29	Apply fastest path heuristic	Off	Off
30	Remove inactive quantities after transition	Off	Off
31	Propagate 2nd order derivatives over proportionalities	Off	Off
32	Propagate 3rd order derivatives over proportionalities	Off	Off
33	Maximum inequality reasoning depth	0	0
34	Allow reasoning assumptions	On	On
35	Remove terminations to unequal for full corresponding quantities	Off	Off

Remove inactive quantities (#30)

Preference #30 is rather special purpose. When on, it removes quantities that are no longer mentioned in the set of active model fragments. Typically this may be a quantity associated with a process, where the process stops and disappears. The added value is again communication.

Propagate derivative information over proportionalities (#31 – #32)

Preferences #31 and #32 impose an assumption on the shape of the functions represented by proportionalities, and because of that allow the problem solver to propagate derivative information over proportionalities. Note that this extends the assumptions made about the underlying function modelled by the proportionality beyond the normal definition of proportionalities. In general, changes of 1st order derivatives of the related quantities can be unequal. Similarly, changes of the 2nd order derivatives of the related quantities can be unequal. Therefore these preferences are off, even for the default expert settings. However, when put on (#31 for 2nd and #32 for 3rd), simulation results tend to become more specific because more information can be applied. Using these preferences, users can thus decide whether the assumption applies to their domain and model, and act accordingly.

Setting maximum search depth (#33) and Miscellaneous (#34 – #35)

Preference #33 restricts the maximum search depth for in/equality reasoning. Restricting the search depth can improve efficiency on complex models that take too long to compute. In in/equality reasoning two parent relations are combined to derive a new relation. This new relation can in turn become a parent of a new relation, etc. This preference controls the amount of parent relations a derived relation can have. Of course, this carries the risk of not finding derivable relations, and consequently generating invalid states. Search depth restrictions in the range of 10 to 20 have been successfully used in the past, but for new models it is recommended to be conservative and apply an experimental approach. In order to not limit the search depth, the preference should be turned off (by setting its value to 0).

When computing a state from a transition or scenario, model fragments are considered. Model fragments with conditions that are explicitly known are applied first. When preference #34 is switched on, the problem solver makes assumptions on conditions to include non-contradictory model fragments with conditions that are not explicitly known.

Turn preference #35 on to remove terminations from equal to unequal from the transitions for quantities that have a full correspondence.

Implementation and Appearance

The problem solver has two default settings for the simulation preferences: Training (T) and Expert (E) (referred to as Modelling in Fig. 3).

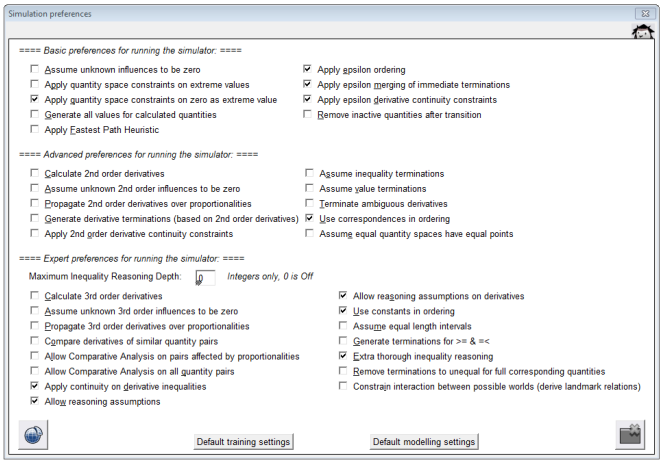

Figure 3. Interactive screen for Simulation Preference selection

Figure 3 shows the default training set (details are also shown in Table 1, 2 and 3). This set takes the position that

models will remain simple and that advanced reasoning may be confusing for a beginner. Hence, the preferences are chosen to accommodate these constraints. As such, spontaneously generating terminations (e.g. by using 'Terminate ambiguous derivatives' (#7)) is not included in this set. The default expert set on the other hand, emphasizes the correctness of the reasoning. Hence, it includes preferences such as 'Calculate 2nd order derivatives' (#15). In both cases however, not all possible constraints are applied. The idea being that full application on average will hamper even expert level modelling, rather than supporting it. However, users can adjust the preferences to their liking and needs.

The selected preferences are stored as part of the model, and automatically imposed upon the reasoning when the model is re-opened and simulated. Taking this approach, the simulation preferences are thus a feature of the model, rather then of the problem solver. They assure that the same results are obtained when the model shared.

There are three groups of preferences that are each strongly associated with a common theoretical concept. These are the preferences concerning epsilon ordering, 2nd order derivatives, and comparative analysis. Preferences in these groups have been placed together to invite the user to consider them all at once. Furthermore a message such as the one in Fig. 4 will be displayed if a user simultaneously selects and deselects preferences in a particular group. This informs the user that these preferences are related to the same theoretical concept and should be turned on or off simultaneously in most cases.

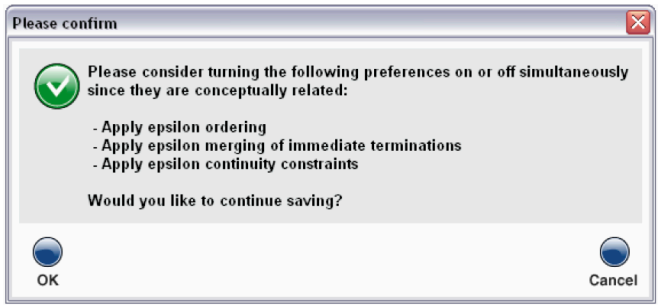

Figure 4. Conceptual relationship between simulation preferences warning

Related work

The work on filters, particularly global filters, developed for the QSIM algorithm [16] seems related to the idea of simulation preferences. However, the development of global filters largely focussed on identifying and removing spurious behaviours, not on the creation of a knowledge engineering instrument. Second, despite their need for ensuring correct simulation results, having the simulation preferences allows users to disable certain key reasoning features, if they think they have reasons to do so. An example is the calculation of 2nd order derivatives (#15). Third, this paper discusses many preference controlled reasoning functions that have no counterpart in QSIM due to fundamental differences in the underlying architecture. E.g. the notion of model fragments does not exist in QSIM, and hence preference #30 (Remove inactive quantities after transition) has no resemblance in that context. Similar arguments apply to #5 (Assume inequality terminations), #18 (Use correspondence in ordering), and many of the other preferences discussed in this paper.

Concluding remarks

This paper gives a review of the simulation preferences that have been developed and implemented as part of the reasoning engine underlying the Garp3 and DynaLearn conceptual knowledge construction workbenches. Drivers for establishing these preferences have been (*i*) ease of understanding versus correct simulation output, (*ii*) maintaining upward compatibility in the context of software development, and (*iii*) knowledge capture and reasoning features developed as such. The main goal of having the simulation preferences available as an interactive feature is to support knowledge workers in building models taking a certain reasoning perspective. Instead of deciding upon which features to actually implement, and potentially end-up with a partially suboptimal tool for users or certain models, we acknowledge that the creation of conceptual models is inherently a heuristic process, and empower the user with the ability to select and fine-tune features as needed. As such, this work is part of our ongoing endeavour to create usable QR workbenches.

Acknowledgement

The work presented in this paper is co-funded by the EC within the 7th FP, Project no. 231526, and Website: http://www.DynaLearn.eu.

References

[1] Bredeweg, B., Linnebank, F., Bouwer, A. and Liem, J. (2009). Garp3 – Workbench for Qualitative Modelling and Simulation. *Ecological Informatics* 4(5-6), 263-281.

[2] Bredeweg, B., Liem, J., Beek, W., Salles, P. and Linnebank, F. (2010). Learning Spaces as Representational Scaffolds for Learning Conceptual Knowledge of System Behaviour. In Wolpers, M., Kirschner, P.A., Scheffel, M., Lindstaedt, S. and Dimitrova, V. (eds.), Sustaining TEL: From Innovation to Learning and Practice (EC-TEL 2010), p47-61, LNCS 6383, Barcelona, Spain.

[3] Bredeweg, B. and Schut, C. (1991). Cognitive plausibility of a conceptual framework for modeling problem solving

expertise. Proceedings of the 13th Conference of Cognitive Science Society, K.J. Hammond and D. Gentner (eds.), August, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 473-479.

[4] De Koning, K., Bredeweg, B., Breuker, J. and Wielinga, B. (2000). Model-Based Reasoning about Learner Behaviour. *Artificial Intelligence*, 117(2), pp.173-229.

[5] Bredeweg, B., Salles, P., Nuttle, T. (2007). Using exogenous quantities in qualitative models about environmental sustainability. AI Communications, 20(1), pp. 49-58.

[6] B. Bredeweg, P. Salles, J. Bertels, D. Rafalowicz, A. Bouwer, J. Liem, G. M. Feltrini, A. L. R. Caldas, M. M. P. Resende, A. Zitek, and T. Nuttle. (2007). Training report on using QR for learning about sustainable development. Naturnet-Redime, EC FP6 STREP project 004074, Deliverable D7.2.

[7] Noble, R., Salles, P., Zitek, A., Mioduser, D., Zuzovsky, R. and Borisova, P. (2011). DynaLearn curriculum reflection and advancement. DynaLearn, EC FP7 STREP project 231526, Deliverable D6.3.

[8] P. Salles and B. Bredeweg. (2007). Library of reusable QR model fragments. Naturnet-Redime, EC FP6 STREP project 004074, Deliverable D6.7.1.

[9] Salles, P., Assumpção Costa e Silva, P., Gontijo de Sá, I., Noble, R., Zitek, A., Uzunov, Y. and Mioduser, D. (2009). DynaLearn environmental science curriculum requirements. DynaLearn, EC FP7 STREP project 231526, Deliverable D6.1.

[10] Forbus, K.D. (1984). Qualitative process theory. *Artificial Intelligence*, 24, pp. 85-168.

[11] Bredeweg, B., Bouwer, A., Jellema, J., Bertels, D., Linnebank, F. and Liem. J. (2006). Garp3 - a new workbench for qualitative reasoning and modelling. In C. Bailey-Kellogg and B. Kuipers (eds), 20th International Workshop on Qualitative Reasoning (QR-06), pp 21–28, Hanover, New Hampshire, USA.

[12] Kleer, de J. and Brown, J.S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, 24, pp. 7–83.

[13] Soloway, E., Guzdial, M. and Hay, K. E. (1994). Learner-centered design: The challenge for HCI in the 21st century. *Interactions*, 1, pp. 36–48.

[14] Bouwer, A., Liem, J., Linnebank, F. and Bredeweg, B. (2007). Analysis of frequently asked questions and improvements of the Garp3 workbench Naturnet-Redime, EC FP6 STREP project 004074, Deliverable D4.2.3.

[15] Weld, D. (1988). Comparative Analysis. *Artificial Intelligence*, 36, pp. 333-374.

[16] Kuipers, B. (1994). Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge, 452 pages, MIT Press.