Providing Feedback for common problems in learning by conceptual modeling using expectation-driven consistency maintenance

Beek, W.G.J.; Bredeweg, B.

Publication date
2012
Document Version
Final published version
Published in
26th International Workshop on Qualitative Reasoning (QR'26), Playa del Rey, CA, USA

Providing Feedback for Common Problems in Learning by Conceptual Modeling using Expectation-Driven Consistency Maintenance

Wouter Beek and **Bert Bredeweg**
Informatics Institute, University of Amsterdam

Abstract

Learning by building models and reflecting on them by running simulations is an important paradigm in education. New modeling tools are emerging that take this approach, and focus on learning conceptual knowledge explaining how systems behave and why they do so. These tools are promising in the learning effect they bring about, but progress gets hampered when learners want, but are unable to adjust their model so that the simulation results align with their expectations. In the research reported here, automated support is being developed that helps learners identify the problem(s) in their model. To this end, model-based techniques developed for diagnosing devices are adapted and used to automate the conceptual modeling assistance. This paper describes the solutions developed thus far.

1 Introduction

Computer-based modeling is a powerful means for humans to develop their understanding of phenomena (Papert 1980; Salles and Bredeweg 2006) by allowing them to express and externalize their thinking. Conceptual modeling environments that allow theories to be visualized and tested can make a significant contribution to improving scientific education (Biswas et al. 2001; Forbus et al. 2001; 2004; Bredeweg et al. 2009; Kinnebrew and Biswas 2011). These tools employ a qualitative vocabulary (Kuipers 1994; Bredeweg and Struss 2003; Forbus 2008) for users to construct their explanation of phenomena, notably about systems and how they behave.

When working with conceptual modeling tools, learners go through a construct-reflect cycle, typically by creating the model and analyzing the simulation results. The simulation becomes an instrument that helps learners reflect on their knowledge and work towards improving it.

When using this approach in practice, learners continuously run into situations in which the simulation results of their model turn out to be different from what they expected them to be. There are essentially two ways to align simulation results with expectations thereof: learners can change their model, or they can change their expectations. Both are relevant for learning, and the two often interact. However, a problem occurs when learners want to adjust their model so as to make the simulation results align with their expectations, but are unable to do so. They get stuck in a model debug-loop without solving the problem.

The work in this paper addresses this problem by developing software that engages learners in a knowledge construction dialogue, which coaches learners to reason in a coherent manner. This allows learners to debug their model so that the simulation results align with their expectations. For this we employ Model-Based Diagnosis (MBD) (de Kleer and Williams 1987; Reiter 1987), which was originally developed for diagnosing electrical circuits, and the adjustment of this technology for cognitive diagnosis (de Koning et al. 2000; de Koning 1997; Self 1993).

This paper reports on ongoing research. We explain how this approach can be applied to typical cases drawn from educational practice, showing that consistency maintenance using diagnostic techniques is a viable approach in learning contexts.

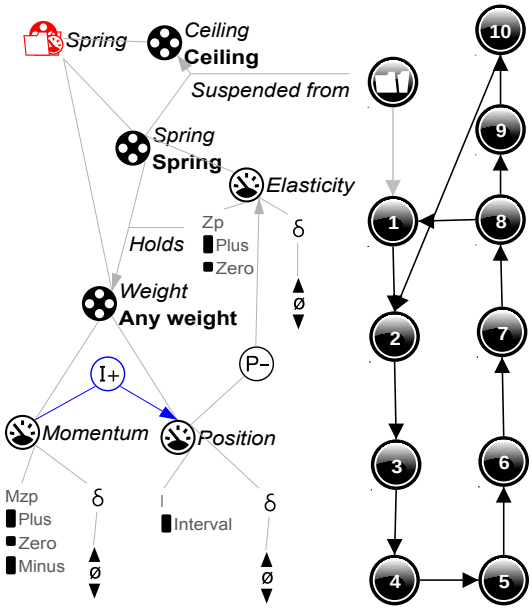
2 Conceptual modeling with DynaLearn

Using the DynaLearn Integrated Learning Environment (ILE) (Bredeweg et al. 2010), learners construct knowledge by manipulating icons and their interrelationships, using a diagrammatic representation. The knowledge representation is based on Garp3 (Bredeweg et al. 2009). The diagrams represent models that can be simulated confronting learners with the behavioral consequences of the knowledge they represented.

In conceptual models, the quantities that describe the dynamic features of a system hold qualitative information concerning the current value and direction of change, using an interval scale consisting of an ordered set of labels. Landmarks are specific point values that refer to situations in which the behavior of the system changes significantly.

Theory on Qualitative Reasoning (QR) has resulted in a set of dependencies that capture cause-effect relationships between quantities. These dependencies are defined so that on the one hand they represent conceptual notions that closely match human reasoning (de Koning and Bredeweg 1998), while on the other hand they are grounded in mathematical formalisms allowing automated computation. Two examples of such dependencies are influences (caused by processes) and proportionalities (causal propagations) (Forbus 1984).

Simulation results represent system behavior and how it



(a) Model fragment (b) State-graph

Figure 1: The model fragment in figure 1a shows the relations between *Momentum*, *Position*, and *Elasticity* in the oscillation model. The state graph in figure 1b shows the system's cyclic behavior.

(a) Simulated value history (b) Expected value history

Figure 2: Figure 2a shows the simulated and figure 2b shows the expected values for the oscillation model. (The state numbers in figure 2b correspond to those in figure 1b.)

evolves over time. Time is represented as a graph of states (including loops) that reflect qualitatively distinct behavior. States have duration, but the exact length of the duration is unknown. State transitions occur when values of quantities change (increase or decrease). Ambiguity is represented by a state having multiple transitions into future states.

3 Common conceptual modeling errors

Learning by conceptual modeling is a strong paradigm for learning. But in classroom situations learners often come across discrepancies between how the model behaves in simulation and how they want the model to behave. This section discusses three often occurring problem types learners experience during conceptual model construction (Bredeweg et al. 2006; Mioduser et al. 2011). The applicability of our approach to these problem types is evaluated in the expert review in section 8.

Causal differentiation

A common modeling problem for learners is the causal differentiation between a direct influence (I) and an indirect influence or proportionality (P). To illustrate the problem, we use a model of an oscillating weight that is attached to a string. The weight has momentum and position (relative to the normal extension to the spring). The momentum directly influences the position (I^+ in figure 1a). The position negatively propagates that change to elasticity (P^- in figure 1a). If we ignore the damping of the spring, then the simulation results show a behavioral cycle as in figure 1b.

When a learner does not differentiate between the two types of causation correctly s/he may model a (positive) proportionality instead of an influence from *Momentum* to *Position*. Under this alteration, the actual simulation results (figure 2a) look very different from the expected results (figure 2b). As occurs often, small changes in structure result in big changes in behavior. The learner can express a dissatisfaction with the results as shown in figure 2a, for instance by stating that s/he was expecting to see the *Position* and *Elasticity* oscillate. Even when a learner is able to express such an expectation, tracking the discrepancy down to the proportionality relation in one of the model fragments is not straightforward. In practice, an intervention by a teacher is often required.

Feedback

A second common problem concerns feedback. Learners often ignore feedback, not seeing the why it is needed. An example is shown in figure 3 with a feedback relation (P^+) from *Pressure* back to *Flow*. Learners often fail to properly represent the dynamics of a feedback and instead simply add a value-correspondence (the V-relation in figure 3) between the *Zero* values without representing how the *Flow* can ever change to *Zero*. For this the proportionality between *Pressure* and *Flow* is necessary.

In case of leaving out a feedback loop, learners often express a dissatisfaction with the simulation results, such as "Why does the container never get empty?", but they typically fail to resolve the problem by themselves.

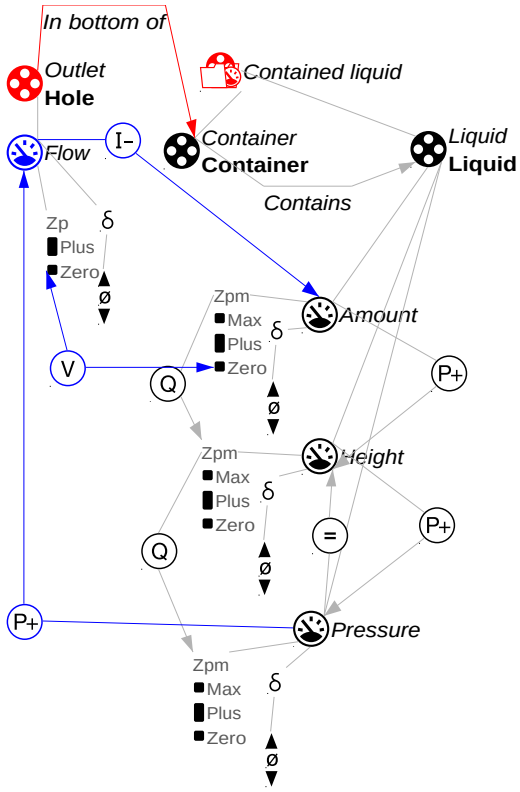


Figure 3: Model fragment representing a container with water flowing out through a hole in the bottom. The *Flow* makes the liquid's *Amount* decrease. The decrease in *Amount* is then propagated to *Height* and to *Pressure*. The *V*-relation states that the *Zero* values of *Flow* and *Amount* must coincide (a value-correspondence).

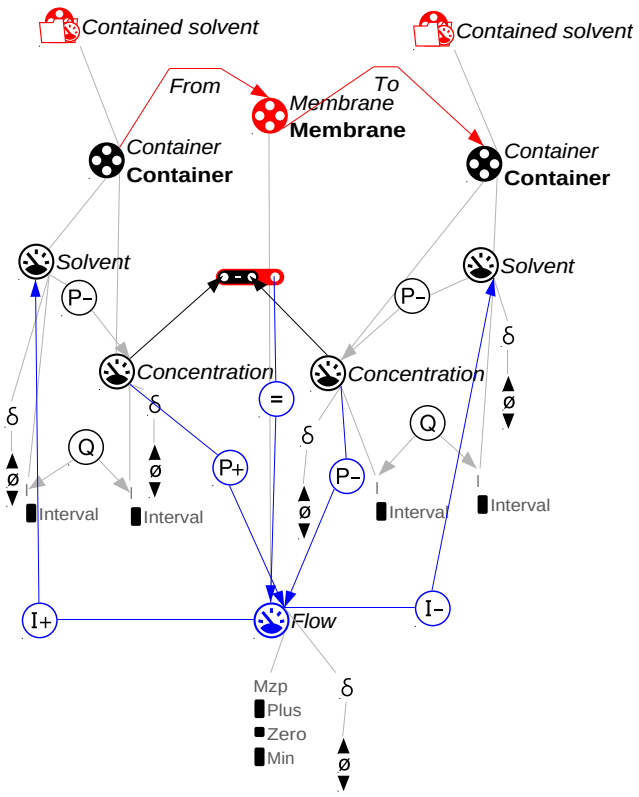


Figure 4: Model fragment representing the process of osmosis between two containers, via a permeable membrane. Both containers have *Concentration* and *Solvent* quantities.

Consistency between (semi)symmetrical sides

A third recurring problem is that of maintaining consistency between two identical sides in a (partially) symmetrical model. Instantiating the same set of model ingredients twice and connecting both parts using multiple dependencies while maintaining a consistent interpretation among those newly added ingredients, turns out to be difficult for learners. Particularly when one of those ingredients is a subtraction or addition.

Figure 4 shows an example of this problem for osmosis 4. Both sides of the osmosis model are connected to the *Flow* quantity via proportionalities and influences. The signs of the causal relationships depend on how we model the flow through the membrane (a convention). In this model we represent fluid moving from the left to the right as a positive flow. The signs of the causal relations are chosen accordingly. The same is true of the calculus, which must be *Flow* = *Concentration Left* – *Concentration Right* under the convention we chose. Learners often make a mistake in one or more of the signs or order of the arguments in the calculation. The discrepancies between the expected outcome and the simulation results under such a mistake are not easy to localize to the causal relations or to the calculus.

4 Model-based diagnosis

The goal of technical diagnosis is to single out the faulty components that cause a device to behave inappropriately. Once found, the device is repaired by replacing these components by ones that behave correctly. The General Diagnostic Engine (GDE) (de Kleer and Williams 1987) is an implementation of this approach that has been studied to a great extent and with successful applications, see (Struss and Price 2003).

Technical diagnosis was adjusted for cognitive diagnosis (Self 1993; de Koning 1997; de Koning et al. 2000) (figure 5), the process of inferring a person's cognitive state from his or her performance (Ohlsson 1986). In particular, cognitive diagnosis can be used to find the faulty inference steps in a learner's reasoning. In technical diagnosis a norm model drives the diagnostic algorithm. Similarly, in cognitive diagnosis learners interact with an existing model (created by experts or teachers) and faulty answers are diagnosed using this model as the norm. While developing the approach measures are taken to ensure didactic plausibility (de Koning and Bredeweg 1998).

Interacting with an existing (norm) model does not line up well with contemporary theories on 'active learning' originating from constructivist perspectives on learning, see (Bruner 1966; Vygotsky 1978; Bruner 1996; Otero, Johnson, and Goldberg 1999)). With the arrival of knowledge construction

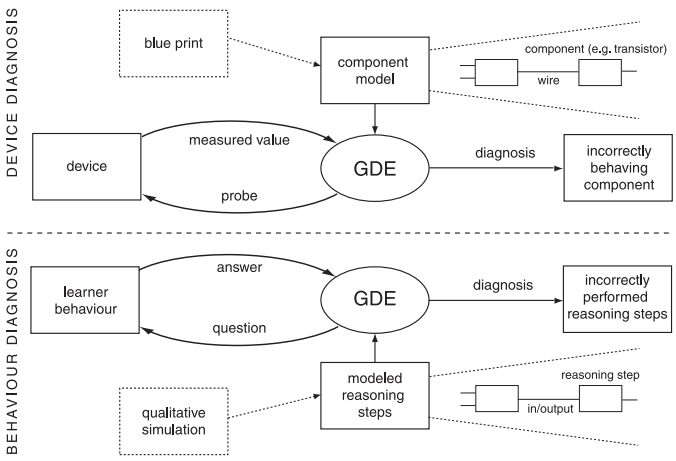


Figure 5: Mapping Model-Based Diagnosis as used for Technical Diagnosis onto Conceptual Diagnosis (from (de Koning et al. 2000)).

tools, active learning is facilitated. The question now is how to automatically generate modeling support for a model and its simulation results without having a norm?

Our approach takes the discrepancies between the actual and the learner-expected simulation results of a model and identifies those aspects of the learner-build model that are accountable for these differences. The goal is to support the learner's knowledge construction endeavor by maintaining consistency between the expression created by the learner and the expectations s/he holds regarding the inferences that can be made on behalf of that expression.

Expectation-based aggregation

Conceptual models present extra complexity for MBD, since they use weak constraints (qualitative instead of quantitative) hampering reasoning with component behavior rules (de Koning et al. 2000). Hierarchies can be used to reduce complexity and warrant tractability. Technical diagnosis typically uses a function-driven approach grouping sets of components into clusters that implement a particular function (Genesereth 1984; Hamscher, Console, and de Kleer 1992). For cognitive diagnosis this approach can not be used because the functions are unknown (the functional units that constitute human problem solving have not been identified). Instead, we have created an algorithm based on the characteristics of the representation used for conceptual models, but tuned to the expectations the learner expresses.

Aggregations made by experts in a norm model put the focus on a particular subset of all possible system behaviors (de Koning et al. 2000), which most likely does not include the modeler's expected behavior. An aggregation algorithm is therefore developed that takes the modeler's expected behavior as the main driver. Since a learner may express any expectation, this tuning of the representation must occur 'online' and be able to work for an extensive number of cases.

5 Component Connection Model

The output from simulations run inside the DynaLearn ILE (Bredeweg et al. 2009) is transformed into a representation that is useful for diagnosis. This representation is called the Component Connection Model (CCM) (Beek and Bredeweg 2011). A partial example of a CCM is given in figure 6. We next discuss the four key ingredients that constitute the representation.

Expressions

Expressions are propositions that encode conceptual knowledge drawn from the simulation results. There are calculi, inequality, causal, correspondence, quantity space, and quantity value expressions. The expressions are added based on an extendable library of expression definitions. In order to overcome intra-state complexity (the number of behavioral states) most reasoning is performed in a-temporal or generic terms and the results are converted to specific spaces.

Points & spaces

Points are expressions that hold within a specific space. A space is a logical context in which certain points and components can occur. We distinguish between the following spaces:

- *Global*: containing the generic information regarding the simulation results.
- *Input*: containing the information that is provided by the simulated scenario.
- *States*: spaces that encode unique configurations of behavior, representing how the modeled system behaves at specific moments in time.
- *Transitions*: spaces that represent the transition from one state into another as well as those from the input space to start states.

A single expression can have multiple points. This means that propositions can hold in multiple spaces. (In figure 6, expression $\delta(A) = +$ is asserted in points 0, 2 and 5.)

Point clouds

For diagnosis it is necessary to represent conflicting information. Point clouds (the ovals in figure 6) allow this to be expressed.

We first introduce the concept of *alternative expressions*. These are expressions that differ only in a single subsentential part. For each type of expression the subsentential part that is allowed to differ is defined apriori. For value assignments, this is the value component. ("The tap's flow is steady" and "The tap's flow is increasing" are examples of alternative expressions.) For inequality statements, the inequality relation type is the part that is allowed to differ. (For example: "The pressure in the left container is larger/smaller than the pressure in the right container.")

Conflicting information is represented by *alternative points*. These are points that belong to the same space and assert alternative expressions. Alternative points are collected within a single *point cloud* (see figure 7). Point clouds

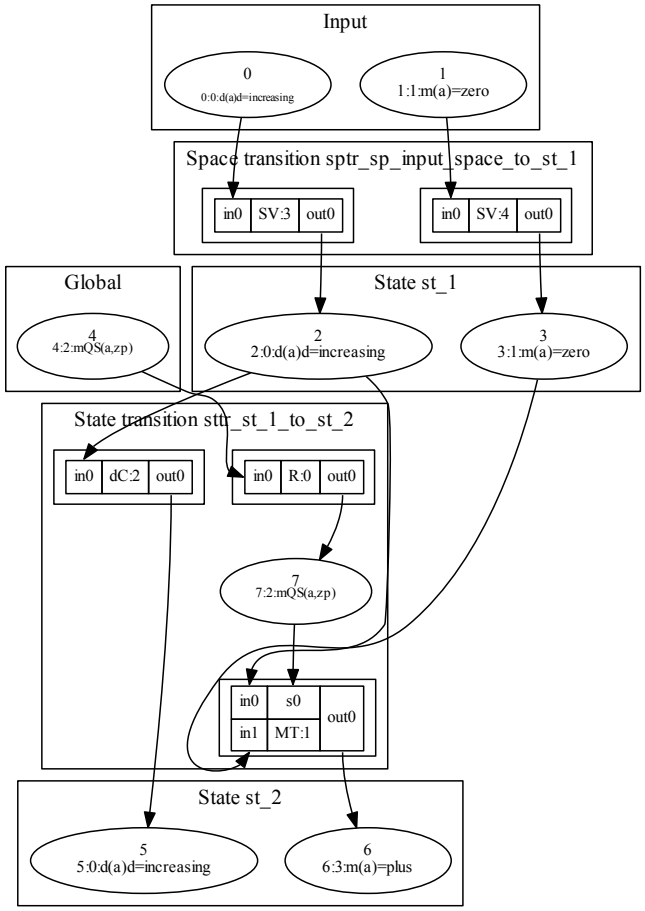


Figure 6: The CCM for a quantity whose magnitude increases. The ovals represent point clouds. The small rectangles represent deduction components. The large rectangles, containing point clouds and components, represent spaces.

therefore indicate the common locale between components in which alternative statements can be asserted. The subsentential parts that are allowed to differ between alternative expressions are precisely chosen so as to ascertain that alternative points take the same place with respect to all the components that can be added to the CCM.

Components

Components are explicit representations of instances of deductions steps the learner must grasp in order to understand the simulation results. A component describes a deduction step that is applied to the expressions that are asserted in the point clouds that connect to its ports (the propositions). The components are added to the CCM based on an extendable library of component definitions. Component definitions specify the number of ports, the port types (which expressions are allowed to occur in connecting point clouds), and the behavioral description of the modeled deduction step.

We distinguish between input (premises), support (describing the deduction step), and output (conclusion) ports. The distinction between in- and output ports refers to the forward propagation of the simulation reasoner and the backward propagation of the causal explanation component (for more information, see (Beek and Bredeweg 2012)). This distinction is irrelevant for the diagnosis reasoner, since in diagnosis each port is allowed to be inferred provided that all the other ports are known.

The behavioral descriptions of components are partial functions for those cases in which a port's outcome is underspecified. For instance the input (stating the magnitude, e.g., $flow = +$) port of the direct influence component cannot be derived from the support (stating the causal relationship, e.g., $I^-(flow, amount)$) and output (stating the derivative, e.g., $\delta(amount) = -$ ports in case the quantity that occurs the input port's expression has a quantity space with more than one positive/negative quantity value label (and no further constraints are present).

The distinction between in- and output ports on the one hand and support ports on the other, refers to the distinction between declarative and procedural knowledge. In terms of the components in the CCM, we can distinguish between two types of discrepancies between the simulation results and a learner's expectations. One discrepancy is caused by a learner having different declarative knowledge. For instance the simulation may use the proposition $I^+(Flow, Amount)$ ("Flow positively influences amount") but the learner may be of the opinion that $I^-(Flow, Amount)$ ("Flow negatively influences amount") is the case. Another discrepancy is caused by a learner not understanding how a deduction rule is performed (i.e., how behavior is calculated). For instance a learner may not be able to infer that *Amount* is increasing in case *Flow* is positive and $I^+(Flow, Amount)$ (e.g., making a different deduction based on the same propositional content). The distinction between declarative and procedural knowledge is represented in the CCM by connecting retrieval components to the support ports of components.

The discrepancies between simulation and expectation can also be due to having wrong input values and/or unrealistic expectations. Components are therefore added for values and inequalities that are set in the scenario or in one of the model fragments as well as for expectations.

6 Expectation-Centric Diagnosis

This section describes the seven main modules that perform expectation-centric diagnosis, from receiving the expectations of the learner to the communication of the diagnostic results.

Expressing expectations

Based on the generated CCM, the diagnosis dialog (figure 8) is populated and displayed to the learner. Expectations can be formulated by selecting items from the lists that are displayed in the dialog. Upon selecting, the expectation label updates a natural language rendition of the current expectation. For example: "I expect that elasticity is increasing in state 3." Incomplete selections result in incomplete sentences, showing what remains to be added. Any expect-

ation can be formed regarding the simulation results. Multiple expectations can be staged, allowing (the equivalent of) compound expectations.

Based on whether a conflict exists between the generated CCM and the formulated expectations, one of the following three approaches is chosen.

1. If the expectations are already met by the simulation, then there is no need to run a diagnosis. The expectation points do not make a change to the CCM. This is communicated to the learner.
2. If expectations add an alternative point to an existing point cloud, then diagnosis is run, since a point cloud with more than one point represents a discrepancy.
3. If expectations are neither identical to nor alternatives of existing points, then new point clouds are created. If the newly created point clouds can be meaningfully related to point clouds that were already in the CCM, a list of one-component extensions that would allow the expectations to be deduced is given. This repair task uses the same behavior rules as the diagnosis as well as the same aggregate components (section 6) in order to suggest 'high-level' fixes.

Aggregation

After expectations have been added to the CCM, the components are aggregated relative to these. Aggregate components are created based on an extendable library of aggregate component definitions. There are two main aggregate component types: competitive and hierarchical.

Competitive components subsume components that need to be taken into account together in order to explain their behavior. Even though the behavior of these subsumed components cannot be determined individually, their combination into a subsuming competitive component, together with an extra input port, can be determined. Competitive components cannot be unpacked by diagnosis (since they model the lowest level at which behavior can be calculated), but they can be unpacked for other uses such as causal explanation.

Hierarchical components subsume components for whom behavior can also be calculated independently. Hierarchical components can thus be unpacked by diagnosis, thereby fulfilling two purposes. Firstly, they reduce within-state complexity in terms of the number of component assumptions in diagnosis (which is convenient for an exponential algorithm). Secondly, they allow for higher-level / more abstract communication of diagnosis results and causal explanations.

It is not know beforehand which expectations a learner will express. Hierarchical aggregation components can abstract away point clouds. For this reason hierarchical aggregation is performed after expectations have been added, and aggregations are generated so that they do not hide expectation points.

Scoping

After aggregation there are often too many components to have a feasible diagnostic procedure. Scoping is used to remove components that cannot be part of conflict sets given the expectation points expressed. This is done by excluding paths that are not related to an expectation point. A path is unrelated if it is not connected, via components and point clouds, to an expectation. Connections that run via observations are not allowed.

If the number of components is still too big a more drastic scoping procedure is used to exclude components that are farthest removed from all the expectations that have been expressed. The scoping and aggregation algorithms ensure that the number of components is kept approximately stable throughout iterations of the diagnosis.

The remaining components form paths connecting expectations to other observations. These are the considered components on which the diagnosis algorithm is run.

Diagnosis

After scoping, the set of relevant components and the set of observations are identified. Observations are points that are registered as premises in the TMS. (These include inputs to retrieval components, inputs to scenario value components, outputs from expectation components, observations due to scoping, and learner-given expectations.) Figure 7 gives an example of a CCM at this stage.

The diagnosis algorithm consists of a hypothesis generator and a problem solver (Struss 2008). The hypotheses in MBD are environments or sets of components that are assumed to be working properly. The environment lattice is generated starting with the empty environment and moving upwards, ending in the maximum environment. In practice the entire environment lattice is rarely generated since only minimal conflicting environments have to be considered.

The problem solver calculates the expressions that are derived based on the observations and a given environment hypothesis. Calculation uses the behavior rules that are specified by the component definitions of the components that are assumed to be working in the given environment. Whenever a new value is calculated based on an environment, the result is registered as a justification in an Assumption-Based TMS (Forbus and de Kleer 1993). Implementation-wise, the ATMS and the CCM are one representation for efficiency reasons, CCM components and points being subtypes of TMS nodes.

Since expectations will generally not agree with the simulation outcomes, alternative expressions will be derived for the same point clouds (calculated under different hypotheses or component assumption sets). If one of the alternative expressions is an observation, then the environment under which that expression is derived is called a conflict. A conflict is a set of components not all of which can be working correctly without giving up consistency. The diagnosis algorithm returns only minimal conflicts.

A set covering procedure is use to extract the minimal candidates based on the minimal conflicts. Candidates are hypotheses for how the CCM of a model that adheres to the expectations differs from the CCM of the actually simulated model.

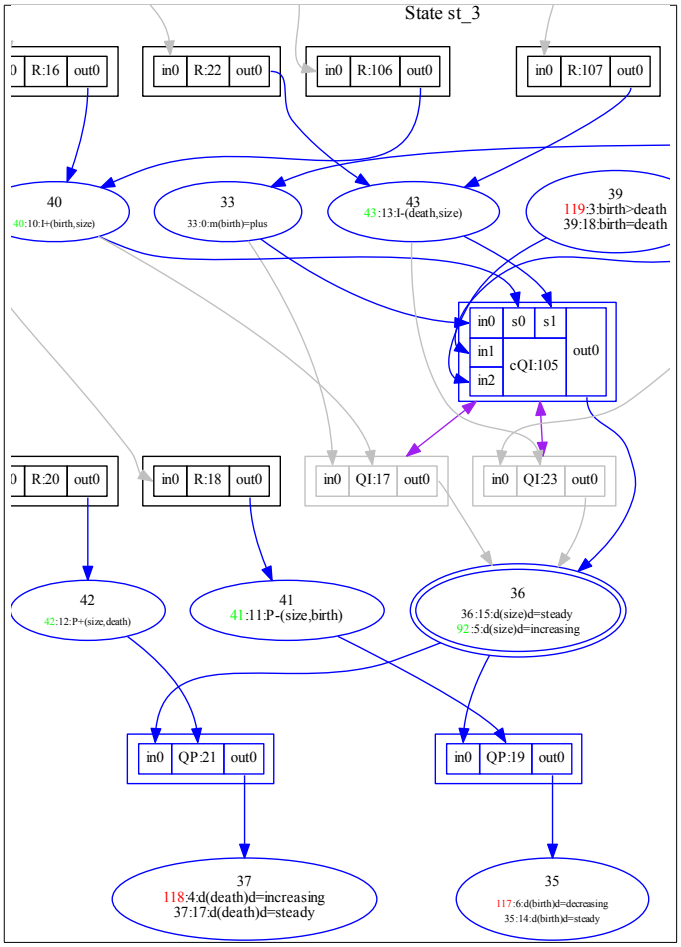


Figure 7: A fragment of a CCM after expectations have been expressed and aggregation and scoping have been performed. Component 105 is a competitive aggregate component. Purple lines connect aggregate components to their subsumed components (17,23) Grey indicates inactivity. Four point clouds contain alternative points (35,36,37,39). A green identifier at the start of a line indicates an observations. A red identifier at the start of a line indicates a point derived by the diagnosis algorithm. The ellipse with a double border indicates an expectation. Abbreviations: *R* retrieval; *QP* quantity proportionality; *QI* quantity influence; *cQI* competitive quantity influence.

Component unpack

Hierarchical component aggregation reduces the complexity of diagnosis (section 6), but also hides point clouds. Sometimes these point clouds can provide additional knowledge for the diagnosis algorithm. In such cases, the aggregate components must be unpacked before additional probes can be performed. Once unpacked, the previously subsumed components can be part of the generated environments, conflicts, and candidates of the next iteration of the diagnosis algorithm.

Hierarchical components are unpacked if they occur in all candidates of the previous diagnosis iteration. Hierarchical components can be unpacked individually, only increasing the number of components if this is strictly necessary.

Probe points

At the end of each diagnosis iteration the possibilities for gaining additional knowledge are determined. Probe points are point clouds in the CCM that are not yet known and whose knowledge could allow the diagnosis algorithm to exclude certain candidate hypotheses. Probe points are sorted according to relevance. The most relevant probe point is turned into a natural language question sentence that is posed in the dialog for the learner to answer. A learner can give answers in the same way as s/he is able to formulate expectations (section 6). Questions and answers can be formulated for each expression that is included through the expression definition library (this also included declarative support knowledge requests).

Since probing means asking a learner questions, we want to perform as little probes as possible to keep the interaction smooth. This is achieved by the weight assignment procedure that ranks the possible point clouds. Firstly, the candidates are given the combined apriori weights of the components they contain. Secondly, the (aposteriori) weights for components are determined by the weights of the candidates in which these components take part. Thirdly, the point clouds are assigned weights depending on the components they connect to (using geometric decay favoring nearby components).

A special case occurs when a point cloud is able to split the graph of considered non-observation point clouds. If such a graph split occurs, then the point cloud involved receives an additional weight gain that is inversely proportional to the size of the largest remaining split (preferring centric splits).

Communicate results

The diagnosis ends when there are no more probe points available. At this point the candidates are the best possible guess of what causes the discrepancy between expectation and simulation results. A second end criterion is reached when there is only one candidate left, so that asking additional questions would not improve the result any further.

In both cases the candidates are communicated via the diagnosis dialog (figure 8). When the learner selects a candidate the corresponding elements in the build model are displayed in the ILE.

It is not always necessary to run a diagnosis until it ends. As soon as all candidates share a component, this can already be communicated as a potential causes of inconsistency. The learner can inspect intermediary outcomes. If s/he does not find intermediate suggestions satisfactory, a probe question can be answered in order to attain better results. The learner is therefore in control as to how long diagnosis takes and whether certain knowledge requests are met.

7 Diagnosis results

The implementation of the diagnosis was run on models of the problem types identified in section 3 (see table 1). The

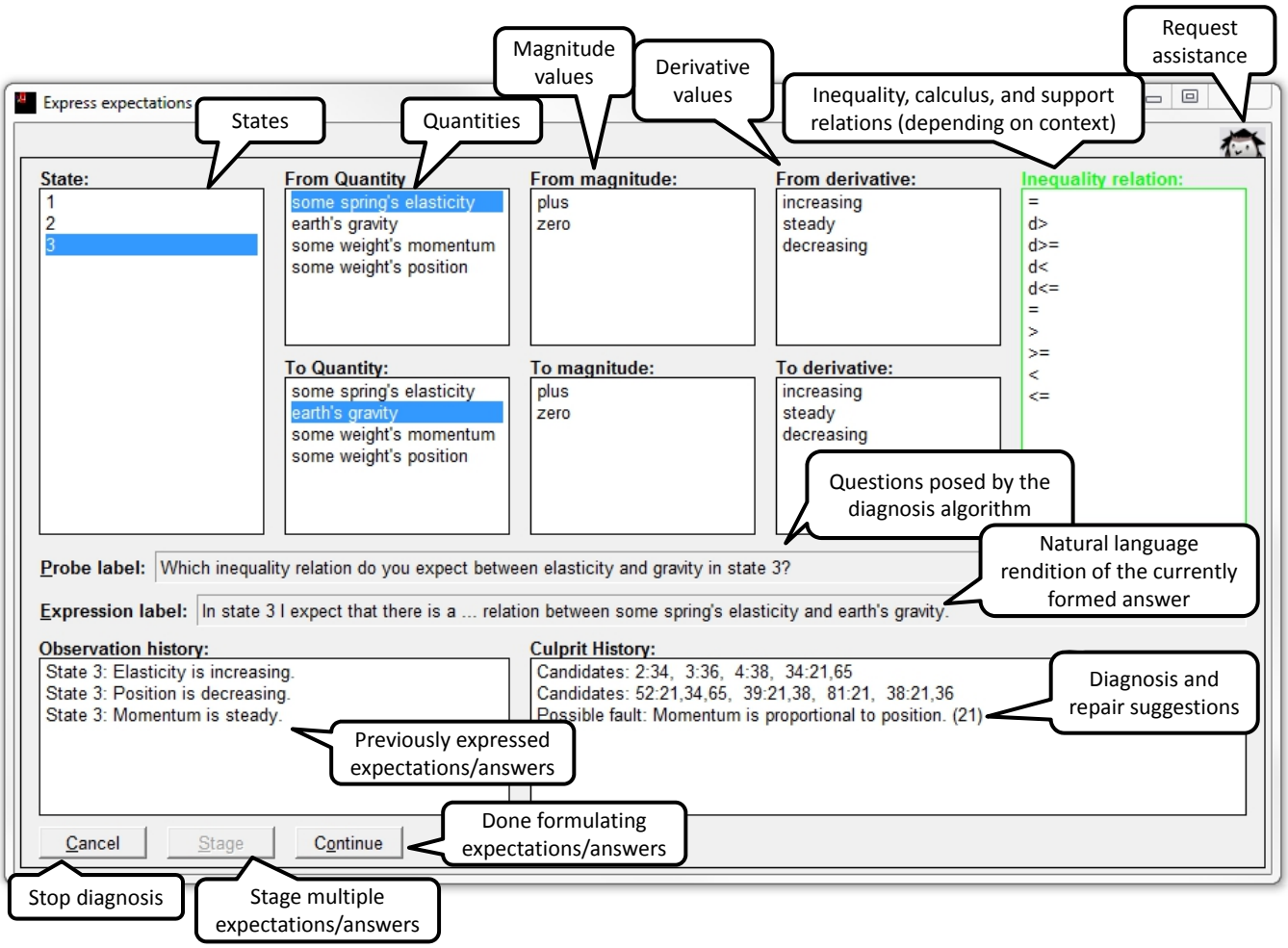


Figure 8: A unified dialog is used for expressing expectations, asking and answering questions, and inspecting the diagnosis and repair suggestions.

diagnosis identifies the culprits that explain the inconsistency between expectation and simulation results. It was able to do so by asking three or less additional questions.

Figure 8 shows the diagnosis dialog after cognitive diagnosis has been performed on the oscillation model (section 3). In this case we started by giving two expectations: "I expect that elasticity is increasing in state 3" and "I expect that position is decreasing in state 3," formed by selecting items from the *state*, *from quantity*, and *from derivative* lists. Based on these expectations no (intermediary) candidates were found and a probe question was posed in the *probe label* field: "What do you expect as the derivative of the weight's momentum in state 3?" The answer the learner gave is shown as the third option in the *observation history*: "I expect that the momentum is steady in state 3". After this answer the diagnosis gives its first intermediary result: "Momentum is proportional to position", in the *culprit history*. A second probe question is then formulated in the *probe label*: "Which inequality relation do you expect between elasticity and gravity in state 3?" The partial answer to this question is shown in the *expression label*: "In state 3 I expect that there is a ... relation between some spring's elasticity and earth's gravity." The missing relation has to be chosen from the *inequality relation* list (which is colored in green to indicate pending input).

8 Expert review

We conducted an expert review with teachers who have taught courses on conceptual modeling at an academic level. We investigated whether the common inconsistencies identified in section 3 and the interactions with the diagnosis are feasible for learners. We recorded five full diagnostic interactions of commonly occurring models and had seven teachers judge the diagnosis process.

The experts had to assess statements on a likert scale of five. The outcomes are displayed in table 2. For each statement there was also a free text form in which the teachers gave comments and explained their scores. The following likert statements were included:

1. The models that diagnosis is run on are instances of dis-

Table 1: Examples of diagnoses of model error types.

Error type	Model name	Expectation	Probe questions	Diagnosis suggestion
Causal differtentiation	Oscillation	I expect that elasticity is increasing and position is decreasing in state 3.	What do you expect as the derivative of the weight's momentum in state 3?	Possible fault: momentum is proportional to position.
Feedback	Bucket	I expect that flow is increasing in state 2.	No questions	Possible fix: Add a positive proportionality between pressure and flow.
	Population	I expect that biomass is	No questions	Possible fix: add a proportionality between death and number-of.
Consistency	Osmosis	I expect that flow is zero and steady in state 1.	What do you expect as the derivative of solvent right?	Possible fault: Flow is equal to left minus right concentration.

Table 2: The results of the answers in the expert review on a likert scale of five.

Question	strong disagree	disagree	neutral	agree	strong agree
Q_1	1	0	1	15	14
Q_2	0	2	10	9	10
Q_3	0	3	5	15	8
Q_4	0	7	11	9	4

crepancies that often occur in classroom situations.

2. The discrepancies would be challenging for a learner to resolve by him/herself.
3. The outputs of the diagnosis pinpoint aspects of the model that are responsible for the discrepancy.
4. The correct diagnosis results can realistically be obtained by a learner who is using the software.

The biggest problem identified by the experts is that it may be difficult for a learner to express expectations and answer probe questions. This requires a primordial understanding of what the model is about and what the learner wants to achieve with the model. The experts indicate that learners often express expectations on a more abstract level (e.g., expecting 'an oscillation' or 'a loop' in behavior) and using vague statements (e.g. "I was expecting more states"). Another problem is that the learner has to decide for himself when he has received sufficient feedback from the diagnosis, since the diagnosis provides no explicit end conditions.

9 Discussion

Diagnosis runs successfully on the problem types for which consistency is maintained by either changing the support knowledge or by changing the component type (or a combination of the two). Amongst these models are those that exhibit one (or more) of the three identified common modeling problem types (section 3). In terms of the modeling language, discrepancies due to the directionality of relations, the inversion of relations, the signs of causal relations, the signs of calculi, the argument order in calculi, and the inequality signs between quantities and their values can be found by the diagnosis. It is also possible to find multiple discrepancies at once (de Kleer and Williams 1987).

Other discrepancies diagnosis handles are a learner having second thoughts about values set in the scenario or in a model fragment, as well as regarding the expectations made. Discrepancies in the changes of values and inequalities over states can also be found (terminations and continuities). Discrepancies that are caused by unintended interactions between model ingredients are found due to the inclusion of competitive components.

Another set of discrepancies are due to missing model ingredients, reflected by reasoning components that are not inside the CCM. This presents a special challenge for future research, because the diagnostic algorithm needs to identify non-existing components in the CCM as being the fault. The diagnostic algorithm needs to be adapted such that, instead of identifying a faulty component, it is able to identify the structural context(s) (input and output terminals of other components) in which a component (or a set of components) is required in order for the model to deliver the expected behavior. Currently only singular missing components (including aggregate components) are found.

Future research is also needed to assist the learner in asserting expectations and answering questions. This can be improved by allowing more generic expectations to be formulated using (modal) quantifiers, such as 'for all', 'always'; by allowing vague formulations, for instance 'almost always', 'at some point'; and by allowing expectations to involve concepts like loops, feedback, and oscillation. Another angle would be to see this as part of the more generic problem that learners do not always have a clear goal in mind when modeling. Future research should investigate how expressing modeling goals beforehand could replace or extend the expression of expectations in diagnosis.

10 Conclusion

Expectation-based diagnosis is able to solve inconsistencies in conceptual models for often occurring problem categories. The current implementation is able to give useful feedback on difficult modeling problems for which no alternative form of automated feedback is given in comparable qualitative learning environments (Leelawong and Biswas 2008; Forbus et al. 2005). The current approach seems fruitful for considering other types of conceptual modeling inconsistencies in the future.

Acknowledgements

This work is co-funded by the European Commission within the 7th Framework Programme project no. 231526. http://www.dynalearn.eu

References

Beek, W., and Bredeweg, B. 2011. Diagnosis and assessment. Deliverable D3.4, DynaLearn.

Beek, W., and Bredeweg, B. 2012. Context-dependent help for novices while acquiring conceptual systems knowledge in dynalearn. In Cerri, A. S.; Clancey, W.; Papadourakis, G.; and Panourgia, K., eds., *Proceedings of ITS 2012*, volume 7315. LNCS.

Biswas, G.; Schwartz, D.; Bransford, J.; and at Vanderbilt (TAG-V), T. T. A. G. 2001. Technology support for complex problem solving: From sad environments to ai. In Forbus, K., and Feltovich, P., eds., *Smart Machines in Education: The Coming Revolution in Education Technology*. AAAI/MIT Press.

Bredeweg, B., and Struss, P. 2003. Current topics in qualitative reasoning. *AI Magazine* 24(4):13–130.

Bredeweg, B.; Liem, J.; Bouwer, A.; and Salles, P. 2006. Curriculum for learning about qr modeling. Deliverable D6.9.1, Naturnet-Redime.

Bredeweg, B.; Linnebank, F.; Bouwer, A.; and Liem, J. 2009. Garp3: Workbench for qualitative modelling and simulation. *Ecological Informatics* 4(5-6):263–281.

Bredeweg, B.; Liem, J.; Linnebank, F.; Bühling, R.; Wissner, M.; del Río, J. G.; Salles, P.; Beek, W.; and Pérez, A. G. 2010. Dynalearn: Architecture and approach for investigating conceptual systems knowledge acquisition. In Aleven, V.; Kay, J.; and Mostow, J., eds., *Intelligent Tutoring Systems 2010*, volume 6095 of *LNCS*, 272–274. Springer-Verlag.

Bruner, J. 1966. *Toward a Theory of Instruction*. Harvard University Press.

Bruner, J. 1996. *The Culture of Education*. Harvard University Press.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

de Koning, K., and Bredeweg, B. 1998. Qualitative reasoning in tutoring interactions. *Interactive Learning Environments* 5:65–80.

de Koning, K.; Bredeweg, B.; Breuker, J.; and Wielinga, B. 2000. Model-based reasoning about learner behaviour. *Artificial Intelligence* 117(2):173–229.

de Koning, K. 1997. *Model-Based Reasoning about Learner Behaviour*. Ph.D. Dissertation, University of Amsterdam.

Forbus, K., and de Kleer, J. 1993. *Building Problem Solvers*. MIT Press.

Forbus, K.; Carney, K.; Harris, R.; and Sherin, H. 2001. A qualitative modeling environment for middle-school students: A progress report. In *The 15th International Workshop on Qualitative Reasoning*.

Forbus, K.; Carney, K.; Sherin, B.; and Ureel, L. 2004. Vmodel: A visual qualitative modeling environment for middle-school students. In *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference*, 820–827.

Forbus, K.; Carney, K.; Sherin, B.; and Ureel, L. 2005. Vmodel: A visual qualitative modeling environment for middle-school students. *AI Magazine* 26(3):63–72.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Forbus, K. 2008. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier Science. chapter Qualitative Modeling, 361–393.

Genesereth, M. 1984. The use of design descriptions in automated diagnosis. *Artificial Intelligence* 24:411–436.

Hamscher, W.; Console, L.; and de Kleer, J., eds. 1992. *Readings in Model-Based Diagnosis*. Morgan Kaufmann.

Kinnebrew, J., and Biswas, G. 2011. Modeling and measuring self-regulated learning in teachable agent environments. *Journal of e-Learning and Knowledge Society* 7(2):19–35.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT.

Leelawong, K., and Biswas, G. 2008. Designing learning by teaching agents: The betty's brain system. *International Journal of Artificial Intelligence in Education* 18(3):181–208.

Mioduser, D.; Zuzovsky, R.; Salles, P.; Souza, A.; Portella, A.; Braga, E.; Wilhelms, L.; Resende, M. P.; Morison, G.; da Costa e Silva, P. A.; de Sá, I. G.; Pires, L.; Simões, A.; Noble, R.; Borisova, P.; Uzunov, Y.; Zurel, D.; Leiba, M.; Benayahu, H.; Nachmias, R.; Ram, J.; Zitek, A.; Poppe, M.; and Stelzhammer, M. 2011. Evaluation of dynalearn: First phase insights. Deliverable D7.2.6, DynaLearn, EC FP7 STREP project 231526.

Ohlsson, S. 1986. Some principles of intelligent tutoring. *Instructional Science* 14:293–326.

Otero, V.; Johnson, A.; and Goldberg, F. 1999. How does the computer facilitate the development of physics knowledge among prospective elementary teachers? *Journal of Education* 181:57–89.

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–96.

Salles, P., and Bredeweg, B. 2006. Modeling population and community dynamics with qualitative reasoning. *Ecological Informatics* 195(1-2):114–128.

Self, J. 1993. Model-based cognitive diagnosis. *UserModeling and User-Adapted Interaction* 3:89–106.

Struss, P., and Price, C. 2003. Model-based systems in the automotive industry. *AI Magazine* 24:17–34.

Struss, P. 2008. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier Science. chapter Model-Based Problem Solving, 89–106.

Vygotsky, L. S. 1978. *Mind in Society*. Harvard University Press.