Reducing Ambiguity by Learning Assembly Behaviour

Bert Bredeweg
Cis Schut
Kees van den Heerik
Maarten van Someren

Department of Social Science Informatics (S.W.I.)
University of Amsterdam, Roetersstraat 15
1018 WB Amsterdam (The Netherlands)
Telephone: +31-20-525 6788, Telefax: +31-20-525 6896
E-mail: bert@swi.psy.uva.nl

June 17, 1992

Abstract

In this paper we present a technique for automatically generating constraints on parameter derivatives that reduce ambiguity in the behaviour prediction. Starting with a behaviour prediction using an initial library containing general domain knowledge the technique employs feedback about correct and incorrect states of behaviour and knowledge about the causal dependencies between the parameters in the model in order to determine the constraints that remove the incorrect or undesired states of behaviour that result from ambiguity. In addition, the technique points out the assembly of physical objects to which the generated constraints apply.

1 Introduction

A recurring issue in qualitative prediction of behaviour (cf. [1; 9]) is the problem of constructing a model that is not ambiguous in the sense that it only predicts behaviours that can actually occur. In particular, when using a library of partial behaviour models modelling general domain knowledge (like processes [5] and device behaviours [3]) the ambiguity introduced by the qualitative calculus, together with the requirement of modelling device behaviour independent from the context in which it operates (the 'no function in structure' principle, cf. [3]), makes it difficult to define adequate prediction models for a specific system. In order to get rid of ambiguity additional constraints must be specified which model: (1) order of magnitudes [8], (2) assembly specific behaviour (functional view), and (3) conservation of quantities for the system as a whole. In this paper we present a technique that automatically derives these constraints by analysing correct and incorrect behaviour predictions from the set of possible behaviours and a model of the underlying causality. In addition, the approach localises the physical structure, and its specific mode of behaviour, to which the constraints apply.

The contents of this paper is as follows. Section 2 provides background information about how the knowledge engineer can be supported during the modelling process. Section 3 describes the framework for qualitative prediction of behaviour as we use it. Section 4 discusses causes of ambiguity and the related problems which we tackle in this paper. Section 5 presents a method for generating the constraints needed for reducing the ambiguity in a behaviour prediction. In particular, it focuses on how to generate candidate constraints and how to discriminate between competing constraints. Section 6 describes how the physical structure can be localised to which the constraints apply. Section 7 discusses the notion of further specification after one or more constraints have been added to the knowledge in the library. Finally, in section 8 we summarise the major results of our research.

2 Supporting the Knowledge Engineer: Automated Modelling

Our approach can be thought of as supporting a knowledge engineer who, on the basis of a library containing general domain knowledge, has to develop a specific model that can be used for a behaviour prediction task.[1] Given such a library with general domain knowledge the knowledge engineer is confronted with two problems: (1) relating the elements from the real-world system that has to be modelled to the canonical entities present in the library, and (2) modelling additional constraints to reduce the ambiguity in the behaviour prediction.

Typically, the knowledge engineer goes through a debugging/refinement process, depending upon the behaviour prediction that the qualitative prediction engine produces. Each predicted state reflects a correct or an incorrect form of behaviour and as such provides feedback for how the models from the initial library must be modified.

The problem of modelling is complex and cannot be automated all at once. In this paper we concentrate on deriving additional constraints on the derivatives of parameters that are required to remove undesired states of behaviour that result from ambiguity. An assumption therefore is that the initial knowledge in the library is sufficient for predicting at least all possible behaviours, but that it can be too general in the sense that it may also predict behaviours that do not occur.

Building and refining qualitative knowledge follows a debugging cycle as depicted in figure 1. After the knowledge engineer has classified (c.q. modelled) some system from the real-world into terms of the canonical elements present in the initial library, the prediction engine generates a graph of possible behaviours. Although in principle this graph may include *all* possible behaviours (correct and incorrect) that can be derived on the basis of the general knowledge in the library, it is usually necessary to limit the number of states to a subset that can still be understood and used by the knowledge engineer (partial behaviour prediction). Next, for each state of behaviour the knowledge engineer determines whether it represents a correct or an incorrect state of behaviour by comparing it with the actual behaviour of the system in the real-world. The sets of correct and incorrect states of behaviour are input for the process of refining the knowledge in the library.[2] The learning

[1] For example, qualitative prediction of behaviour of a device as a subtask of a diagnostic problem solver.

[2] It could also be the case that the canonical description has to be changed (redo classify), but we will

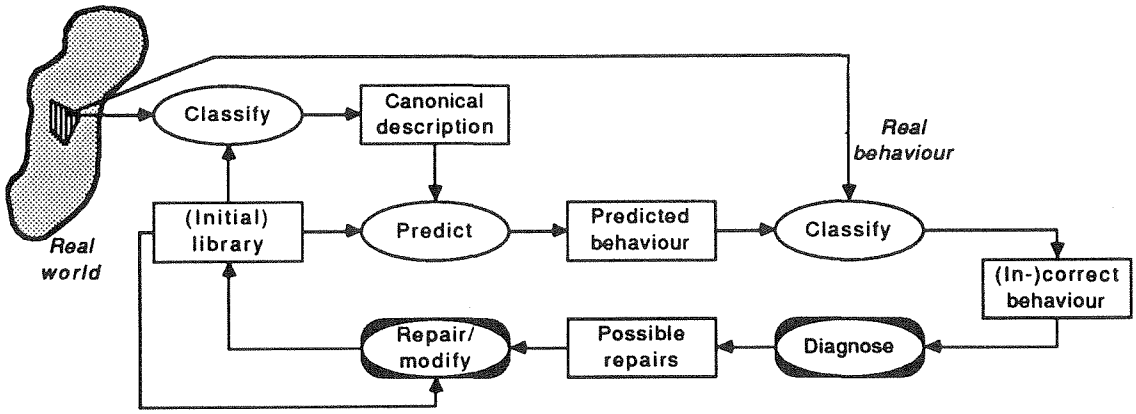

Figure 1: Steps in building prediction models from general library knowledge

task is now to find new constraints that will exclude false predictions (without excluding the correct predictions). It must diagnose the set of correct and incorrect behaviours and determine *which* constraints have to be added to *what* parts of the knowledge in the library in order to remove the ambiguity.

3 Framework for Qualitative Prediction of Behaviour

In this section we describe some important aspects of the framework for qualitative prediction of behaviour that we use. This framework is implemented as a domain independent qualitative reasoning shell, called *GARP*, which can be used by a knowledge engineer for developing prediction models. (cf. [2]).

System element is an important (often implicit) notion in the process of building qualitative models. It refers to (1) how objects from the real-world are represented in the prediction model, and (2) how these representations are applied to guide the behaviour analysis (i.e. the search for applicable behaviour models).

In contrast to using a pure component oriented approach [3] (modelling the physical world as components connected by conduits) or a pure process oriented approach [5] (modelling the physical world as physical objects that interact via processes), we claim that it is essential to use both component and process oriented abstractions in a single prediction model. In addition, system elements may also refer to functional abstractions of the physical reality and as such do not have to map directly onto physical objects.

Similar to the component and process oriented approach, our qualitative prediction engine uses a library of *partial behaviour models* for determining the behaviour of some real-world system. The knowledge in the library discriminates between static, process and agent models. Static models represent general properties of system elements. They can be further divided into single description, composition, and decomposition models, referring to modelling the properties of a single system element, a collection of system elements or to how a system element can be decomposed into its sub-structure. Processes describe changes that are based on inequalities between interacting quantities of different system

not discuss this option in this paper.

elements. Agent models are used for modelling changes that are caused by agents and may have their impact on one or more system elements.

The knowledge representation for modelling partial behaviours is as follows:

Super type relation The partial model can be a subtype of other partial models (multiple inheritance). This means that the super behaviour models must be applicable in order for the subtype to be applicable.

Conditions Each partial model has its own specific conditions that must hold before the knowledge that is specified in the consequences of the model can be used. The following five knowledge types can be conditions:

1. *System elements:* The abstraction from the physical world to which the partial model applies.
2. *Parameters:* Properties of system elements used by parameter values and/or relations.
3. *Parameter values:* Parameter values that must hold.
4. *Parameter relations:* Relations (constraints) between parameters that must hold.
5. *Partial behaviour models:* Other partial models that specify certain knowledge about the behaviour of the real-world system that must be known before the partial model may be used (=applies-to hierarchy).

Consequences When a partial model is applicable the consequences specify the additional knowledge about the behaviour of the real-world system that is derivable. The following five knowledge types can be derived:

1. *System elements:* For processes it may be the case that new entities in the real-world are created because of the behaviour of the system (for example: gas when boiling liquid).
2. *Parameters:* (New) properties that are introduced by the partial model.
3. *Parameter values:* New values for parameters that hold.
4. *Parameter relations:* Additional constraints that hold between parameters.
5. *Partial behaviour models:* (Other) partial models that can be derived.

Finally, table 1 presents an overview of the parameter relations that can be used for modelling the behaviour dependencies between parameter derivatives. The proportionalities and influences are similar to those defined by Forbus [5]. The inequalities between derivatives are similar to the notion of confluences defined by de Kleer [3].[3] The proportionalities and influences can be used for modelling causal dependencies, whereas the inequalities can

[3] In the example discussed in this paper $Arg1$ refers to a parameter and $Arg2$ refers to either a parameter or to the value zero. The former can be used for modelling constraints between a pair of parameters, whereas the latter can be used for relating the derivative of a parameter to zero. The relation d_equal(zero, plus(Arg1, Arg2)) is a specific version of modelling that a sum equals zero. In general, each argument (Arg) of an inequality constraint may (recursively) refer to the sum of two derivatives. For the example presented in this paper a 'sum' constraint between two derivatives is sufficient.

Types	Specific relations	Reference nr.
Inequalities	d_smaller-or_equal(Arg1, Arg2). d_greater_or_equal(Arg1, Arg2). d_equal(Arg1, Arg2). d_greater(Arg1, Arg2). d_smaller(Arg1, Arg2).	1 2 3 4 5
	d_equal(zero, plus(Arg1, Arg2)).	6
Proportionalities	prop_pos(Par1, Par2). prop_neg(Par1, Par2).	
Influences	inf_pos_by(Par1, Par2). inf_neg_by(Par1, Par2).	

Table 1: Dependencies between parameter derivatives

be used to further constrain the ambiguity introduced by these causal relations or for modelling constraints on derivatives that lack a clear causal dependency. As argued in [2] both the causal and non-causal dependencies are essential features of a qualitative model.

4 Causes of Ambiguity

The need for refinement of the knowledge present in the initial library can be illustrated with a prediction model for the refrigerator. Figure 2 visualises the important physical objects of the refrigerator. To model the behaviour of these objects a combination of

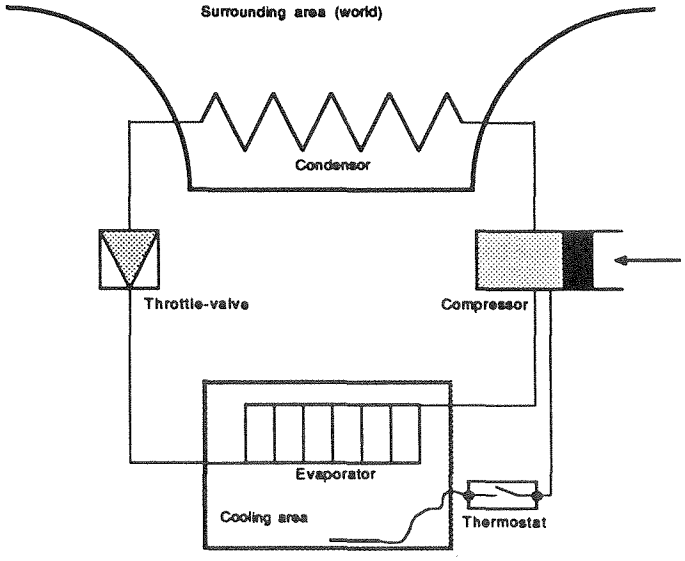

Figure 2: A model of the refrigerator

modelling device behaviour and processes between physical objects is required (see [2] for more details). Both the compressor and the throttle valve can be modelled as agent models that influence the *amount* of substance in the condensor and in the evaporator.

These substances can be modelled as 'closed contained substances' as described in the process centered approach (cf. [5]). Also the world and the cooling area can be modelled in this way, although the knowledge represented by these models may neglect the *pressure* and the *amount* of substance. A direct proportional relation between the *temperature* and the *heat* is sufficient to model the behaviour of these entities. Finally, the prediction model must include processes like heat flow, evaporation and condensation. The causal model (proportionalities and influences) that is represented by these partial behaviour models is shown in figure 3.

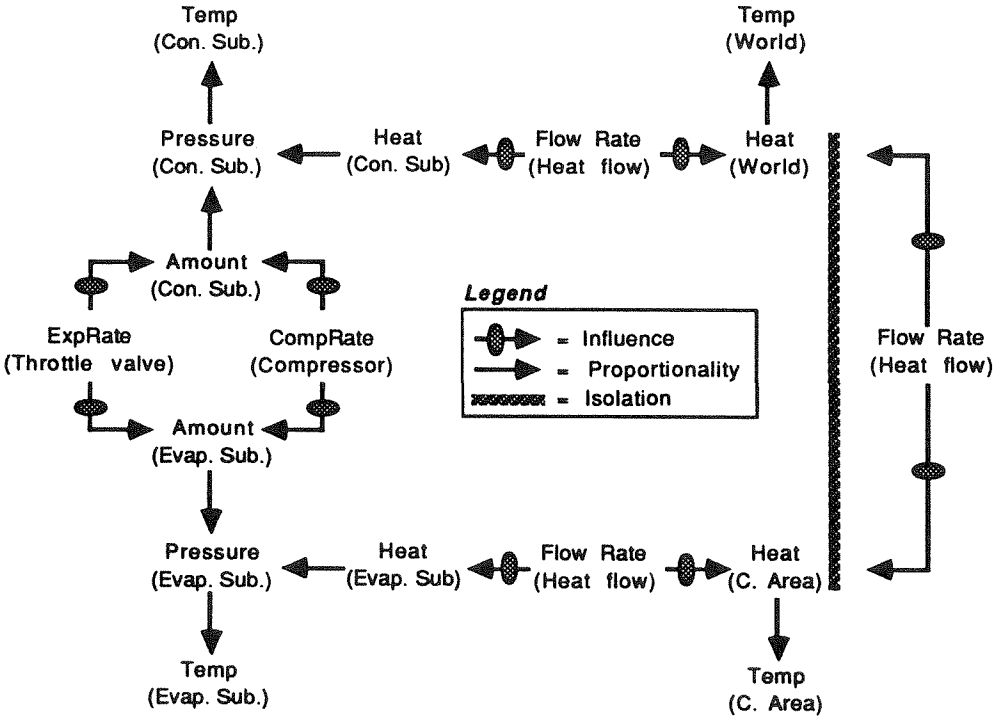

Figure 3: Causal dependencies in the refrigerator

Presenting this model to the prediction engine (GARP) results in 179 possible states of behaviour (total envisionment). The first specification step (finding all the sets of partial behaviour models that apply to the canonical problem description) produces already 15 possible states of behaviour. The values of the parameter derivatives in these states are shown in table 2.

Most of the 179 states represent incorrect behaviour of the refrigerator, i.e. states of behaviour that are qualitatively possible, but which do not represent actual behaviour of the refrigerator. A typical example of such behaviour is a heat flow from the surrounding world into the substance contained by the condensor which eventually results in an increase in the *temperature* of the cooling area. In a model representing correct behaviour of the refrigerator, no such heat flow should occur. This requires an extra constraint on the heat flow from the surrounding world to the substance in the condensor.

A large number of incorrect behaviours results from ambiguity, i.e. the relative impact of competing influences is not represented in the general library of partial behaviour models. Additional constraints are required which represent assembly specific behaviour and

Parameters	States														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Press *(Evap. Sub.)*	-	-	-	0	+	-	-	-	0	+	-	-	-	0	+
Amount *(Evap. Sub.)*	-	0	+	+	+	-	0	+	+	+	-	0	+	+	+
Heat *(Evap. Sub.)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Temp *(Evap. Sub.)*	-	-	-	0	+	-	-	-	0	+	-	-	-	0	+
Press *(Con. Sub.)*	-	-	-	-	-	0	0	0	0	0	+	+	+	+	+
Amount *(Con. Sub.)*	-	-	-	-	-	0	0	0	0	0	+	+	+	+	+
Heat *(Con. Sub.)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Temp *(Con. Sub.)*	-	-	-	-	-	0	0	0	0	0	+	+	+	+	+
Heat *(C. Area)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Temp *(C. Area)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Heat *(World)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Temp *(World)*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-Rate *(T-valve)*	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
C-Rate *(Compressjor)*	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 2: The first 15 states generated for the refrigerator (only 11 is correct)

thereby reduce the amount of ambiguity. Consider, for example, the *amount* of substance in the condensor. The condensor has a positive influence on this when it is working, but the throttle valve allows substance to flow out and consequently influences the *amount* negatively. The resulting derivative for the *amount* is ambiguous, i.e. it can increase (the effect of the compressor is greater), decrease (the out flow via the throttle valve is greater), or stay constant (the effects are equal). When the *amount* decreases this leads to a *pressure* and *temperature* decrease and therefore to a heat flow from the world into the condensor. This is an undesired sequence of behaviour. When the compressor is 'working' the *amount* of substance may only increase, or stay constant, but not decrease.

The problem is even worse because the influence of heat flow counters the effect of the *pressure* increase. Again there is ambiguity: the *temperature* may increase (the effect of the *pressure* increase is greater), decrease (the effect of the heat flow is greater), or stay constant (the effects are equal). However, as mentioned before, for a correct functioning refrigerator the effect of the *pressure* increase caused by a 'working' compressor is greater.

In both examples the specific configuration of physical objects is such that only certain behaviours take place. A behaviour prediction consisting of correct states of behaviours can only be obtained when the constraints for assembly specific behaviour are added to the knowledge in the (initial) library.

A second source for ambiguity is the lack of conservation constraints. Starting with the initial library there is no knowledge available that represents the conservation of quantities for a certain device. In the case of the refrigerator there is at least a need for a constraint that models conservation of substance between the condensor and the evaporator. When this constraint is lacking this means that the ambiguity discussed above for the condensor is mirrored for the evaporator, which results in 9 times 9 possible states of behaviour. Including behaviours in which both the *amount* of substance in the condensor and in the evaporator are increasing or decreasing (see for example state 1 in table 2). Conservation

of quantities is essential for disambiguating these faults in the prediction model.

5 Determining Candidate Constraints

This section describes how the constraints for disambiguation of the prediction model can be derived on the basis of feedback about correct and incorrect states of behaviour and the causality that underlies the model. The first section describes how candidate specialisations of the library in the form of extra constraints can be generated and the second section discusses how we can discriminate between competing constraints.

5.1 Generating Constraints

For generating constraints two sources of information are available: (1) the states of behaviour that represent correct behaviour and (2) the states of behaviour that represent incorrect behaviour. The conclusions that can be derived from these sets depend on the scope of the prediction. If the prediction is complete, i.e. all correct and all incorrect states have been generated, then the parameter derivatives can be used to generate sets of constraints that exclude all states of behaviour that have incorrect values for derivatives (table 3) and/or incorrect combinations of derivatives (table 4).

Derivative	Consistent constraints	Inconsistent constraints
−	1 ∨ 5	2 ∨ 3 ∨ 4
0	1 ∨ 2 ∨ 3	4 ∨ 5
+	2 ∨ 4	1 ∨ 3 ∨ 5

Table 3: Derivatives and related constraints

Generating a set of constraints is not self-evident, because there may be multiple (independent) causes responsible for the generation of an incorrect state of behaviour. The problem is to determine which constraints between which parameters should remove which states of behaviour. There is, for example, no way to decide whether the first incorrect state of behaviour (see table 2) is caused by lack of knowledge about order of magnitude (*amount* of substance in condensor always increases or stays constant, but never decreases) or by a lack of knowledge about conservation of quantities (the changes in the *amount* of substance in the condensor and in the evaporator should equal zero).

The process is even more complicated when the behaviour prediction is partial, which is usually the case. Although the reliability of the information stemming from the incorrect states becomes greater when more states of behaviour have been predicted, it is very likely that not all faults manifest themselves in a certain partial behaviour prediction. In other words, constraints which first seemed to represent a discriminating factor between correct and incorrect states may turn out to be inappropriate. The problem is not to select a constraint that will rule out a correct state of behaviour that was not predicted yet. Specifying, for example, that the derivative of the *temperature* of the substance in the condensor is always greater than the derivative of the *temperature* of the substance in the evaporator erroneously excludes future states of behaviour in which the two are equal.

In order to cope with the two problems described above we can focus primarily on the information that can be derived from the correct states of behaviour. The constraints that

Derivative pairs	Consistent constraints	Inconsistent constraints
-,-	1 ∨ 2 ∨ 3	4 ∨ 5 ∨ 6
-,0	1 ∨ 5	2 ∨ 3 ∨ 4 ∨ 6
-,+	1 ∨ 5 ∨ 6	2 ∨ 3 ∨ 4
0,-	2 ∨ 4	1 ∨ 3 ∨ 5 ∨ 6
0,0	1 ∨ 2 ∨ 3 ∨ 6	4 ∨ 5
0,+	1 ∨ 5	2 ∨ 3 ∨ 4 ∨ 6
+,-	2 ∨ 4 ∨ 6	1 ∨ 3 ∨ 5
+,0	2 ∨ 4	1 ∨ 3 ∨ 5 ∨ 6
+,+	1 ∨ 2 ∨ 3	4 ∨ 5 ∨ 6

Table 4: Derivative combinations and related constraints

exclude incorrect states of behaviour should be consistent with the information captured in the correct states of behaviour. In other words:

- *the constraints that exclude incorrect states of behaviour should be in the set of all the constraints that are consistent with the derivatives in the known correct states of behaviour.*

There are two potential problems to this approach. Firstly, the set of parameters describing a state of behaviour should not be too large, in order for the set of possible constraints to explode, and secondly, there is a danger of proposing dependencies between parameters which are false, because these parameters are essentially unrelated. The problem of proposing relations between unrelated parameters could lead to exclusion of correct states of behaviour. However, the selection procedure, discussed below, is based on the causality between parameters and therefore automatically prevents the selection of constraints between unrelated parameters.

The generation of constraints consists of the identification of three types of constraints:

1. The derivative of a certain parameter may only have specific values.
2. The derivatives of two parameters must have corresponding values.
3. The derivatives of certain parameters may be limited, because of conservation of quantities.

The constraints for single derivatives are generated according to table 3. For each parameter all constraints are found that are consistent with all derivatives of this parameter in all the correct states of behaviour. If, for example, a parameter has derivatives + or 0 in the correct states of behaviour then constraints 2 ∨ 4 and 1 ∨ 2 ∨ 3 are consistent with these derivatives. Constraint 2 is consistent with both derivatives and is therefore a candidate for disambiguating the behaviour prediction. In our example (see table 2) there is one correct state of behaviour (state 11) and 14 parameters (2 with unknown derivatives, which therefore may not be used). Using table 3 this initially results in 30 candidate constraints for disambiguation of the prediction.

The constraints for related derivatives are generated according to table 4. For each pair of parameters all constraints are found that are consistent with all the derivatives of

parameters in the correct states of behaviour. If, for example, a pair of parameters has the derivatives $(+,+)$ or $(-,0)$ in the correct states of behaviour, then the constraints $1 \vee 2 \vee 3$ and $1 \vee 5$ are consistent with these derivatives. Constraint 1 is consistent with both pairs of derivatives. In the refrigerator example, the correct state of behaviour facilitates 140 candidate constraints between pairs of parameter derivatives.

Generating constraints for conservation of quantities can be guided by knowledge about the causal relations between parameters in order to limit the number of constraints that will be found. The idea is that dependencies between parameters can be factored into clusters that influence each other, but are independent of other parameters.[4] We shall call these clusters *causal units*. A causal unit starts with a parameter that is being influenced (by an influence relation), traverses via the proportionally related parameters, and ends with a parameter that has no causal effect on any other parameter by means of a proportionality. A causal unit is essentially a graph that may have recursive loops, more than one starting point and more than one terminal node (notice that a causal unit can consist of one or more causal paths).[5] In our example of the refrigerator (see figure 3) there are 6 causal units. Two of those are:

Heat *(Con. Sub.)* → Press *(Con. Sub.)* → Temp *(Con. Sub.)*
Amount *(Con. Sub.)* → Press *(Con. Sub.)* → Temp *(Con. Sub.)*

Constraints for conservation of quantities may be defined between parameters that: (1) belong to different causal units, (2) model the same type of quantity, and (3) are influenced by the same influence. Usually an influence (c.q. a flow rate) consists of two parts, one that decreases a quantity and one that increases a quantity. Both are required before a conservation constraint may be defined. In the case of the refrigerator conservation constraints are generated between the parameters:

Amount *(Con. Sub.)* & Amount *(Evap. Sub.)*
Heat *(Con. Sub.)* & Heat *(World)*
Heat *(Evap. Sub.)* & Heat *(C. Area)*

The model does not allow a heat flow between the cooling area and the world, otherwise the conservation constraint for *heat(s)* would have included the four *heat* parameters in a *single* conservation constraint.

5.2 Selection of Appropriate Constraints

The generation of constraints as described above results in 173 candidate constraints for disambiguation of the prediction model. A number of rules can be defined to discriminate between these constraints.

Remove non-discriminative constraints Constraints that do not discriminate between correct and incorrect states of behaviour can be disregarded. This rule applies for each of the three types of constraints discussed above. Usually, non discriminative constraints result from equal derivatives in both correct and incorrect states of

[4] This corresponds to the idea of 'factoring', see [6].

[5] We differ from Forbus [5] who does not allow a parameter to be influenced both directly and indirectly. However, the specific choice in this respect has no effect on the technique presented in this paper.

behaviour. For example, specifying that the *heat* of the world equals zero does not discriminate between correct and incorrect states of behaviour.

Prefer weaker constraints above stronger constraints d_greater_or_equal is a weaker constraint (allowing more values) than either greater or equal. In general, combinations are weaker, and should be preferred. Therefore, if a parameter (or a pair of parameters) has both the stronger and the weaker constraint then the former must be removed. This rule prohibits that on the basis of incomplete information a too restrictive constraint is selected. The more restrictive constraints will only be selected after the weaker constraints fail to exclude incorrect states of behaviour. This corresponds to a search strategy in the search for specialisations from general to specific. The *prefer weaker* rule does not effect conservation constraints because they are always of type equal.

These two rules remove a large number of the constraints that were proposed during the generation step. In fact only 54 constraints remain after applying these two rules: 6 on specific derivatives, 47 concerned with pairs of derivatives, and 1 conservation constraint.

It should not come as a surprise that most of the removed constraints are concerned with specific derivatives and with pairs of derivatives. In contrast to the generation of conservation constraints the generation of these constraints is not guided by any specific knowledge. For efficiency reasons we may of course decide to make the generation more knowledge intensive by including the above two rules in the generation step and directly limit the number of constraints being generated.

The next set of rules, that can be used to select among competing candidates, is based on the notion of causal units in the domain model (see also figure 3). In general we want to (1) remove constraints between causally independent parameters, (2) define constraints on parameters that directly effect each other (i.e. adjacent in the causal unit), and (3) constrain parameters 'early' in the causal unit.

Remove causally independent constraints Remove all constraints proposing relations between parameters that are not part of a single causal unit. Ambiguity always results from multiple influences (proportionalities and/or influences) on a certain parameter, therefore, the disambiguation has to effect the parameters that are related in this way. For example, a constraint between the *temperature* of the substance in the evaporator and the *pressure* of the substance in the condensor is not allowed. This rule effects the constraints between pairs of parameter derivatives and has an important impact. It removes 41 of the candidate constraints between pairs.

Prefer adjacent constraints above intermediated constraints Prefer constraints between adjacent parameters in a causal unit above constraints that are related via one or more intermediate parameters. If, for example, a constraint is proposed both between the *amount* & the *pressure* and between the *amount* & the *temperature* of the substance in the condensor, then the former should be preferred (see also figure 3). The rationale behind this rule is that the disambiguation should propose constraints that follow the causal dependencies in a causal unit. This rule only effects constraints between pairs of parameter derivatives. In the example it removes 3 of the 6 remaining constraints, as for instance, preferring the constraint between the

amount & the *pressure* above the constraint between the *amount* & the *temperature* of the substance in the evaporator.

Prefer constraints on early parameters in the causal unit This rule applies to all three types of constraints. The rationale behind this rule is that the disambiguation should start at the beginning of the causal dependencies. In the case of pairs of derivatives it should prefer constraints between adjacent parameters that appear early in the causal units above constraints between adjacent parameters that are positioned later in this unit. If, for example, a constraint is proposed both between the *amount* & the *pressure* and between the *pressure* & the *temperature* of the substance in the condensor, then the former should be preferred.

This rule can also be used for constraints on a single parameter derivative. A constraint on the derivative of the *amount*, for example, should be preferred above a constraint on the *temperature*. In the refrigerator example, this rule removes 4 constraints on specific parameters by preferring constraints on the *amount* above constraints on the *pressure* and the *temperature* for both the substance in the condensor and in the evaporator.

In addition to pairs and single parameter derivatives this rule can also be applied for conservation constraints: if there exists more than one conservation of quantity constraint between quantities belonging to different causal units and the causal path is similar for these units (within the range of these constraints) then the constraint between the parameters earlier in the causal path should be preferred. However, we could not think of an example for the refrigerator fitting this rule. It seems that this case is impossible by definition.

The set of constraints that remains after all the above rules have been applied is shown in table 5. The set can be divided into two parts, one concerned with conservation of quantities and one with order of magnitudes (or assembly specific behaviour). The constraints on specific derivatives and between pairs of derivatives belong to the latter.

Constraints	**Eliminated states**
Constraint for conservation	
d_equal(zero, plus(Amount-ConSub, Amount-EvapSub))	1, 2, 6, 8, 9, 10, 12, 13, 14, 15
Constraints on derivative pairs	
d_greater_or_equal(Press-EvapSub, Amount-EvapSub)	2, 3, 4, 7, 8, 9, 12, 13, 14
d_smaller_or_equal(Pressure-EvapSub, Heat-EvapSub)	5, 10, 15
d_smaller_or_equal(Heat-ConSub, Press-ConSub)	1, 2, 3, 4, 5
Constraints on specific derivatives	
d_greater_or_equal(Amount-ConSub, zero)	1, 2, 3, 4, 5
d_smaller_or_equal(Amount-EvapSub, zero)	3, 4, 5, 8, 9, 10, 13, 14, 15

Table 5: Remaining constraints with the states that are eliminated.

Thus far, we have not been concerned with the number of incorrect states that is being excluded. The rules are only based on the causality that has been represented in prediction model. The constraints that remain after applying the above described rules cannot be

discriminated any further in this respect. For deciding upon which constraints must be added to the knowledge in the library, the following aspects are important:

Independent constraints Constraints that only exclude states of behaviour that are not excluded by any other constraints are necessary for the disambiguation of the model and must be added to the library knowledge. However, there are no independent constraints in table 5.

Overlapping constraints Overlapping constraints cannot be discriminated any further, except for the notion that the 'super' constraint excludes the most states of incorrect behaviour. It seems reasonable to prefer this constraint. In table 5 this would imply that the constraints between the *pressure* and the *heat* (both for the substance in the condensor and for the substance in the evaporator should be removed in favour of the constraints on the derivatives of the *amounts*

Partially overlapping constraints Partially overlapping constraints that exclude some states of behaviour that are not excluded by any other constraints, but they also exclude similar states of behaviour. These constraints are necessary for the disambiguation of the model and must be added to the library knowledge. In table 5 the conservation constraint, the two constraints on the derivatives of the *amount* of substance in the condensor and the *amount* of substance in the evaporator, and the constraint between the *pressure* and the *amount* of substance in the evaporator, are partially overlapping. All four constraints propose constraints that remove states of behaviour that are not removed by the other constraints.

A possible heuristic for dealing with partially overlapping constraints is the following:

- Prefer conservation constraints above constraints between pairs of parameter derivatives and constraints on a single parameter derivative.
- Prefer constraints between pairs of parameter derivatives above constraints on a single parameter derivative.

The order introduced by this heuristic is based on the idea that more complex constraints are more likely to be correct than less complex constraints. However, in the case of a partial behaviour prediction the only safe way is to have the knowledge engineer decide among the final set of competing constraints.

In the case of a full behaviour prediction overlapping constraints introduce no problem because all correct states of behaviour are facilitated by the constraints and they can thus simply be added to the model. Still it is impressive to discover how even a small set of correct states (in our example just 1) provides sufficient information for the above described technique to generate exactly those constraints that fully disambiguated the prediction model (see table 6).

6 Localise Partial Behaviour Model

After the constraints for disambiguation on the behaviour prediction have been determined, the next step is to determine the place in the library where each of the constraints

Constraints	Eliminated states
d_equal(zero, plus(Amount-ConSub, Amount-EvapSub))	1, 2, 6, 8, 9, 10, 12, 13, 14, 15
d_greater_or_equal(Press-EvapSub, Amount-EvapSub)	2, 3, 4, 7, 8, 9, 12, 13, 14
d_greater_or_equal(Amount-ConSub, zero)	1, 2, 3, 4, 5
d_smaller_or_equal(Amount-EvapSub, zero)	3, 4, 5, 8, 9, 10, 13, 14, 15

Table 6: Resulting set of constraints that fully disambiguates the prediction model

should be added. Essentially two ways of augmenting the library are possible: either the constraints are added to an already existing partial behaviour model, or the constraints have to be represented in a new behaviour model. The method for localising the partial behaviour model is similar in both cases.

Firstly, the dependencies that the parameters in the proposed constraint have with existing influence relations and proportionality relations have to be determined. Each of these relations contributes to the ambiguity that has to be removed by the constraint. The behaviour models that introduce these relations are therefore conditional for the behaviour model to which the new constraint has to be added. If, for example, a constraint has to be added between the *amount* and the *pressure* of the substance in the condensor then (1) the influence of the *compression rate*, (2) the influence of the *expansion rate*, (3) the proportional relation with the *heat* and consequently (4) the influence of the *flow rate* (from the heat flow) are contributors to the ambiguity that will be reduced by this constraint (see also figure 3). The notion of causal units is again important, in the sense that the set of contributing relations is a subset of the causal units to which the parameters belong. In particular, this subset starts with the influencing relations, moves on via proportionalities up to the place where the parameters themselves are located. The relations located higher in the causal unit do not contribute to the ambiguity.

Having found the set of relations that contributes to the ambiguity the second step is to find the set of behaviour models that introduced these relations. In the example mentioned above these models are: (1) the heat flow process, (2) the model of the active compressor, (3) the model of the active throttle valve, and (4) the behaviour model for the closed contained substance (=the condensor). In order to place the new constraint there must be either a partial behaviour model that has this list of behaviour models as a condition or a new assembly has to be created. In the latter case, which is more likely, the conditional partial behaviour models constitute the aggregate that introduces the disambiguating constraint. In the above example this assembly could be referred to as the 'condensing assembly' of the refrigerator.

7 Further Specification

Instead of immediately adding all constraints that have been proposed by the technique discussed above to the library, the behaviour prediction can be enlarged in order to create more discriminative power for grounding the constraints. In our example we could have added one transformation step which would have created an initial set of 38 states of behaviour.

However, also if one or more of the candidate constraints has been added to the library

it may still be necessary to further specify the knowledge in the library. There are in fact three reasons why this may be the case: (1) stronger constraints for already constraint parameter(s), (2) constraints for parameter(s) that did not behave incorrectly yet, and (3) constraints for newly introduced parameter(s).

It is very likely that weak constraints added to the library knowledge eventually must be replaced by stronger ones. Also it is to be expected that parameters that appeared to be correct, start behaving in ways that are incorrect and/or undesired. So further prediction is required to find these parameters. Finally, it may be the case that some behaviour model introduces parameters that were not in the behaviour prediction yet. Required constraints on these parameters must also be found.

8 Concluding Remarks

In this paper we have presented a technique for automated generation of the constraints that are needed for disambiguation of a behaviour prediction. Starting with a behaviour prediction using an initial library containing general domain knowledge the technique employs feedback about correct and incorrect states of behaviour and knowledge about the causal dependencies between the parameters in the model in order to determine the constraints that remove the incorrect or undesired states of behaviour that result from ambiguity. In addition, the technique points out the assembly of physical objects to which the constraints apply.

The work that is most similar to this is by Mozetic (e.g. [7]). By representing models in a logical language the refinement problem becomes similar to refinement of logic programs. To our knowledge there has been no previous work on refining knowledge that is represented as qualitative constraints. DeJong's work ([4]) focuses on explanation-based learning in the context of plausible reasoning rather than refinement.

The refinement algorithm presented in this paper is similar to general incremental learning techniques. It is special in the representation and inference engine, an important addition is the use of factoring into causal units and the learning bias based on the causal structure and the global conservation constraints.

The scope of our technique is limited in that only deals with derivatives. Incorrect states of behaviour resulting from parameter values and relations between these values cannot be dealt with (although it seems likely that parts of the technique can be used for this purpose as well). Also the technique requires that the knowledge present in the initial library is correct and sufficient, i.e. it should at least predict all possible states of behaviour.

However, these problems are not limitations of the technique. Instead, they refer to different aspects of the modelling process that must be dealt with in order to further automate the process of qualitative model construction. The technique for generating the disambiguating constraints for parameter derivatives presents an important step in this direction.

References

[1] D.G. Bobrow, editor. *Qualitative Reasoning about Physical Systems.* Elseviers Science

Publishers B.V., Amsterdam, 1984.

[2] B. Bredeweg. *Expertise in qualitative prediction of behaviour.* PhD thesis, University of Amsterdam, Amsterdam, March 1992.

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

[4] G. DeJong. Explanation-based learning with plausible inferencing. In K. Morik, editor, *Proceedings European Working Session on Learning EWSL-89*, pages 1–10. Pitman, 1989.

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

[6] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, Los Altos, California, 1987.

[7] I. Mozetič. The role of abstractions in learning qualitative models. In *Proceedings of the 4th International Workshop on Machine Learning*, pages 242–255, Irvine, 1987. University of California.

[8] O. Raiman. Order of magnitude reasoning. In *Proceedings of the AAAI*, pages 100–104, San Mateo, California, 1986. Morgan Kaufmann.

[9] D.S. Weld and J. De Kleer. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann Publishers Inc., 1990.

Acknowledgements

This research was supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organisation for Scientific Research (NWO) (project number: 612-322-016).