Chapter 4

Qualitative Prediction of Behaviour

This chapter describes an integrated conceptual framework for qualitative prediction of behaviour (cf. [20; 15; 17]). In order to clarify the boundaries between prediction and other related tasks, such as modelling, the first section classifies the tasks relevant to behaviour prediction according to the criteria proposed by the theory discussed in the previous chapter. Having set these boundaries the description of the four layer model for qualitative prediction of behaviour is presented. In particular, the discussion of the framework focuses on an extended world view for representing partial behaviour models, the representation of parameter specific quantity spaces, an integrated set of parameter relations with additional functionality for causal value correspondence, and a transformation step between states of behaviour that encorporates an explicit selection and ordering of possible terminations.

Having described the problem solving roles (meta classes) in detail, the canonical inferences, from a conceptual point of view, turn out to be relatively straightforward. However, realising their problem solving potential in a computer program is a complex matter. The algorithms developed for that purpose are discussed in the next chapter which describes the design and implementation of the conceptual model.

4.1 Three Basic Tasks

The theory of problem solving, described in the previous chapter, proposes a number of criteria for distinguishing between problem solving tasks. Tasks can be classified according to the *goal* they realise, the *input* they require, and the *output* they produce. In addition, we take into account which tasks can be carried out independently from other tasks by a single *agent*. Together these criteria lead to three global tasks that are relevant for qualitative prediction of behaviour: *modelling*, *prediction*, and *interpretation*.

4.1.1 The Modelling Task

Modelling refers to all the work that must be done by a knowledge engineer in order to provide a qualitative inference engine with a model of some real-world system that it can use for behaviour prediction. Realising the goal of the modelling task requires the following two outputs to be produced:

- a framework that specifies the knowledge representation and the related reasoning techniques that the qualitative reasoner may use, and
- a description of the real-world system, that is going to be the object of the behaviour prediction, in the language defined by the framework.

Developing a *prediction model* necessarily starts with formulating the overall framework for representing the knowledge relevant to the prediction task. During this *generic* modelling step decisions have to be made concerning the knowledge representations and the corresponding reasoning techniques that will constitute the specific approach to qualitative reasoning. Modelling systems from the real-world into the framework can be further divided into:

- modelling the general knowledge about the physical world, and
- representing the specific system that is the object of behaviour prediction.

General knowledge about the physical world is usually related to certain domains, such as thermodynamics or mechanics. The purpose of the *domain* modelling is to represent the general knowledge relevant to such a domain. The *case* modelling concerns representing the specific system that is input for the prediction task. The three modelling tasks are depicted in figure 4.1.

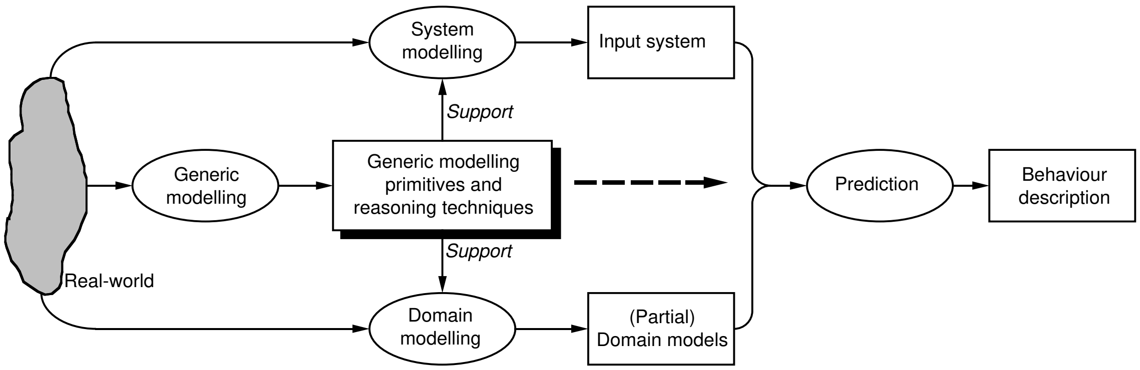

Figure 4.1: Modelling activities related to qualitative reasoning

The output of generic modelling together with the domain modelling is what Steels (cf. [123]) defines as the *domain model*, that is a model built of some part of the real-world that encompasses a certain view on that world.

There is a tradeoff between case and domain modelling in the sense that the complexity of the case modelling depends on how much general knowledge has been represented during the domain modelling. In the component and process centred approaches a library is used to store the general domain knowledge that can be applied to partial descriptions of systems relevant within that domain. The assumption behind this technique is that a sufficient amount of domain modelling will simplify the complexity of the case modelling to only providing the qualitative inference engine with a description of the physical objects and their structural relations (=structural description). The qualitative inference engine will then derive all the knowledge needed for behaviour prediction from the domain knowledge library. This technique, which has been defined as 'deriving behaviour of a device from a description of its physical structure', must deal with the problem that a single structure may manifest different behaviours depending on the specific context in which it is situated. The predicted behaviour should only capture those behaviours that are relevant to that context. Reasoning about the behaviour of a refrigerator, for example, focuses on different properties depending on whether we want to understand the *substance-flow* of the refrigerant, or whether we want to understand how the *heat-exchange* between the refrigerator and its environment, effects this environment (cf. [61]).

An interesting enhancement in this respect is the composite modelling proposed by Forbus [67; 68]. This technique generates a behaviour prediction that is specific for answering a certain question about the behaviour of a system. Instead of using all the knowledge that is in principle available (as a result of the domain modelling), the qualitative inference engine only considers those parts of the knowledge that are relevant for answering the question. The question focuses the search for possible behaviours.

However, the problem is a fundamental one: it is not possible to anticipate all behaviours that *any* structure may manifest at *any* point in time. Therefore the library of domain knowledge will never be complete. To cope with this problem we could abandon the notion of modelling general knowledge in a library all together and model all the knowledge needed for a certain behaviour prediction in the input system. This basically matches the constraint centred approach where all the knowledge must be presented in the input system. From our point of view this is an undesired solution to the problem, for two reasons:

- There exists general knowledge which can be applied for a whole range of prediction problems. Not representing this knowledge unnecessarily introduces redundant modelling for new problems.

- Representing general domain knowledge provides support for how new problems can be formulated and solved. Not representing this general knowledge also means that this support cannot be given.

Modelling is one of the major problems in artificial intelligence. An approach that provides some guidelines for supporting the modelling process should be favoured above one that gives no support at all. In our framework we therefore use a library that represents the general knowledge about the entities and processes in the physical domain. However, additional knowledge, relevant to the behaviour prediction, can be modelled in the input system. The notion of input system is further discussed in section 4.2.1.13.

4.1.2 The Prediction Task

Prediction of behaviour is the central problem solving task in qualitative reasoning. Given a partial description of some aspect of the real-world, the goal is to identify qualitative distinct states of behaviour that the system can reach in the course of time, given the constraints and laws that govern behaviour in the physical world. The most general model for qualitative reasoning about the physical world consists of two main reasoning steps. The first step takes as input a partial description of some situation and produces as output a full description of the physical system in terms of qualitative properties. This step is similar to the traditional form of solving physics problems where we are given a partial description of some situation in the physical world and where the solution consists of determining one or more values of state variables (cf. [32; 105]). The second reasoning step determines possible transitions from the computed state to other states that the system may have. This step constructs the envisionment of the behaviour of the system in time. Figure 4.2 shows the inference structure of this reasoning process.

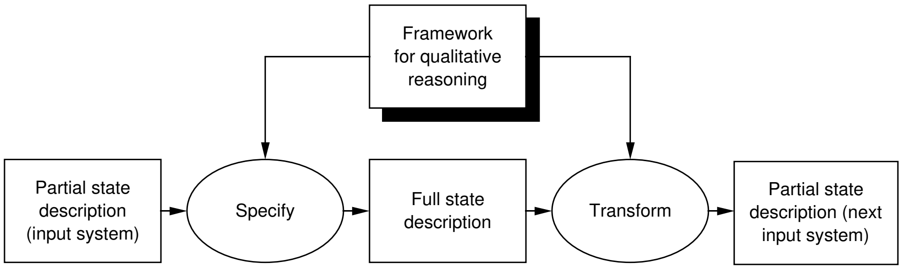

Figure 4.2: Global inference structure for behaviour analysis

Control of the inference steps can take several forms. Given a partial description of a system it is possible to generate all instances of states that are consistent with that input description. Alternatively, one can choose to generate a most plausible full state description and only consider other states when necessary.

This model of qualitative reasoning not only captures the various approaches to qualitative reasoning, but also models reasoning about behaviour of systems based on explicit causal models. For example, if one wants to predict the development of an illness of a patient over time, the first step is to establish a full account of the current state of the patient in terms of the causes of the illness and its progression. Subsequently, possible transitions to new qualitative states of behaviour are determined on the basis of the causal network. Qualitative reasoning can be viewed as a specific case of reasoning about states and state transitions in complex physical systems. The prediction task is further discussed in section 4.2.

4.1.3 The Interpretation Task

The interpretation task refers to the work that needs to be done by some agent, after the prediction task is completed, in order to analyse the output of the qualitative inference engine and classify the produced output in terms of behaviour classes. Typically, this agent should identify forms of recursive behaviour, such as oscillation, and point out paths of causal behaviour propagation. More generally it should classify the output of a qualitative inference engine in terms of the requirements set by another problem solver (see also figure 4.3). Take for example, a diagnostic inference engine that has to find the causes that explain the malfunctioning of a device. The malfunctioning is observed by comparing the parameter values of the real-world system with those predicted on the basis of a model of that system [59]. Given a discrepancy between the predicted and the observed values it is the task of the diagnoser to identify the physical objects that produced this discrepancy. In order to establish such a diagnosis, the diagnostic inference engine needs a causal account in terms of which elements from the system contribute to each of the parameter values. Instead of directly presenting the states of behaviour produced

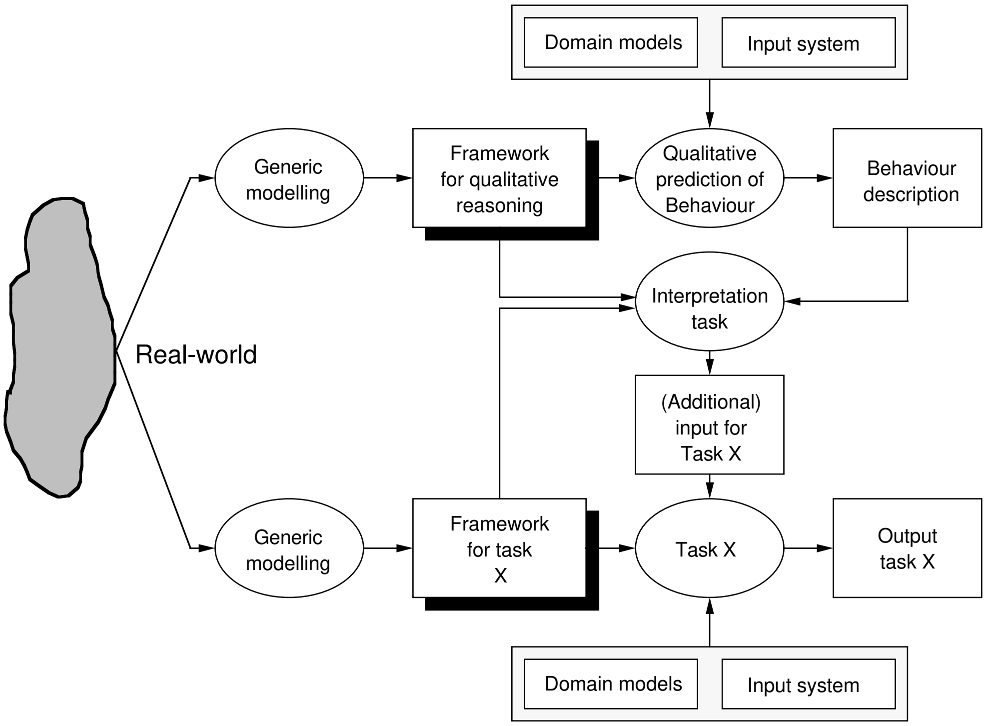

Figure 4.3: Visualising the interpretation task

by the qualitative inference engine to the diagnoser, the interpretation task modifies the output of the qualitative inference engine such that for each parameter value it is known which element from the system contributed to that value.

There is an interaction between the framework used for prediction and the interpretations that can be derived from the output (cf. [21]). If, for example, the physical objects are not represented explicitly in the prediction model, then it is impossible to provide the causal account discussed above.

Special attention in this respect has been given to producing causal interpretations of the behaviour predicted by a qualitative inference engine. The process centred approach uses indirect and direct causal dependencies for deriving the behaviour of some system. A causal account of the predicted behaviour is therefore a relatively straightforward trace of the influences and the proportionality relations that were used. The dependencies between parameter values in the component centred approach are not directed. Moreover, a generate and test method is used for deriving new values when the confluences are underspecified and are not capable of inferring the next value (see also section 2.2.1.6). The problem is that there is no knowledge present in the confluences for determining which value to focus on. Therefore, a causal account of the produced behaviour is not available.

Solutions, such as, *mythical causality* [57] and *causal ordering* [87; 58; 88] have been proposed to provide causal explanations for this reasoning technique. Mythical causality uses heuristics for guiding the generation of new values. The causal knowledge is captured in these heuristics. Causal ordering tries to generate a dependency tree of values and constraints such that each subset is a necessary requirement for its super set (=self contained). The subset can be regarded as a precondition for the values derived in the superset and therefore the superset values are 'caused-by' the subset.

Recently, the use of *bond-graphs* has been proposed as a further enhancement for generating causal interpretations [131; 132]. This technique differs from the above two in that it uses knowledge about physics as a guideline for building models. The physical laws represented in the bond-graphs provide specific constraints about how the physical world must be modelled. A specific set of physical mechanisms is used for modelling physical behaviour and applied to abstractions of the physical world. As a result the approach does not derive behaviour directly from the physical structure. In comparison with causal ordering, the causal interpretations that can be derived from bond-graphs provide more information.

Additional aspects of the interpretation task will be discussed in chapter 7.

4.2 Conceptual Model for Prediction of Behaviour

This section describes an integrated conceptual framework for qualitative prediction of behaviour, based on the three original approaches to qualitative reasoning (section 2.2). The theory of problem solving defined by the *KADS* methodology (section 3.1) provides a framework that distinguishes between domain, inference, task and strategic knowledge. This four layer knowledge typing, together with the modelling primitives provided at each layer, is used as a method for integration. In addition it provides the basis for pointing out similarities and differences between the approaches.

The description of the conceptual model starts with the roles that the domain knowledge plays in the reasoning process. We then move on by describing the canonical inferences that can be made on the basis of these knowledge roles. The control of the inference process, in terms of task and strategic knowledge, is described in the third section.

4.2.1 Meta Classes: Roles Played by the Domain Knowledge

The approaches to qualitative reasoning provide different sets of ontological primitives for modelling domain specific knowledge. Typical examples are: components, qualitative states, views, processes, influences, and qualitative differential equations. Each modelling primitive is used by the qualitative inference engine in a certain way. This use represents the *role* that the modelling primitive plays in the reasoning process.

It is unclear how the modelling primitives of the different approaches compare with each other. In particular, the role of the domain knowledge is often not separated from the modelling primitive used to represent that knowledge. *KADS* tackles this problem by *explicitly* distinguishing knowledge representation at the domain layer from a description of how this knowledge is used by the inference layer. In the following section we will investigate which knowledge roles are relevant to qualitative prediction of behaviour (an overview of all the knowledge roles can be found in table 4.10). In particular, we will focus on:

- how the physical world is represented (system elements),
- what is considered as behaviour (parameters, values and relations), and

- how knowledge about behaviour is represented (partial behaviour models, system model descriptions and transformation rules)

In addition, we will discuss to what extent each approach to qualitative reasoning uses these knowledge roles.

4.2.1.1 System Elements

An essential step in qualitative reasoning is to determine

- how objects from the real-world are represented in the prediction model, and
- how these representations are applied to guide the behaviour analysis.

The domain knowledge that is used for this purpose performs the role of system elements.

The representation of the physical reality can provide an important focus for modelling the real-world behaviour. Two types of abstraction have been proposed:

- modelling the physical world as components connected by conduits, and
- modelling the physical world as physical objects that interact via processes.

Each of these abstractions provides specific guidelines according to which the real-world must be modelled. These guidelines can in addition be used as handles for developing general purpose libraries. As soon as a model has been constructed of the behaviour of a certain abstraction from the physical world (system element), this behaviour model can be stored in a library and used again in new situations. Both the library of component models and the library of views and processes are based on this principle. A disadvantage of using a specific world view is that the reasoning capabilities are necessarily limited to the primitives defined by the abstraction. A component model, for example, excludes reasoning about processes. The constraint centred approach has no commitment to a specific abstraction of the physical world.

As mentioned before, modelling is one of the major problems in artificial intelligence and a world view that provides modelling guidelines is important. However, as discussed in sections 2.3.1 and 2.3.2, the world views underlying the component and process centred approaches are limited and, depending on the kind of behaviour that has to be analysed, may turn out to be problematic. We present an alternative abstraction from the physical reality that overcomes these shortcomings. Similar to the component and process centred approaches we represent knowledge about real-world entities in small units (partial models) which must be aggregated into larger models for understanding the behaviour of the system as a whole. In addition, our world view is based the following requirements:

- It is essential to use both component and process oriented abstractions in a single prediction model.
- System elements in the prediction model may be functional abstractions of the physical reality and as such do not have to map directly onto physical objects.
- The *no-function-in-structure* principle applies to behaviour models of single system elements. Behaviour models defined for aggregates may refer to the behaviour defined in the models for single system elements that constitute the aggregate.

The types of (partial) behaviour models that can be constructed on the basis of this world view are discussed further in section 4.2.1.6. Below we describe the three types of system elements (see table 4.1) that are needed for representing this integrated and extended world view.

System element facets	**Description of problem solving role**
Generic concept	A hierarchy of generic descriptions used to focus the reasoning on certain entities of the real-world.
Instances	Instances of concepts from the hierarchy used for specifying the real-world objects that the qualitative inference engine reasons about.
Relations	Relations between instances of concepts used for for specifying the structural dependencies that exist between the real-world objects.

Table 4.1: Three types of system elements

A hierarchy of generic concepts, as for example depicted in figure 4.4, provides the qualitative inference engine with a description of the entities that are believed to be present in the real-world. A system that is the object of behaviour prediction consists of a subset

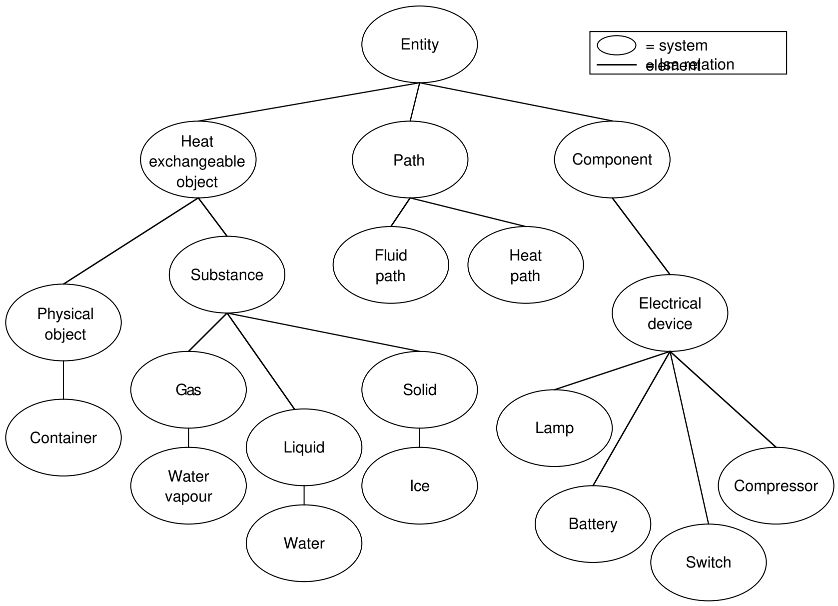

Figure 4.4: A simple hierarchy of generic concepts

of the entities defined in the hierarchy. The hierarchy gives information about which behaviour models have to be considered for behaviour prediction. If, for example, the system that is reasoned about consists of *liquid* then the qualitative inference engine may look for knowledge about *substances* and *heat-exchangeable-objects*. Components are just another group of elements in the hierarchy. Notice that system elements may also refer to functional abstractions, such as *heat-exchangeable-objects* and *heat-paths*.

Instances refer to the specific entities that constitute the system that is the object of behaviour prediction. Instances are always related to elements in the hierarchy. They specify which elements must be used for assembling the partial behaviour models in order to understand the overall behaviour of the system.

Relations are used for modelling structural dependencies between system elements, for example: a container *containing* liquid or two heat-exchangeable objects having a *heat-path connection.*[1] We do not distinguish between generic relations and instances thereof. Relations are always 'instances' and specific for certain instantiated system elements.

4.2.1.2 Parameters

Qualitative prediction of behaviour is particularly concerned with reasoning about the properties of the physical world that gradually change over time. The domain knowledge that represents this class of properties plays the role of parameters. Table 4.2 shows which aspects are relevant for modelling the role of parameters. It is important to distinguish

Parameter facets	Description of problem solving role
Generic name	A generic name of the parameter referring to a *physical quantity.* For example: temperature, volume and heat.
System element	A reference to the system element whose property is described by the parameter.
Instance name	A unique name for the parameter.
Type	Refers to the kind of values a parameter can have. The type can be *discrete*, *qualitative* or *quantitative.*
Quantity space	The values that a parameter can have.

Table 4.2: Modelling the role of parameters

between a parameter and its value, because the possible values that a parameter can have differ depending on the type of the parameter.

4.2.1.3 Parameter Values

The role of parameter values is to represent the specific value a parameter has. As mentioned before, the kind of values that a parameter can have depend on the type of a parameter. The parameter types represent different views on how parameters reflect the properties of system elements in time. They can be defined as follows:

Quantitative This type refers to describing the value of a parameter in real time, i.e. at each moment in time the value of a parameter can be described with a quantitative measure. Differential equations can be used for reasoning about the change of parameter values in time. Traditional physics uses quantitative parameter values.

Qualitative This is how parameter values are usually described in qualitative reasoning. The quantitative values of a parameter are abstracted into a set of ordered intervals.

[1] Relations should not be confused with parameters which describe properties of system elements that *continuously* change over time. These are discussed in section 4.2.1.2.

This abstraction corresponds to an abstraction of real time into adjacent intervals. At each time interval the parameter has a value from this ordered set of intervals. In addition, the derivative of the parameter represents how the quantitative value of the parameter changes in real time (*decreasing*, *steady* and *increasing*). For example the value of parameter *volume* is *plus* and *increases* (see figure 4.5).

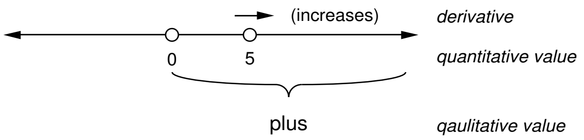

Figure 4.5: Parameter values

Discrete The granularity in representing time for this parameter type is such that the value of a parameter changes immediately at a certain point in time. In other words, the parameter does not change continuously, but jumps from one value to another as the system changes its behaviour. The exact quantitative value may be known, but the value may also refer to some set of discrete intervals. A typical example is a switch whose parameter *status* can have the values *on* and *off*.

Given this classification of parameter types the knowledge required for modelling parameter values can be defined as shown in table 4.3.

Parameter value facets	**Description of problem solving role**
Parameter reference	A reference specifying to which parameter the value belongs. This is the instance name of the parameter.
Quantitative value	The quantitative value a parameter has.
Qualitative value	The qualitative value a parameter has. The actual quantitative value is somewhere on this interval (or point).
Derivative	The derivative represents how the quantitative value changes over time. As mentioned before, it can be either increasing (*plus*), steady (*zero*) or decreasing (*min*).

Table 4.3: Modelling the role of parameters values

4.2.1.4 Quantity Spaces

The notion of quantity space was introduced by the process centred approach as a lattice of partially ordered parameter values (see figure 2.10). The definition used by Forbus states that quantity spaces specify two aspects:

- the values that a parameter can have, and
- how these values are related to values of other parameters.

For the purpose of integration it is relevant to separate these two different types of knowledge and their *roles* in the problem solving. We therefore redefine the term quantity space to refer to the domain knowledge that performs the role of representing an ordered set of parameter values that a *specific* parameter can have. Relating parameter values of *different* parameters is the role of parameter relations and will be discussed in section 4.2.1.5.

A crucial point in qualitative reasoning is to determine how the quantitative values of a parameter should be abstracted into a set of qualitative values. Two issues have to be considered here:

- what will be the underlying representation for a quantity space, and
- how can the quantitative values of a specific parameter be abstracted into this representation.

As mentioned before, the formalism used for representing quantity spaces strongly interacts with the representation of time. Both in the component and process centred approach time is represented as intervals (lasting a certain amount of real time) during which the behaviour of the system is constant, i.e. does not change its qualitative state of behaviour. This means that real time is abstracted into intervals during which the quantitative value of each parameter stays within the boundaries of the current qualitative value. Each change of qualitative value, because the quantitative value crossed the borderline of the qualitative interval, introduces a new time interval.

The qualitative values of a parameter are divided into intervals and points, for both the component and the process centred approach. In particular, they are represented as a sequence of alternating points and intervals. A parameter can therefore, during a certain period of time (of constant qualitative behaviour), have either a point or an interval as its value. The intuitive understanding behind this approach is illustrated in figure 4.6 for the parameter temperature as it is used to described the characteristics of a substance. All the quantitative values a substance temperature can have, are divided

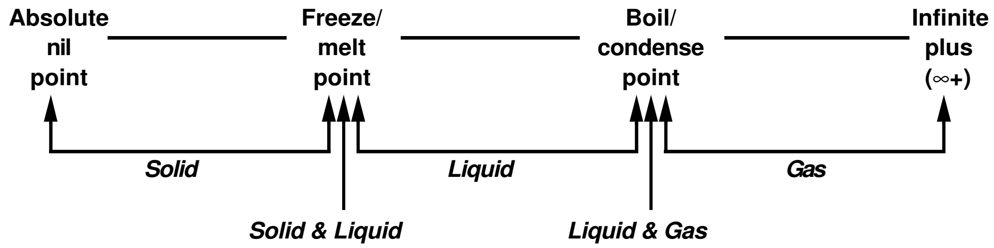

Figure 4.6: The quantity space for the temperature of a substance

into six qualitative values, consisting of three intervals and three points. Each value resembles a characteristic period of constant qualitative behaviour for the substance. If, for example, the temperature has a quantitative value somewhere between freezing point and boiling point and this value increases, then the substance shows constant qualitative behaviour, until it reaches its boiling point, namely 'being a liquid'. As soon as it reaches this boiling point, the substance arrives at a new time interval in which it again shows constant qualitative behaviour, namely boiling.

Using temperature as an example automatically introduces the issue of how the quantitative values of a specific parameter can be abstracted into a sequence of alternating points and intervals. For the temperature of a substance this seems obvious, because the material that is described by the parameter changes considerably (solid, liquid, gas or mixtures) depending on the value of the temperature. However, there are no general guidelines that can be applied to support the abstraction step. Consider for example the temperature of the environment (the earth as a whole) that surrounds a refrigerator. The heat exchange between the refrigerator and this environment may cause the temperature of the environment to increase, but compared with the behaviour of the refrigerant this increase in temperature negligible. The temperature of the environment can therefore be abstracted into a single interval without using freezing point and boiling point, because this distinction is irrelevant with respect to the behaviour that is being analysed. If however the refrigerator is placed in an environment consisting of frozen water, then the quantity space of a single interval will be insufficient. Forbus [70] refers to this as the *relevance principle:* the distinctions made by a quantisation must be relevant to the reasoning being performed.

The notion of quantity space as redefined above, not only generalises the use of parameter values in the process and component centred approaches, but also incorporates the notion of landmarks used in the constraint centred approach. Each approach represents the values that a *specific* parameter can have as an ordered set of alternating points and intervals. Kuipers slightly differs in his implementation in the sense that he only assigns names to points (landmarks) (l_j) and represents intervals as values in between two points (l_j, l_{j+1}). There seems to be a conceptual difference with respect to how the constraint centred approach represents time. Kuipers using the notion of time-points: the behaviour of a system can either be at a certain time-point or in between two time points. Although, each parameter can have both point or interval values at a time-point or in between two time-points, it can only *change*:

- from having a landmark value to having a value in between two landmarks, while going from a time-point to in between two time-points, and
- from having a value in between two landmarks to having a landmark value, while going from in between two time-points to a time-point.

Thus, it is not possible for one parameter to go from a point to an interval while at the same time another parameter goes from an interval to a point. The reason behind this is that the constraint centred approach requires that *all* points for which the qualitative behaviour is going to change, are defined in the quantity space. Therefore if one parameter is going to reach a point, the other parameter necessarily also has to reach a point (or keep its current value). This requirement also explains the need for generation of new landmarks. It is very likely that not all relevant points are modelled in the quantity space of each parameter. Generation of new landmarks allows the creation of these additional points when the overall behaviour of the system requires it.

In our framework we do not represent this requirement explicitly, although it could be imposed upon the transformation inference by including additional precedence rules (section 4.2.1.14). However, there is no need for such a constraint because the inequalities between parameters allow states of behaviour to be reached that were not anticipated in the quantity spaces of the parameters.

The integrated definition of quantity spaces points out a more detailed understanding of the ordering between parameter values. For a single parameter the values are always completely ordered and initially unrelated to values of another parameter. Using parameter relations, in particular those concerned with inequalities, allows representing a partial or complete order between the values of different parameters.

An additional advantage of making quantity spaces specific for a certain parameter and not for the system as a whole, is that parameter values can be defined *independently from a certain context*. We can use the characteristics of the parameters themselves for defining (default) quantity spaces. The values of an amount-of liquid for example, are always defined by the quantity space *zero-plus*, meaning that there is no liquid, or some liquid, respectively. The parameter height only exists when the liquid is being contained by a container of some sort and as such is specific for a certain kind of configuration. Its quantity space is always defined by *zero-plus-maximum*, referring to the height being zero, positive, or having its maximum value. This is true for both open and closed containers. However, this does not necessarily mean that the quantity space of the parameter amount-of has to change to *zero-plus-maximum* (although we may do so!). Instead we allow the parameters to reach a certain (quantitative) value in an (qualitative) interval (*plus*) and become steady. In this case the amount of liquid becomes qualitatively equal to the maximum height of the container and becomes steady (derivative becomes zero), but still keeps the interval value *plus*. This approach makes the generation of new landmarks (points in a quantity space), superfluous. All the landmarks a parameter can reach are defined in its own quantity space. New landmarks (as in the constraint centred approach) always represent interactions between parameters. Instead of arbitrarily introducing new landmarks when the parameter has an interval value and does not change (derivative equals zero), we use inequality relations to explicitly model relevant relations between parameters.

Summarising, a parameter in our approach can have an interval as value and be steady, because it becomes 'equal' to another parameter which has a point as value and does not change. This is typically what may happen in the U-tube example. The height of the liquid column in one container becomes equal to the column height in the other container (=parameter relation) and thereby the system reaches equilibrium (derivatives become steady), but the values of both parameters remain in the interval *plus*. It is not necessary to create new landmarks for those heights.

It is important to realise that a point value always corresponds to a specific quantitative value (even though the precise value may be unknown). This in contrast to interval values which correspond to a whole range of quantitative values. This has specific effects for reasoning about inequalities as discussed in the next section. Usually, the qualitative point value *zero* is regarded as being equal to the quantitative value *zero*. We therefore define the value *zero* to be equal in all quantity spaces. This means that quantity spaces are partially related as soon as they use the value *zero*.[2]

4.2.1.5 Parameter Relations, Qualitative Calculi and Mathematical Models

Parameter relations represent the dependencies between the properties of system elements. In other words, they specify the constraints that hold between parameters of system elements. The qualitative calculus defines the semantics of a relation, i.e. expresses how the relation must be used. The parameter relations that are true in a certain state of

[2] In *GARP* {min, zero, plus} is used as the default quantity space (see also chapter 5).

behaviour represent the mathematical model of the behaviour of some system in the real-world.

Each approach has its own particular set of parameter relations, with its own particular semantics. The component and the constraint centred approach both use different qualitative relations derived from traditional mathematical equations. The process centred approach focuses more on directed relations such as influences and proportionality relations. In this section we propose an integrated set of parameter relations, based on the following requirements:

- Both directed and undirected relations are needed. As pointed out in section 2.3.1.3, only using undirected relations (such as confluences) unnecessarily neglects the causal knowledge that is sometimes available. On the other hand, only using directed relations can also be troublesome, because the causal dependencies in the domain may not be known (see section 2.3.2.3).
- The expressiveness of the resulting set of relations must be such that the relations used by each of the old approaches can be represented.
- Additional expressiveness is required for representing dependencies between quantity spaces and modelling causal relations between corresponding parameter values.

The different ways in which a parameter relation can be applied in our framework are summarised in table 4.4 and further discussed in the following paragraphs.

	Quantity space	**Parameter**	**Derivative**
Quantity space	[-1-]	[-2-]	[-4-]
Parameter		[-3-]	[-5-]
Derivative			[-6-]

Table 4.4: Different ways of using parameter relations

[-1-] Relations between quantity spaces

In our framework each quantity space is specific for a parameter. This implies that values, apart from *zero*, from different quantity spaces are unrelated. If we want to provide the qualitative inference engine with more information about how values from different quantity spaces are related, additional parameter relations between quantity spaces must be specified. Take for example the height of each liquid column in the U-tube problem. For both heights the quantity space *zero-plus-maximum* can be used. Because *zero* appears in both quantity spaces it is known that both the value *plus* and *maximum* of each column are greater than zero, but there is no information concerning the relation between the maximum values of both heights. It cannot be derived whether these maximum values are equal or not (see also figure 4.7). This may lead to ambiguity in the behaviour prediction. The inequality relations given in table 4.5 can be used for defining additional dependencies between the points from different quantity spaces.

There is no need to specify inequality between intervals from different quantity spaces, because their ordering is defined by the combination of (1) the order of values within each specific quantity space, and (2) the inequality defined between points from different

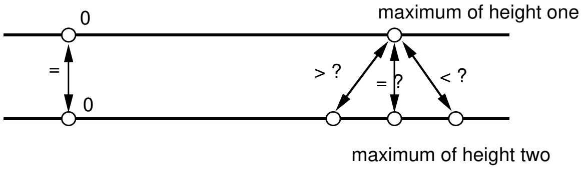

Figure 4.7: Relating points in quantity spaces

equal
greater-than
greater-than or equal
less-than
less-than or equal

Table 4.5: Standard inequality relations

quantity spaces. In the example of the connected tanks, we can derive that value *plus* in one quantity space is less than the value *maximum* in the other quantity space after the values *maximum* have been defined equal.

Relations between quantity spaces differ from relations between parameters (see [-3-]) in the sense that they do not specify what value a certain parameter must have in relation to the value of another parameter. They only specify inequality relations between quantity spaces.

[-2-] Relations between parameters and quantity spaces

The inequality relations (table 4.5) can also be used for relating the value of a parameter to its quantity space. Specifying such a relation does not necessarily imply that the parameter gets a specific qualitative value from its quantity space. It is, for example, possible to specify that the value of a parameter is greater-than (or less-than) a certain point in the quantity space, which means that there may still be a number of qualitative values that the parameter can have. If in the U-tube example we define that the value of the height parameter is greater-than *zero*, then we know that its current value is either *plus* or *maximum*.[3] If on the other hand the parameter is defined as being equal to a certain point in its quantity space, then the parameter has this specific value.

[-3-] Relations between parameters

Relations between parameters differ from relations between quantity spaces and relations between parameters and quantity spaces, because they represent behavioural constraints that one parameter imposes on another parameter. Relations between parameters must therefore be divided into *directed* and *undirected* relations, modelling causal relationships and non-causal relationships respectively. One way of defining undirected relations is to use the inequality relations. The values of two parameters can be: *equal, greater-than, greater-than or equal, less-than,* or *less-than or equal.*

Reasoning with inequality relations between parameters is complex and sometimes

[3]In order to give the parameter the (interval) value *plus*, we have to make the parameter greater-than *zero* and less-than *maximum*.

confusing. If, for example, two parameters have the 'same' interval as value this does not necessarily imply that they are equal. If the two heights in the U-tube system both have the value *plus*, they seem to have the 'same' value, but in fact each *plus* value belongs to a different instantiated version of the *zero-plus-maximum* quantity space. For the two quantity spaces only the value *zero* is related (see figure 4.7). In other words the parameter values are not equal. Adding an *equal* relation between the values *maximum* from the two quantity spaces does not solve the issue, but does specify that the values of the parameters are now the same qualitative interval (the upper and lower boundary of the intervals are equal). However, this does not imply that the parameters are now equal. The reason for this is that the actual **quantitative** value (somewhere in the interval specified by the qualitative value) can still be different for the two parameters. Equality in a qualitative interval must be defined explicitly by an equality relation between parameters (or via equality relations between (1) quantity spaces and (2) quantity spaces and parameters). The following somewhat counter-intuitive statements are therefore true:

- two parameters can be (quantitatively) equal but still have different qualitative values (points and/or intervals), and
- two parameters can have the same qualitative value (points and/or intervals) but still be (quantitatively) unequal.

However, if two parameters both have a point as value and these point values are equal, then the parameter values and therefore the parameters are equal. Equality between two interval values can only be derived if:

- both the upper and lower boundary of the intervals are equal, *and*
- the parameters themselves are equal.

If in the connected tanks system it is known that both heights are equal and that one height has value *plus* then we can only derive that the value of the other height is greater-than *zero*. If it is also known that the *maximum* values of the quantity spaces are equal (so both the lower and the upper boundary of the two *plus* values are equal), then it can unambiguously be derived that the value of the second height is also *plus* (but notice that the parameters themselves are not necessarily equal).

Because having the same value is not the same as being equal, additional relations are needed for defining 'equal' values between parameters. These *value correspondence* relations are based on the notion of correspondence as introduced by Forbus [70]. We have extended this notion resulting in four relations:

- *Value correspondence*
  In general, the relation between two parameters is not known except for specific values that always correspond between the two parameters. In other words the value of one parameter always goes together with a certain value of the other parameter. If one of the values (either point or interval) is known the other value (either point or interval) can immediately be derived. Forbus gives the example of the *length* of a spring and the *force* exerted on the spring. Their values are unrelated except for the *rest-length* of the spring which corresponds to the force being *zero*. There is no causality between these two corresponding values.

- *Directed value correspondence*
  We introduce the directed value correspondence. This is a specialised version of the value correspondence, in the sense that the relation can only be used in one direction. Only if the value of a specific parameter is known (the causing parameter), the value of the other parameter (the effected parameter) can be derived. In real-world systems a specific parameter value often determines the value of another parameter, but the reverse is not always true. The substance flow through a pipe, for example, will be zero if the flow area of the pipe is zero, but if the substance flow is zero it does not necessarily imply that the flow area is zero. Absence of a pressure difference can also be the reason why there is no substance flow.

- *Quantity space correspondence*
  Another extension of the value correspondence is the quantity space correspondence. This relation specifies a value correspondence for *all* values in the quantity spaces. The value correspondence specifies a dependency between two values from two quantity spaces that have been assigned to two different parameters. The quantity space correspondence extends this notion by representing that for each value in the quantity space of one parameter there is a corresponding value in the quantity space of the other parameter. An essential precondition for using the quantity space correspondence is that the quantity spaces of the two parameters have an equal number of alternating points and intervals.

- *Directed quantity space correspondence*
  The directed quantity space correspondence is a further specification of the quantity space correspondence, in the sense that is directed and that therefore the value of the effected parameter only can be determined when the value of the causing parameter is known and not the other way around.

Finally, we have the *subtraction* and *addition* relations. These can be used to add or subtract two parameter values or derivatives.

[-4-] Relations between derivatives and the min-zero-plus quantity space

The values that a derivative can have are usually defined by the min_zero_plus quantity space. The derivative of a parameter can be negative (the parameter value decreases), zero (the parameter value does not change), or positive (the parameter value increases). This can respectively be modelled by the following inequality relations: *less-than zero, equal-to zero*, and *greater-than zero*.

[-5-] Relations between parameter values and derivatives

The following set of relations models dependencies between the *value* of one parameter and the *derivative* of another parameter. It can be used to represent how the existence of a certain parameter (e.g. $flow\_rate$) changes the (quantitative) value of another parameter (e.g. $amount\_of$ liquid). Again there is a distinction between directed and undirected relations. Similar to Forbus [70] we define the directed relation as an influence (see table 2.8). For determining the effect of influences it is not sufficient to simply compare the value of the influencing parameter with the derivative of the effected parameter. Instead, *all* the influences effecting a parameter have to be summed in order to determine the derivative of the effected parameter. Adding influences should proceed as defined in the standard calculus for addition (see table 2.2).

Kuipers [92; 93] uses the derivative relation in an undirected way. When either the value or the derivative is known, the other can be derived. In our framework this relation can be modelled with the implication relation (see also [-7-]).

Also, in $QSIM$ the derivative relation represents a one-to-one relation (two-tuple) between the interval value and the derivative. If more derivative relations effect a particular parameter, this has to be modelled with the help of the *addition* constraint.

[-6-] Relations between parameter derivatives

For relating derivates of different parameters the distinction between directed and undirected is again important. For undirected relations between derivatives the inequality relations as discussed above in [-1-] can be used. However, in most of the systems modelled in our framework we used only the *equal* relation, representing that two parameters change in the same way (either decreasing, steady, or increasing).[4] By using other inequality relations, such as greater-than, it is possible to model order of magnitudes [111] between derivatives.

The opposite undirected derivative relation (if one parameter increases than another parameter decreases) can be modelled with the implication relation (see [-7-]).

Similar to Forbus [70] we use proportionality relations to model causality between derivatives (see table 2.7 for details). Beside the fact that proportionality relations are directed they also distinguish from equality between derivatives in the sense that they do not compare the derivative of one parameter with the derivative of another parameter. Instead all the proportionality relations that effect a certain parameter have to be summed according to the qualitative calculus for addition (see table 2.2) and the resulting qualitative value (often ambiguous) becomes the derivative of the effected parameter.

[-7-] The general implication relation

Finally in our framework we distinguish between two types of general implication relations that can be used to model conditional statements between two relations. All previously defined parameter relations can be used in implication relations. The directed implication relation specifies that if the causing relation holds then the implied relation must be true. The undirected implication relation can be used to specify that if one of the two relations is true than the other relation must also be true.

4.2.1.6 Partial Behaviour Models

The role of partial models is to represent knowledge about the behaviour of entities from the real-world in small units (partial behaviour models), which can be assembled into larger models that represent the behaviour of some real-world system as a whole. The partial models represent knowledge about physics and the physical world that is relevant to certain abstractions of that physical world (=system elements). They specify what features are important for such an abstraction as well as how these features are related to each other. More specifically, they are used to establish a mathematical model that

[4] Note that the M^+ from the constraint centred approach is a combination of both values and derivatives being equal between two parameters.

specifies the behaviour of some system in the real-world.

The constraint centred approach does not use the notion of partial models, but both the component and process centred approach propose modelling primitives for representing partial behaviour models: *qualitative states*, and *views* and *processes*. For purposes of integration the following issues concerning these modelling primitives must be resolved (these issues are discussed in the following sections):

1. Based on the problem solving roles defined in the previous sections, the conceptual relations between the different modelling primitives must be defined.

2. Given this conceptual relation the modelling primitives have to be compared for similarities and differences.

3. We have to decide whether the knowledge that can be presented within these modelling primitives is sufficient for modelling the extended world view that was presented in section 4.2.1.1.

In order to use partial models the qualitative inference engine must be able to derive when a particular partial model applies. Partial models are therefore conditional. The *conditions* specify what knowledge must be known in order for the partial model to be applicable. The *consequences* of a partial model specify the new knowledge that can be derived and added to the behaviour description of the real-world system when the partial model is applied. In table 4.6 an example of a partial model is shown that represents a *liquid-flow* process (this description is based on the liquid-flow process used in the process centred approach). The knowledge captured in this partial model specifies that a liquid

IF a contained-liquid L1 exists
    and L has pressure P1
    and a contained-liquid L2 exists
    and L2 has pressure P2
    and a fluid-path between L1 and L2 exists
    and P1 is greater-than P2
THEN a liquid-flow F exists
    and F = P1 - P2
    and the amount A1 of L1 is negatively influenced by F
    and the amount A2 of L2 is positively influenced by F

Table 4.6: Partial model of the liquid flow process

flow is possible between two contained liquids when they are connected by a fluid path and the pressure of one contained liquid is greater-than the pressure of the other contained liquid. The influences of this process are a decrease of the amount of liquid with the highest pressure and an increase in the amount of liquid with the lowest pressure.

4.2.1.7 Conceptual Relation between Types of Knowledge used in Conditions

As discussed in section 2.3.2 the conditions used in the process centred approach do not sufficiently separate between *type* of knowledge and *use* of knowledge. The *individuals* in a process definition, for example, not only refer to physical objects that must be present but sometimes also refer to other processes that must be applicable. Although both are conditional, they represent different conceptualisations. In particular, for comparison with other modelling primitives, it is relevant that we distinguish between these different types of knowledge and separate this from how they are used. In order to realise this distinction we can use the knowledge roles described in the previous sections. Focusing on the *conditions* first, the following knowledge types can be identified for the modelling primitives used in the process centred approach:

Super-type relation A partial model can be a subtype of another partial model. The subtype inherits the knowledge represented in the super-type (only one super-type relation, no multiple inheritance).

System elements Each partial model requires certain abstractions from the physical world (objects/individuals) to be present.[5]

Parameter relations and values Inequality relations between parameters can be specified as conditions. This also implies that the parameters used in the equality relations have to be defined somewhere. How this is done is partly left implicit. Inequality relations may also refer to specific parameter values that must be present. Correspondence relations cannot be used as conditions

Applies-to relation Partial models can require that other partial models are first applied. The relation differs from the super-type relation, because the partial model is not a more specific version of the other partial models[6] and because multiple models may be defined here.

Conditions for partial models in the component centred approach are less advanced compared with the conditions that may be used for processes and views. Consequently less knowledge can be modelled in those partial models:

System elements Each partial model refers to a specific system element (which is a component) and describes the behaviour of this single system element.

Parameter relations and values The *specifications* of a qualitative state are simple inequality relations that must be true in order for the behaviour model to apply.

Although not explicitly represented, the component centred approach claims to be capable of facilitating a *structural decomposition* for certain components. However, no additional behaviour may be defined for the overall structure. Also it is not clear how the approach derives or represents the conduits between the components resulting from a decomposition.

[5] The notion of preconditions should be interpreted as relations/attributes of system elements.

[6] It is interesting to see how conceptually different notions may require similar computational techniques. From a computational point of view both the super-type and applies-to partial models have to be true for the partial model itself to be applicable, whereas from a conceptual point of view they model completely different types of knowledge.

4.2.1.8 Conceptual Relation between Types of Knowledge used in Consequences

Studying the *consequences* of (applicable) partial models, there is again the problem in the process centred approach of confusing type of knowledge with use of knowledge. However, abstracting from this limitation we can identify the following types:

System elements New entities may appear when a certain behaviour model is applicable (for example *gas* being generated during boiling).

Parameter relations and values Inequality relations between parameters may be defined and therefore must hold when the partial model holds. It is also possible to define new parameters (for example: *let generation be a quantity*), but the exact status of this is unclear. Correspondence relations must also be defined here.

Proportionality relations may be defined in all partial models (relating derivatives), but a special status is given to influences. They may only appear in process definitions.

Consequences, similar to conditions, are in the component centred approach less advanced compared with those used for processes and views:

Parameter relations and values Addition and subtraction relations between parameters and between derivatives (confluences).

4.2.1.9 Conceptual Similarities between Qualitative States, Views and Processes

Comparing the knowledge that can be represented by modelling primitives from the component and process centred approaches, it becomes clear that the latter supports a wider range of possibilities, i.e. more knowledge can be represented. This is true for both the conditions and the consequences. However, from a conceptual point of view the process centred approach can be enhanced by introducing more *conceptual clarity*. It is also interesting to see that, judging from the knowledge that they can represent, views and component models (qualitative states) are not really different. In particular, from a conceptual point of view, both views and component models represent dependencies between properties of system elements. We can refer to these as the *static* dependencies, because by themselves they do not initiate any changes in the behaviour of the system that they model. Either processes (influences) or additional inputs (parameter increase/decrease) are required to provide the *derivatives* of parameters with values.

Of course views do not completely incorporate component models, because the parameter relations are less restrictive in the component models. Parameter relations in component models are undirected and as such impose no causal order on the 'behaviour' that is propagated through the parameter relations. However, a distinction must be made between the kind of knowledge that is represented by a partial model and the degree of restriction (c.q. causality) that can be imposed on the behaviour. There is no fundamental reason why the parameter relations used in views and component models cannot be exchanged. Moreover, as pointed out in section 4.2.1.5, it is essential to use a wider range of parameter relations than only those proposed by the original approaches. The only real difference seems to be that component models apply to a specific group of system elements, namely components, but again there is no reason in principle why behaviour of components cannot be modelled by views.

The similarity between views and qualitative states can be explained by showing the different partial models that can be defined for a substance contained by a closed container. We can model this as one 'component' having five qualitative states depending on the value of the temperature. But we can also model it as a set of views that applies to: substance, liquid, container, contained-liquid, etc. In both cases we will arrive at five qualitatively distinct states of behaviour depending on the value of the temperature (see figure 4.8).

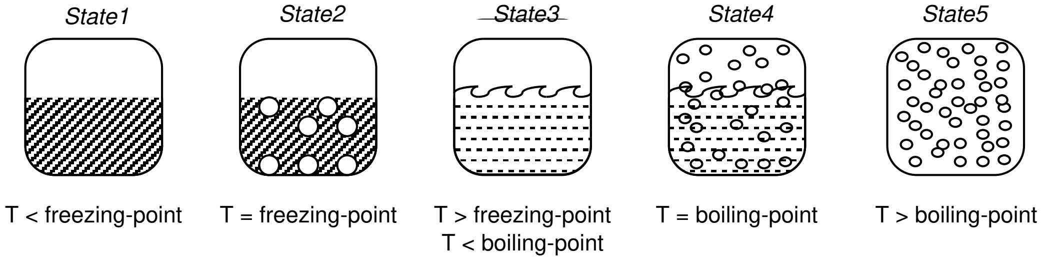

Figure 4.8: States of behaviour of a contained substance

The contents of views can be brought closer to that of qualitative states (they can even be made identical) by changing the amount of detail. Fewer views can be defined, by including the features modelled by the more general views (e.g. substance view) as a part of the more specific views (e.g. liquid). Eventually, this will lead to five specific views, each one representing a different state of behaviour of the contained-substance as pictured in figure 4.8.[7] The argument made here is that, apart from implementation details about how to specify conditions and other such issues, the uses of qualitative states and views are similar: modelling static features of (sets of) system elements. As a result of the chosen domain it always appeared as if component models were better suited for modelling components. But this is a misunderstanding. Components are just as easily modelled by views (or the other way around). The level of detail is not an inherent characteristic of the modelling primitives, but a confounding factor introduced by the domain of application. Of course in some domains, as for example electronics, it is a good heuristic to identify components (switch, lamp etc.) as the system elements, but this does not change the fact that they can both be modelled by views or qualitative states.

The real difference between the approaches is how they realise the changes in the behaviour models. They differ in how one (or more) derivative(s) can be given an initial value. In the component centred approach the derivative of a certain parameter is specified as an additional input for the prediction engine. As this derivative is added to the prediction model from outside, this models the notion of some external agent that enforces a change upon the system. In the pressure regulator[8] example these agents are called *sup-*

[7]Instead of reducing the number of views we can also define more qualitative states, in fact one for each 'system element' in the system, and end up with a more detailed description that closely resembles the initial set of views mentioned above. However, it will be impossible to obey the no function-in-structure principle.

[8]One of the best known examples that has been modelled with the component centred approach is the pressure regulator. The purpose of this device is to ensure a constant pressure at the output, despite a

ply (input) and *load* (output) and can be either decreasing, steady, or increasing (see also figure 4.9). The changes introduced by these agents propagate through all the parameter

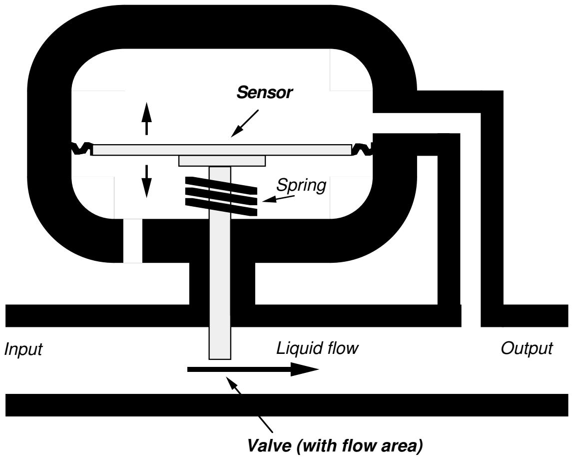

Figure 4.9: The pressure regulator

relations defined between derivatives, resulting in a description of a particular state of behaviour. Next, the transformation inference takes this behaviour as input to find out if the state of behaviour terminates. Among others, this inference imposes constraints (i.e. parameter relations) on the value of the derivatives in the next state in order to guarantee a sequence of behaviour descriptions that changes continuously over time.

This technique by itself is not restricted to the component centred approach, but can also be used in the process centred approach. Namely by finding all the views that describe some set of objects, for example of a contained liquid, and specify (as an initial value) that the temperature increases. The only additional requirement this technique has, is that the constraint satisfier tries to find *all* the derivatives that match the set of parameter relations (generate and test). Clearly, as the influence relations disappear this also reduces the amount of causality represented directly in the final sequence of behaviour, but that is a different issue.

The second way to model changes is basically the opposite of the one described above. It is not based on active manipulations but it is the result of nature's tendency to seek an equilibrium. These are the type of changes used in the process centred approach. They are called processes, triggered by inequalities between parameters and influence the system such that a state of equilibrium is likely to be reached. This technique is (again) not restricted to the process centred approach, but can also be used in the component centred approach. Instead of specifying the derivatives of the supply and load, we can model an inequality between the output and the input pressure that triggers a liquid flow process between them, which causes the supply and load to change.

The exchangeability of (1) using either views or qualitative states to model the static

changing load. The device achieves this goal by controlling the substance (material) flow from the input to the output. When the pressure at the output drops it enlarges the flow area in order to let more liquid flow from the input to the output. If, on the other hand, the pressure increases it reduces the flow area so that less liquid can flow through. Finally, when the pressure at the output is constant it does not change the flow area.

features of a system, and (2) introducing a change by either using a process that is triggered by an inequality or defining changes as additional inputs to the prediction engine, must lead to the conclusion that the distinction between a component centred model versus a process centred model for qualitative reasoning is not a fundamental one.

There are however two guidelines for developing prediction models that result from the different use of parameter relations between the two approaches that are worthwhile taken into account:

- the notion of processes should be favoured over just adding some derivatives to induce changes, because processes explicitly model which inequality causes the changes, and
- in deciding which parameter relations to use, the most restrictive ones should be preferred, because they allow a causal interpretation of the produced behaviour model more easily.

4.2.1.10 Requirements for an Integrated Set of Modelling Primitives

The third question to be addressed concerns the usability of the original modelling primitives for an integrated framework for qualitative prediction of behaviour. Two aspects are relevant for this question:

- the notion of conceptual clarity (confusing knowledge *use* with knowledge *types*), and
- the amount of knowledge that can be represented by the primitives.

The component models allow only a limited amount of knowledge to be represented and therefore provide insufficient flexibility for modelling the rich behaviour that can be found in the real-world. They are therefore not well equipped to be the principal modelling primitives in an integrated framework. However, component models are conceptually clear and the partial models used in the framework should be such that these types of partial models can be represented.

The modelling primitives from the process centred approach allow a richer representation of knowledge and therefore seem to be better candidates for modelling primitives of an integrated framework. Unfortunately, the conceptual clarity is low and is a drawback in the utility of these modelling primitives, in particular, with respect to the knowledge engineer who has to build the prediction models. The modelling primitives of an integrated framework should not confuse the conceptual notions that have to be represented by a modelling primitive with how a specific implementation of that primitive is realised. Moreover, three additional modelling primitives can be identified which are not distinguished by the process centred approach. They result from the following questions:

- how can changes be represented that do not originate from inequalities, and
- is it possible (and necessary) to further distinguish between different types of modelling primitives that represent static properties.

There is a hidden notion in the component centred approach with respect to changes that do not result from inequalities (although the notion as such is not used) namely that some changes are forced upon the world by *agents*. There is, for example, no straightforward inequality that causes a compressor in a refrigerator to increase the pressure. The compressor is more like an agent that enforces certain changes on the physical world, without an explicit cause.[9]

With respect to further distinguishing between modelling primitives for representing static properties, the solution is relatively simple if we take into account how they apply to system elements. There are three cases: the modelling primitive describes a single system element, it describes a combination of system elements, or it decomposes the system element into its subparts.

4.2.1.11 An Integrated Set of Modelling Primitives

Based on the modelling primitives defined by the original approaches and the additional functionality as described above, we propose the following integrated set of modelling primitives for representing partial behaviour models:

Static models describe static dependencies between properties of system elements. Both component models [57] and views [70] fall within this class of partial behaviour models. Three types can be identified, depending on whether the models are context dependent or not:

- **Single description models** are context independent, i.e. they obey the *no-function-in-structure* principle. They describe (static) properties of *one* system element. For example the properties of a container or the behaviour of a component.
- **Composition models** are context dependent. They model properties of a configuration (of or number) of system elements. For instance a contained liquid, that has three system elements, a container, a liquid and a contain relation between those two. Usually a composition model requires other single description models to be active first. Given these behaviour descriptions of single system elements, a composition model adds additional behaviours to the assembly as a whole. The *no-function-in-structure* principle is obeyed in the sense that the behaviour specified for the composition model may not refer to aspects of system elements outside the assembly.
- **Decomposition models** are modelling primitives that can be used for focusing on a specific system element, i.e. for studying the behaviour of some part of the system at another level of detail. For example, decomposing a component into its subparts and reasoning about the behaviour of those subparts. Of particular importance is the question of how the (input and output) behaviour of the decomposed system element relates to the behaviour of the subparts.

[9]Component descriptions are in fact abstractions over a set of processes that are not represented explicitly in the model [21]. Take for example the compressor of a refrigerator. For understanding the cooling-behaviour of the refrigerator it is not relevant to know how the compressor works in detail (e.g. how the piston moves, how its electrical circuit functions, etc). The only relevant aspect of the compressor, in this case, is that it causes a *substance_flow* from the evaporator to the condensor. A decomposition model can be used to focus on the compressor and study its behaviour in more detail.

Process models represent changes in behaviour that happen because a number of system elements interact as a result of an inequality between them (with respect to some quantity). More specifically, a process is triggered by an inequality relation between parameters of these system elements. The descriptions are necessarily context dependent and are similar to the notions of processes in the process centred approach [70].

Agent models are used to model actors that enforce changes upon a system. These models are context independent, because they apply to a single system element (usually a component). An important difference between static models and agent models is that the latter *change* values of parameters. For example, a compressor, if it is modelled as a component that causes an increase in the pressure (which is different from describing the input-output dependencies).[10]

Next, we have to decide upon the formalism that is needed for representing the partial behaviour models. Many of the knowledge roles required within such a behaviour model have already been discussed in the previous sections. In particular, the notions of system elements, parameters, parameter values and parameter relations as well as the distinction between conditions and consequences. In addition the different relations between partial models, such as applies-to and is-supertype-of, have to be taken into account. As a result, each of the modelling primitives for partial behaviour models must be represented in terms of the following knowledge representation:

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).

[10] Agent models are related to the notion of exogenous parameters [87] in the sense that an agent model may represent an exogenous parameter. However, the notion of agent models explicitly refers to a change being forced upon the system.

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.[11]

The isa hierarchy allows the representation of detailed partial models as subtypes of more general ones. The general type represents the features that hold for the whole class whereas the more detailed ones represent features that are specific for the subclass. In contrast with previous approaches we use *multiple inheritance* which defines that each subtype inherits the features from all its *supertypes*.

In figure 4.10 a hierarchy of partial behaviour models for heart diseases is depicted. The partial behaviour models of this hierarchy are further discussed in section 5.4.2.

The function of the applies-to hierarchy is to define which other partial models have to be active in order to allow a particular partial model to be active. For example, a liquid-flow process (table 4.6) requires two open-contained-liquids to be active before the flow process itself can be active. Composition models and process models often require other partial behaviour models to be active.

4.2.1.12 System Model Description

Central to qualitative reasoning is the way in which a system is described during *a period of time in which the behaviour of the system does not change.* The notion of change is subtle, because the actual (real-world) system may change whereas from a qualitative point of view its behaviour remains constant. During the evaporation of a contained liquid, for example, the liquid is transformed into a gas and as such the system changes, but from a qualitative point of view this process is seen as being in a constant state of behaviour, because none of the *qualitative* values that describe the system, change. The modelling primitive that is used for representing a model of the real-world system during a period of constant behaviour plays the role of system model description.[12]

[11]When searching for partial models the qualitative inference engine needs to know which partial models are applicable. Therefore the partial model *itself* should be included here as a partial model that is from now on applicable.

[12]We deliberately defined a new term and did not use the term state, because the latter has connotations with respect to the *situation calculus* in which two or more states of *equilibrium* are related by processes. This is different from the meaning of a state of behaviour in qualitative reasoning. First, a constant state

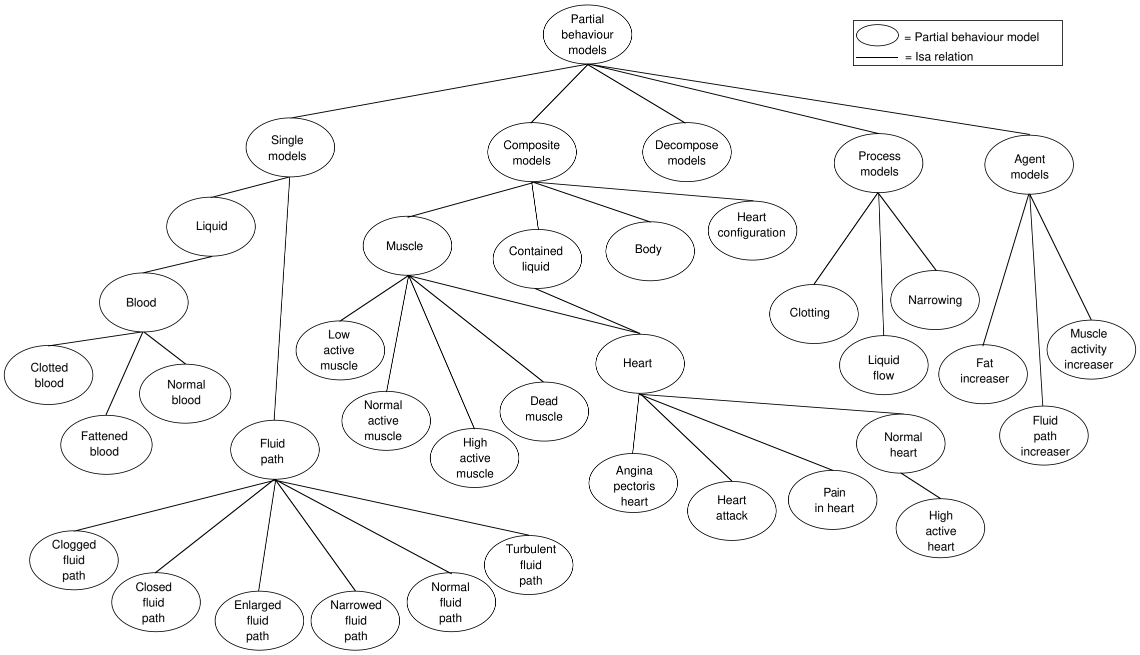

Figure 4.10: A hierarchy of partial behaviour models for heart diseases

Although a system model description should contain all the knowledge relevant to a certain state of behaviour, there are five aspects of special interest. These are used for referring to domain knowledge that models changeable aspects of the real-world system: system elements, parameters, parameter values, parameter relations, and partial models.[13] If the knowledge referred to by one of these aspects changes, then the system model description has to change as well, because the current system model description has become an incompatible model with respect to the changing aspect. In general a new system model description has to be created if *one* of the following changes takes place:

- the parameters that describe the behaviour of the system change their interval values,
- the set of inequalities between parameters changes, and
- the set of system elements, describing the structural configuration of the real-world system, changes.

Each system model description refers to a specific state of behaviour that is manifested by the real-world system.

System model descriptions differ between approaches because certain parts of the knowledge are not explicitly used in the reasoning process. In the constraint centred approach, for instance, system elements are not part of the system model description. Although the $QSIM$ constraints refer to a specific system in the real-world, the physical structure of that system is not represented explicitly. In other words, the qualitative inference engine has no knowledge of the physical structure it reasons about.

of behaviour does not need to be an equilibrium, and second, processes are *part of* the state of behaviour, instead of in between two states of behaviour.

[13]Calculi, for example, do not change when the behaviour of the real-world system changes.

4.2.1.13 Input System

The input system can be regarded as a specific type of system model description. It is used for representing a partial model of some real-world system. This model is used by the qualitative inference engine for behaviour prediction. An input system may contain knowledge about: system elements, parameters, parameter values, parameter relations, and partial behaviour models.

It is interesting to see how the contents of input systems differ between approaches. In the component centred approach the device topology consists only of a number of components and conduits between them (system elements). In the process centred approach a number of physical objects in a certain structure are specified, possibly augmented with some information concerning inequality relations (parameter relations and/or values). Finally, in the constraint centred approach the input system consists of constraints (parameters, relations and values) and landmarks (quantity spaces). The input system defined above can easily model each of these cases.

4.2.1.14 Transformation Rules

Knowledge used for deriving how a certain state of behaviour changes into another state of behaviour is represented by the original approaches in the form of rules. We will therefore refer to this knowledge as performing the role of transformation rules.

These sets of rules are flat, in the sense that they are all of the same type, and are usually applied as constraints that must hold between states of behaviour. In our approach we abstract from these specific sets and distinguish between three types:

- termination rules
- precedence rules, and
- continuity rules.

Each group of rules refers to a specific type of knowledge that is represented and will be further discussed below.

Termination rules specify the conditions under which a particular state of behaviour will terminate because the behaviour represented in the system model description is no longer compatible with the actual behaviour of the real-world system. Termination rules identify one of the following cases:

- A parameter takes a higher value from its quantity space, because its quantitative value increased ($derivative = plus$). For example, the temperature of liquid increases until it reaches its boiling point.
- A parameter takes a lower value from its quantity space, because its quantitative value decreased ($derivative = min$). For example, the temperature of liquid decreases until it reaches its freezing point.
- Two parameters which are equal become unequal ($greater\_than$ or $less\_than$) because they change in opposite (or at least dissimilar) ways.[14] For example: two

[14] In the examples that we modelled there was no need to include the case of two equal parameters

equal pressures become unequal because one pressure starts increasing and becomes *greater_than* the other pressure.

- Two unequal (*greater_than* or *less_than*) parameters become equal because they change towards each other. For example: two unequal pressures become equal because the lower increases.

Although changes in parameter values and parameter relations are the basis for terminations, other aspects can change as well. For example, the system element 'liquid' will disappear when its *amount_of* becomes *zero*.

An example of a termination rule is shown in table 4.7. The knowledge in this rule represents a part of the *limit rule* [57]. It specifies that when a parameter has an interval value in the current state of behaviour and decreases, that it will have the point value below this interval in the next state of behaviour. In other words, when the quantitative value of a parameter is somewhere in an interval, decreases, and there is a point below this interval, then the quantitative value of the parameter will reach this point.

IF a parameter Par has qualitative value V
    and Par is decreasing
    and Par has quantity space QS
    and V is an interval value from QS
    and P is the next value in QS below V
THEN in the next state of behaviour SMD
    Par has value P

Table 4.7: Termination rule for changing to a point 'below'

Having found all possible terminations the precedence rules specify the order in which changes take place. If, in a particular state of behaviour, more than one termination is possible then the precedence rules attempt to determine which of these terminations should be applied first. Take for example, two different kinds of liquids in one container. If the temperature increases then both liquids will reach their boiling temperature, introducing ambiguity because of two possible terminations. However, if it is known that the boiling point of one liquid is lower than the boiling point of the other liquid, the ambiguity can be solved. Only the liquid with the lower boiling point will reach its boiling point in the next state of behaviour.

It is also possible that two, or more, terminations reflect aspects of the same change in behaviour. If, for example, the *amount_of* liquid goes to *zero* then also its *volume* and its *pressure* will go to *zero*. Such a similarity can be represented with correspondence relations (both undirected and directed value and quantity space correspondence). Parameters which are subject to termination and that have a correspondence relation, should therefore be merged into one termination. This is shown in table 4.8.

becoming different because they behave with different magnitudes in the same direction, but such a rule could be included.

IF a parameter Par1 terminates to interval value V1 below and a parameter Par2 terminates to interval value V2 below and Par1 and Par2 have a correspondence relation for V1 and V2 **THEN** consider the two terminations as one termination (merge)

Table 4.8: Precedence rule for merging corresponding terminations

Finally, continuity rules specify additional conditions that must be satisfied by the new state of behaviour in order to be a valid successor state of behaviour. In particular, they deal with those aspects present in the current state of behaviour that are not part of a termination. All these aspects should stay constant between the old and the new state of behaviour. An example is shown in table 4.9 which specifies that the qualitative value of a parameter has to be the same in the next state of behaviour, but that the derivative may change from decreasing in the current state, to decreasing or steady in the next state of behaviour.

IF a parameter Par has qualitative value V and Par is decreasing **THEN** in the next state of behaviour SMD Par has qualitative value V and Par may either decrease or stay constant

Table 4.9: Continuity rule for parameter values

It is interesting to see, that the I- and P-transitions from the constraint centred approach are implicit combinations of termination and transition rules. Applying the I- and P-transitions incorporates a kind of catch-all procedure that tries out all possible transformations between two states. The constraint centred approach does not have explicit relations between parameters and therefore it cannot take the available knowledge about parameter relations into account.

A relevant feature of applying the rules as described above is that the transformation between two states of behaviour is not found by satisfying constraints, but by a *directed* search which provides a *causal account* of why a state terminated, how this termination was ordered with respect to other terminations and how this is used, together with continuity rules, for deriving the next state of behaviour.

It is important to notice that the examples given above can all be realised with knowledge that is domain independent. However, in some cases it may be worthwhile to add domain specific knowledge in the rule used. For example, the parameter *heat* is unlikely to go to *zero* in the real-world and therefore the termination for that parameter can be removed. However, in most cases it is possible to solve the problem without using domain dependent knowledge. In the case of the parameter *heat*, the termination will not be applied if the parameter is assigned to a quantity space that has only one, positive, interval.

4.2.1.15 Behaviour Descriptions

The behaviour description is a graph of possible system model descriptions (states of behaviour), with for each description specified:

- its internal behaviour (in terms of system elements, parameters, parameter values, parameter relations, and partial models),
- its preceding system model descriptions (from-list), and
- its successor system model descriptions (to-list).

Each system model description has a *to-list* and a *from-list*. The to-list specifies which terminations (and possibly how they were merged) caused each transformation to the next system model description. The from-list specifies which system model description preceded the current system model description.

4.2.2 Inference Structure

The knowledge at the inference layer abstracts from the domain specific modelling primitives by:

- describing the *canonical* inferences used in the reasoning process, and
- pointing out the *role* the domain knowledge plays in this reasoning process.

In previous sections the problem solving roles relevant to qualitative prediction of behaviour have been described. Table 4.10 gives an overview of these meta-classes. Given this list of meta-classes, the canonical inferences (knowledge sources) used in qualitative prediction of behaviour can be described. Figure 4.11 gives an overview of these inferences. In the following sections each knowledge source is discussed in more detail.

4.2.2.1 Compound Specify

The purpose of the specification inference is to develop a complete[15] description of a particular state of behaviour of a system (=augmented SMD), which means satisfying the appropriate parameter relations and assigning qualitative values to parameters that are used to describe the system. More specifically, it has to find the partial behaviour models whose conditions satisfy the knowledge represented in the input system. If the conditions can be satisfied, the knowledge represented in the partial model becomes part of the system model description. This additional knowledge may lead to satisfaction of conditions of other partial models (which could not be satisfied before). Finally, there may be partial models that have conditions which cannot be derived from the knowledge present in the system model description, but which are *consistent* with that knowledge. In such cases the

[15]Complete with respect to the knowledge that is available to the qualitative inference engine.

Meta-class name	*Description of the role*
System model description	A model that describes the real-world system during a period of time in which the behaviour of that system does not change.
System elements	Abstractions from the real-world to which partial behaviour models apply.
Parameters	Properties of system elements.
Parameter values	Values of parameters.
Quantity spaces	An ordered set of values that a specific parameter can have.
Parameter relations	Dependencies between parameters.
Qualitative calculi	Semantics of a parameter relation.
Mathematical models	A set of parameter relations that holds during a particular system model description.
Partial models	Small units representing partial behaviours which are assembled into larger models that represent the behaviour of some real-world system.
Transformation rules	Knowledge about how to find successive system model descriptions. Three types of rules have been identified: *termination*, *precedence* and *continuity*.
Behaviour descriptions	A set of system model descriptions ordered in time, representing the potential behaviour of a system.

Table 4.10: Meta-classes in qualitative prediction of behaviour

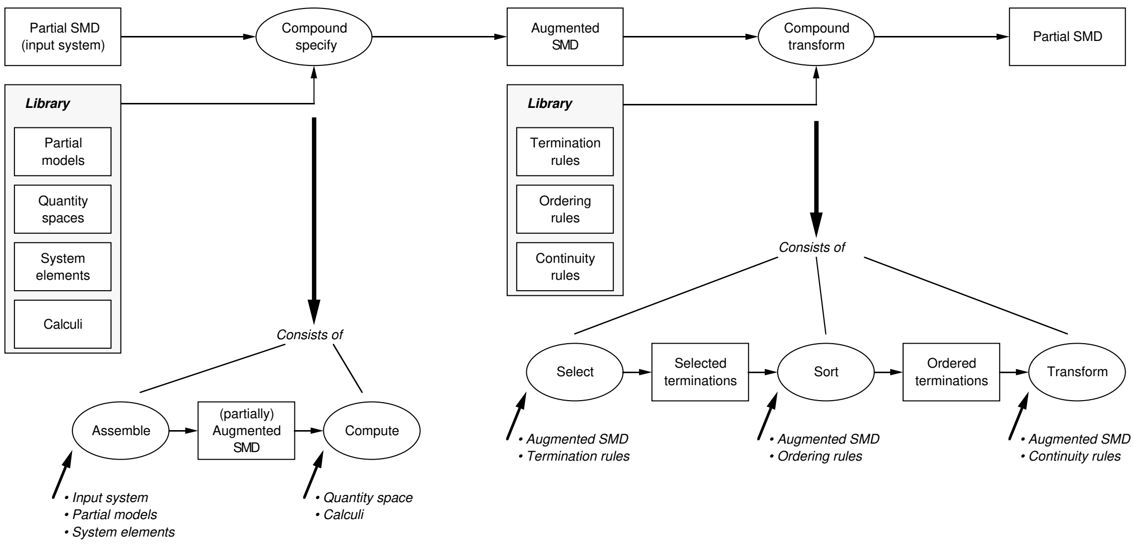

Figure 4.11: The inference structure for prediction of behaviour

compound specification inference should assume that these conditions are derivable, and add the knowledge of those partial models to the system model description. In the case of conflicting assumptions, alternative specifications must be made for each of these. The compound specification inference can be decomposed into two other inferences that must be carried out in order to arrive at such a description, namely *assembling* a mathematical model and *computing* the corresponding qualitative values.

Assemble

Assembling the mathematical model (=partially augmented SMD) is done by finding all the partial models that apply to the input system (=partial SMD). In the constraint centred approach the mathematical model is provided by the user. Consequently, the qualitative reasoning approach is not concerned with establishing this model, it only computes the qualitative values (see below). In the process and component centred approaches, on the other hand, the user only provides a partial system description and therefore the qualitative inference engine must determine the corresponding mathematical model itself. In both approaches the mathematical model is built by finding partial models that apply to the initial input description. Each applicable partial model introduces parameter relations and parameter values. Together these make up the mathematical model. The partial models also specify which parameters are going to be used for describing the properties of the system elements. In the component centred approach, for instance, the confluences defined in a component model immediately specify the parameters that are considered by the mathematical model.

Compute

Once the system model description is augmented with a mathematical model, the computation of qualitative values and the consistency checking of parameter relations can be done. The following outputs are possible:

- *contradiction*, which means there is no set of parameter values consistent with the current set of parameter relations and/or there are inconsistent parameter relations,
- *solution*, which means one or more sets of parameter values are consistent with the current set of parameter relations and there are no inconsistent parameter relations, and
- *unknown*, which means that there are parameter relations and/or parameter values which cannot be derived from the current parameter relations and parameter values specified in the system model description.

In the case of an inconsistency the partial model will not be added to the system model description. If no partial model is consistent with the input system then the input system cannot be specified. In the case of a solution the partial model can be added to the system model description. In the case of multiple solutions, or ambiguity, alternative system model descriptions will be created. Finally, in the unknown case the inference should assume that the partial model applies. As mentioned before, inconsistent assumptions must lead to alternative specifications.

If there occurs an inconsistency, or any other kind of problem, when a partial model is being added to a system model description, then the system model description as a whole must be removed. An inconsistency in the consequences of a partial model means that the knowledge has not been modelled adequately. In other words, there is an inconsistency in the available domain knowledge.

4.2.2.2 Compound Transform

The transformation inference is concerned with identifying successive states of behaviour. A current state of behaviour terminates when the behaviour of some part of the system model description changes and as a result is no longer compatible with the current system model description. The compound transform can be decomposed into *selecting* terminations, *sorting* terminations, and *transformation* (subsume or specify) to a new state of behaviour.

Select

Termination rules specify the conditions under which a particular state of behaviour will terminate and are used to select the possible terminations from the current system model description.

Sort

Precedence rules specify the order in which changes take place and are used to sort the possible terminations. There are three steps that must be carried out:

- *merge* all the terminations that are related and therefore form a single termination,
- *remove* those terminations that do not take place because of other terminations happening first, and
- *assemble* the cross-product of all terminations that have not been removed. The resulting set resembles all terminations that can take place.

Transform (Subsume or Specify)

The *continuity* rules specify how unchanging aspects in the old system model description should reappear in the new system model description. They are, together with the ordered terminations, the input for transforming the current system model description into its successor. With respect to the new system model description there are two possibilities, either it has been generated before or it is a new state of behaviour. In the former case the partial system model description is a *subset* of an already existing system model description. In the latter case a new state of behaviour has to be *specified.*

4.2.3 Task and Strategic Knowledge

The task layer is used to represent typical chains of inferences that experts make in solving a particular, well-known task. However, in the prevailing approaches to qualitative reasoning the task layer is only minimally filled. They distinguish between:

- finding all states of behaviour (total envisionment), or
- finding one specific trace of behaviour (attainable envisionment).

In the case of the latter additional input parameter values are taken into account to limit the number of system model descriptions that can be found.

Strategic knowledge, in the sense of the four layer model, is not present in the original approaches to qualitative reasoning. The approaches always execute the same task structure, are not able to monitor their own inference process, and as such are not able to modify or change their own reasoning process.[16] However, it is likely that the reasoning process will encounter difficulties, i.e. that the problem solving goal cannot be reached. Strategic knowledge will be further discussed in the chapters 6 and 7.

4.3 Concluding Remarks

In this chapter we have described a conceptual framework for qualitative prediction of behaviour, that unifies the three basic approaches to this problem solving task. The unification is based on the $KADS$ methodology and extends the previous approaches by distinguishing between domain, inference, task and strategic knowledge. This presents a frame of reference for comparing the original approaches on how they use these different types of knowledge. Of particular interest is the inference layer, because it abstracts from the domain specific notions by describing the *canonical* inferences (see figure 4.11) used in the reasoning process, and pointing out the *role* (see table 4.10) the domain knowledge plays in this reasoning process. The three original approaches to qualitative reasoning can be viewed as using parts of the inference layer described in this thesis for qualitative prediction of behaviour.

The unified approach presented in this chapter advances each of the original approaches as a result of the integrated view. The most important contributions can be summarised as follows:

- The notion of system model description, which is used in different ways in all three approaches, is extended to include a full description of the elements in the physical system, the partial behaviour models, the parameters, the parameter values and the parameter relations.

- The notions of view, qualitative state and process are unified and extended to a domain ontology for partial behaviour models that 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 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 elements. Agent models are used for modelling changes that are caused by agents.

- The set of parameter relations provided by the integrated framework enables a broader functionality for specifying dependencies between parameters. In particular, we can use both directed (causal) and undirected (non-causal) dependencies

[16] There is some work going on as an extension of the constraint centred approach to filter (and thereby reduce) the number of generated states (cf. [95]).

between derivatives and between parameter values in a single behaviour model. In addition, the notions of directed and undirected quantity space correspondences and directed value correspondences are new.

- The transformation inference explicitly distinguishes between *termination*, *precedence* and *continuity* rules. Each of these rules refers to a different type of knowledge used in the transformation inference. In qualitative reasoning the transformation step, in particular, tends to cause unmanageable branching of possible states of behaviour. In our approach this ambiguity is reduced by representing precedence knowledge for merging related transitions or filtering out undesired transitions. The distinction between finding terminations and then explicitly ordering them was not present in earlier approaches.