# Abstraction Framework for Compositional Modeling

**Diane Chi and Yumi Iwasaki**
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C, Palo Alto, CA 94304
chi@ksl.stanford.edu and iwasaki@ksl.stanford.edu

**Abstract**

Abstractions transform the representation of a complex problem into a simpler, more manageable form. Many researchers have proposed methods of abstraction for various types of problems in specific problem areas, but it is difficult to compare these methods. The representation formalisms of the abstraction methods vary widely, and even the definition of abstraction changes from one group to another.

In this paper, we present a framework for characterizing various abstraction relations in the context of compositional modeling. Our framework classifies abstractions along two dimensions: the *method* used to transform the representation and the *representational element* to which the method is applied. We limit our discussion to the model fragments used in compositional modeling so that we may precisely define each abstraction method and analyze their representational and computational consequences. Such a framework is an important step towards automatic model formulation as well as automatic generation model fragments.

## Introduction

Abstraction is an essential concept in modeling complex phenomena. For a given phenomenon, there are many possible abstraction levels at which it can be modeled. There is no single "correct" level of abstraction, since any model is necessarily an abstraction and the goodness of the abstraction depends on one's goal, i.e. the problem one is trying to solve by constructing the model. For a model to be useful, it must be at the appropriate level of abstraction, which means it must contain enough information to answer the given question with sufficient precision and accuracy but without containing too much unnecessary detail.

A host of different methods of abstraction have been proposed in many problem areas such as modeling (Iwasaki & Simon 1994; Amador & Weld 1990; Williams 1991), planning (Amarel 1981; Fikes, Hart, & Nilsson 1972; Sacerdoti 1974; Knoblock 1989), learning (T. M. Mitchell 1990; Minton 1988; Knoblock 1990; Giordana & Saitta 1990), and theorem-proving (Giunchiglia & Walsh 1992; Plaisted 1981; 1986). However, it is difficult to compare the computational and representational implications of each abstraction technique. For one thing, the representation formalisms of the methods vary widely. Furthermore, the abstraction methods themselves range from the simple act of deleting elements from the representation to using a completely different ontology, seeming to render the term "abstraction" as a catch-all term for any transformation of a representation. We note however, that all transformation techniques deemed as abstraction share a common goal of simplifying the problem representation with the intention of simplifying the problem solving process as a result with some metric of simplicity.

Giunchiglia and Walsh introduced a theory of abstraction based on work in theorem-proving and planning (Giunchiglia & Walsh 1992), where they informally defined abstraction as "the process of transforming the representation into another form that is simpler to handle yet retains the desirable characteristics of the original problem". In this paper, we adopt this general definition and consider various types of abstraction transformations in the context of compositional modeling. In particular, we present a framework for characterizing various abstraction relations among model fragments and their components.

Our motivation for this work is to extend our model formulation algorithm to include all types of abstraction methods on model fragments. We have been working on a technique for automatically formulating a model that is appropriate for answering a given query. Iwasaki and Levy reported on an approach based on the relevance and irrelevance of knowledge that characterizes modeling assumptions underlying model fragments and selects among them to formulate an adequate model for a given query (Iwasaki & Levy 1994). This relevance-based approach is appropriate when the difference between model fragments can be adequately characterized by the difference in the aspects of the situation that are considered *relevant* (and are thus included in the model fragment). We found that though most cases of abstraction relations among models can indeed be characterized by such differences, there are cases where such differences are awkward to capture in this manner. Such cases include situations where a model fragment is abstracted using simpler mathematical relations and situations where one quantity that is hard to measure accurately is approximated by another whose value is easy to obtain. Thus, a broader framework for characterizing relations among alternative descriptions is needed to determine how to extend our current model formulation mechanism.

Our framework classifies abstractions along two dimensions: the method used to transform the representation and the aspect of the representation to which the method is applied. The methods are aggregation, elimination, and approximation. Generally speaking, aggregation replaces a set of elements (of the same type) in the representation by one aggregate element. Elimination removes selected elements from the representation. Approximation replaces an element by another that is deemed "close" to the original by some measure of closeness.

These three general methods are applied to different types of representational elements. In the compositional modeling paradigm, a model consists of model fragments, which in turn consist of conditions and consequences. Both conditions and consequences consist of relations on quantities.[1] The three general methods are applied to each of these representational elements with various computational implications. Limiting our discussion to model fragment abstraction allows us to define each abstraction method precisely. It also allows us to analyze their representational and computational consequences in concrete terms.

Though many specific abstraction techniques have been presented, only a few general theories of abstraction have been proposed. Giunchiglia and Walsh classify abstractions into Theorem Decreasing (TD) and Theorem Increasing (TI) abstractions based on whether the abstract representation has less theorems (and thus more interpretations) than the original representation or vice versa (Giunchiglia & Walsh 1992). Their framework examines the properties of logical theories (which correspond to models in our context) where one is an abstraction of the other. However, the classification of all abstractions as TD or TI is too coarse to provide much insight into the effects of different abstraction models on the consequences of simulation. Furthermore, we find that some common types of model transformations that may be considered as types of abstraction are not TD nor TI.

Our classification approach more closely resembles the approach taken by Struss in his theory of model simplification and abstraction. He formally defined abstraction, approximation, and simplification, and analyzed some representational consequences of such transformations (Struss 1991). Our three general methods—aggregation, elimination, and approximation—generally correspond to his definitions of abstraction, simplification, and approximation. One important difference is that while Struss considers relations among (complete) models, we study abstraction relations among *model fragments*, which must be combined to compose a complete model. The fact that a the problem of model fragment abstraction is much less constrained than abstraction of complete models makes it more difficult to prove general theorems about the representational and computational implications of performing abstractions on them.

This document is organized as follows: In the remainder of this section, we briefly describe the compositional modeling paradigm and our approach to model formulation. The second section presents our abstraction framework and a small example of a graph of model fragments related by the abstraction relations described in the framework. The final section discusses the implications of our work.

## Compositional Modeling and Relevance-based Model Formulation

In this section, we briefly describe the compositional modeling paradigm for representing physical knowledge and predicting behavior (Falkenhainer & Forbus 1991), as well as our method for model formulation based on relevance (Iwasaki & Levy 1994).

A physical situation is modeled as a collection of model fragments [2] Each model fragment represents some aspect of a physical object or a physical phenomenon. A model fragment is composed of conditions and consequences. The conditions specify the individuals that must exist and the requirements they must satisfy for the phenomenon to occur. The consequences specify the functional relations among the attributes of the objects that are entailed by the phenomenon.

If there exists individuals $a_1, \ldots, a_n$ that satisfy the operating conditions of a model fragment $M$ at time $t$, we say that an instance of $M$ is active at that time. We will call $a_1, \ldots, a_n$ the *participants* of the instance of $M$. We denote the particular instance by $M(a_1, \ldots, a_n)$.

The prediction mechanism on model fragments works generally as follows: For a given situation, the system identifies the model fragments whose conditions hold as the active model fragment instances. In each state, this set of active model fragments forms the *simulation model*. The simulation model gives rise to equations that must hold among variables as a consequence of the phenomenon taking place. The prediction mechanism uses the equations to determine the next state

[1] By *quantities*, we mean any kind of attribute of a model fragment. Quantities may be numerical or non-numerical attributes. In this paper, we assume that the range of a quantity forms some kind of a metric space, though this restriction is not strictly necessary for application of most of the abstraction methods discussed in the section on the Abstraction Framework.

[2] We will use the definition of model fragments as given in (Farquhar *et al.* 1993).

of the simulation. Each state has a simulation model along with a set of variable values and predicates that hold. The prediction mechanism outputs a sequence of states. If prediction is performed qualitatively, the output can be represented as a graph. Each path through the graph from the initial state represents a possible behavior of the system. Such a path is called a *trajectory*.

Our model formulation approach described in (Iwasaki & Levy 1994) consists of making two choices. The first choice is deciding what phenomena to represent in the model. The second choice is selecting the model fragment(s) to include in the model from the set of all possible model fragments. The set of possible model fragments includes different descriptions of the modeled phenomena using different modeling assumptions. The first choice is made by backward chaining through the possible causal influences on the variables of interest to the user. The second is made by reasoning about the modeling assumptions necessary to answer the given query. To facilitate this choice, model fragments in the knowledge base are organized into structures called *assumption classes*. An assumption class is a graph of model fragments representing different ways to describe the same phenomenon. The language of relevance and irrelevance of knowledge (Levy 1993) is the basic language used to represent the modeling assumptions underlying different model fragments in an assumption class. Our goals for developing the abstraction framework presented in this paper are (1) to expand this language, especially in the direction of being able to characterize a broader class of abstraction relations more precisely, and (2) to facilitate selection among model fragments by enabling analysis of the representational and computational implications of using different abstract model fragments.

The following section presents our abstraction framework, which provides a classification of ways to operate on each type of representational element to produce a more abstract version. We consider only transformation techniques that do not add new information to the representation, since it is debatable whether transformations that add information are abstractions at all. Also, we do not consider cases where the abstract representation employs a completely new ontology of the domain unless such an ontology is actually a product of applying one of the abstraction methods discussed below.

## Abstraction Framework

We classify all abstraction methods into three general classes, namely elimination, approximation, and aggregation.

- **Elimination** is the removal of all references to some selected elements of the representation except in cases where such removal results from applying approximation or aggregation.
- **Approximation** replaces an element by another element that is less accurate, but closely resembles the original element. Approximation applies only to elements in some metric space that can be used to establish a measure of similarity.
- **Aggregation** involves grouping related elements into aggregates and representing the problem in terms of the aggregates.

The general abstraction methods can be applied to the representational elements of the knowledge base and predicted behaviors. Table 1 summarizes the applicability of the methods to each type of element.[3] A table entry "s" indicates that the method applies to a single element (i.e. a single quantity, a single model fragment, etc.), while "m" indicates that the method applies to multiple elements. As shown in the table, aggregation requires more than one element of a particular type, while the other methods apply to individual elements.

In the following, we provide detailed explanations of each entry in the knowledge base part of the table. As described in the previous section, a knowledge base consists of model fragments composed of conditions and consequences. Conditions and consequences are defined in terms of relations among quantities. Although we discuss abstraction of each type of representational element individually, we must note that abstractions of different elements are not independent. Abstraction of one type of element often necessitates abstraction of another type of element as discussed below.

### Quantity Abstraction

A quantity $Q_1$ of a model fragment $MF$ is formally defined as a mapping from the set of all instances of $MF$ to a set of functions that map time to actual values. In other words, if $MF_1$ is an instance of $MF$, then $QF_{11}$ is a function of $Q_1$ and $MF_1$ such that $QF_{11}(t)$ for some time $t$ is the value of the quantity $Q_1$ of the model fragment $MF_1$ at the time $t$. We distinguish abstraction of the range of a quantity[4] and abstraction of the quantity function ($Q_1$) itself.

**Range Abstraction.** When abstracting the range of a quantity, one changes the set of possible values for the quantity. Elimination and approximation abstract individual values in the range of a quantity. Aggrega-

[3] For the sake of completeness, the table includes behavior abstraction, since the general methods can also be applied to the representation of behaviors. In this paper however, we limit our discussion to the abstraction of model fragments.

[4] We are actually referring to the range of the the function $QF_{11}$ returned by the quantity function applied to a model fragment instance. We will call it the *range* or *value range* of the quantity for brevity unless the meaning is not clear from the context.

| Repr. Element / Abstr. Method | Knowledge Base | | | | | Behavior | |
|---|---|---|---|---|---|---|---|
| | Model Fragment | | | | | State | Traj. |
| | Quantity | | Cond. | Conseq. | | | |
| | Range | | | | | | |
| Elimination | s | s | s | s | s | s | s |
| Approximation | s | s | s | s | - | - | - |
| Aggregation | m | m | m | m | m | m | m |

Table 1: Applicability of Abstraction Methods to Elements of the Representation

tion partitions the range into subsets so that the values are represented in terms of the subsets.

**Range elimination** removes an individual value or a set of values from the range. The removal of selected values is often motivated by knowledge that certain values are unattainable or highly unlikely for the given quantity. For instance, in a model fragment representing a thruster component of a jet propulsion system, one may eliminate all negative values in the range of `pressure-differential`, which is defined as the difference between the input pressure and the output pressure. Elimination of all negative values amounts to making the assumption that there is no possibility of reverse pressure. Such an abstraction can reduce the computational effort of a prediction mechanism by pruning unlikely behaviors.

**Range approximation** replaces a value (or a subrange of values) in the value range of a single quantity with another value (or subrange of values) that is somehow "good enough" for the given problem. Examples of range approximation include changing the precision for quantity values and idealization of quantities. If one changes the precision of numerical values by rounding them all to the second decimal place, a set of values is replaced by a single approximate value. *Idealization* is an extreme example of such an approximation where one replaces the entire value range of a quantity with a single extreme value. The approximation of the interactions between solid objects as frictionless surfaces is an example of idealization. Range approximation results in an abstract representation that produces less accurate numerical predictions.

**Range aggregation** occurs by grouping elements in the range into subsets and using the subsets to represent values. The grouping creates either regular or uneven intervals. Discretization of a continuous range of a quantity into intervals bounded by landmark values, as in qualitative calculus, is an example of range aggregation. Range aggregation results in less precise predictions. When the quantity abstracted is time, range aggregation results in temporal abstraction.

**Quantity Function Abstraction.** We now consider abstraction of quantity functions.

**Quantity elimination** removes a quantity from the representation. For example, consider a model fragment representing a tank with two pressure sensors provided for redundancy. If the original model fragment has two pressure quantities corresponding to each of the sensor readings, one can abstract the model fragment by eliminating one of the pressure quantities. This generates a more compact model fragment; however, as in this example, quantity elimination may eliminate the redundancy present in the actual device and result in a less robust model.

**Quantity approximation** replaces a quantity with a similar quantity where the value of the new quantity might be easier to obtain. For instance, consider a detailed model fragment of a fuel tank including the quantity `amount-of-fuel`. If the exact measurement of the remaining amount is difficult to obtain due to the structure of the tank, but the initial amount and the history of the fuel consumption are available, one might create another model fragment that contains the quantity `computed-fuel-amount` based on these variables to replace `amount-of-fuel`. However, if the approximation is only good under some assumptions, then a model consisting of the approximate model fragment might fail to reflect reality when the assumptions do not hold. For instance, in the case of `amount-of-fuel`, if there exists a large leak between the tank and the component that actually consumes the fuel, `computed-fuel-amount` will not adequately approximate `amount-of-fuel`.

**Quantity aggregation** replaces a set of quantities with a new representative quantity. There are a variety of ways to define the representative, including summing and averaging. For a tank having two pressure sensors that monitor the pressure within the tank, one may replace the quantities `pressure-reading-from-sensor-a` and `pressure-reading-from-sensor-b` with an `average-pressure-reading` quantity whose value is the average of the original two quantities. Quantity aggregation reduces the number of quantities, but it may or may not affect the prediction accuracy depending on the other equations in the simulation model.

Since quantities are used to state the conditions and consequences of model fragments, abstraction of a quantity necessitates the modification of conditions and consequences that reference the abstracted quantity. In ABSTRIPS, abstraction layers are defined in this manner (Sacerdoti 1974). The planning operators in STRIPS are comparable to model fragments with the applicability conditions and the consequences (consisting of ADD and DELETE lists) of the operators corresponding to the conditions and consequences of model fragments. ABSTRIPS ranks the predicates used in specifying the conditions and consequences of the planning operators according to their criticality. To define an abstract planning space, abstract operators are defined by ignoring less critical predicates in the conditions and consequences of the original operators.

## Condition Abstraction

The *conditions* part of a model fragment is a list of atomic formulae, which is an implicit conjunction of the conditions.

**Condition elimination** removes selected conditions from the list of conditions in a model fragment. Condition elimination may occur as a necessary consequence of quantity abstraction, but it is also performed independently when a condition is deemed uncritical for some purpose. Statistical information may motivate such elimination. If a condition is known almost never to fail, one may decide to eliminate the condition to produce a simpler model. We also include in this category replacement of a condition $C_1$ by another condition $C_2$ that is strictly weaker than $C_1$, since $C_1$ is logically equivalent to $C_1 \wedge C_2$ if $C_1 \rightarrow C_2$.

**Condition approximation** replaces a condition with another similar condition, which is simpler in some respect. For instance, the activation condition for a tank model fragment may require that `(> pressure-differential 1e-5)`. In an abstract model, we might replace this condition with `(> pressure-differential 0)`. Such replacement might occur as a necessary result of range approximation, or it might result as an independent decision.

**Condition aggregation** replaces a set of conditions with an aggregate condition. For instance, one may replace the conditions `(= inflow-1 outflow-1)` and `(= inflow-2 outflow-2)` with `(= (+ inflow-1 inflow-2) (+ outflow-1 outflow-2)`. Condition aggregation reduces the number of conditions in a model fragment, but the new conditions may be more complex syntactically. The aggregate condition is often weaker than the conjunction of the original set of conditions.

## Consequence Abstraction

The *consequences* part of a model fragment is a list of atomic formulae, which is interpreted as an implicit conjunction. Consequences cannot contain embedded conditions.

**Consequence elimination** removes selected consequences from model fragments. As with condition elimination, consequence elimination may occur as a necessary result of quantity elimination. This category includes replacement of a consequence by another that is strictly weaker than the original.

**Consequence approximation** replaces a relation in the consequence of a model fragment with a simpler relation. For instance, one might replace a complex equation with a simpler, approximate equation. Examples of such approximations include equilibration, exogenization (Iwasaki & Simon 1994), and piecewise linear approximation.

Equilibration is applicable to the consequence equations of one model fragment representing some mechanism that restores equilibrium much quicker than other mechanisms in the system. In this situation, one can regard the fast mechanism as acting instantaneously. The equilibration operation replaces a dynamic equation representing a fast mechanism by its respective equilibrium equation. For example, using a model fragment for a pressure sensor, one might assume that `(= sensed-pressure pressure-reading)`. This assumption treats the reading of the pressure on the sensor as an instantaneous process, giving the pressure at the current instant. Determining the `pressure-reading` is considered a fast mechanism compared to other processes in the model. Similarly, exogenization—replacement of a slow mechanism by a constant equation—is also a type of consequence approximation.

Piecewise linear approximation may be used to simplify complex equations given in the consequences of a model fragment. In piecewise linear approximation, one approximates a non-linear equation by pieces each of which can be approximated by a linear equation. [5]

**Consequence aggregation** combines a set of consequences into an aggregate. For instance, two consequences of the model fragment representing a heat exchanger are the conservation of mass equations for the hot and cold flows. In an abstract model, one might combine them into a single conservation of mass equation. Consequence aggregation decreases the number of equations in a model fragment, but the new consequences may be weaker than the conjunction of the original consequences.

## Model Fragment Abstraction

Methods of abstracting model fragments include eliminating selected model fragments and aggregating a set of model fragments into one model fragment. Model fragment abstraction is often motivated by knowledge

[5] If the formal definition of a model fragment does not allow conditional consequences as it is not in CML, each linear piece will have to be represented by separate model fragments. This is a rare case where abstraction of one model fragment results in a set of model fragments.

of their structural, functional, or statistical relationships.

**Model fragment elimination** removes the model fragments representing certain elements in the problem description. Often, functional or statistical knowledge motivates model fragment elimination. For instance, consider a device which contains a secondary valve as a backup in cases of failure of its primary valve. One may choose to eliminate the model fragment for the secondary valve, based on the knowledge that the secondary valve functions solely as a backup. Statistical knowledge about a population of individuals may also motivate model fragment elimination. If it is known that 99.9% of a certain population of carbon molecules have the molecular weight of 24, one may choose to eliminate all model fragments representing carbon molecules having other weights.

**Model fragment aggregation** replaces a set of model fragments with an aggregate model fragment. Again, functions and population statistics are common bases for performing such an abstraction. Aggregation reduces the total number of model fragments and allows one to ignore the unnecessary details.

Structural or functional aggregation replaces a set of model fragments representing components by a model fragment that represents their structural or functional super-component. Aggregation of a nearly decomposable dynamic system (Simon & Ando 1961) is an example of model fragment aggregation if we view each mechanism represented by an equation in the original system as a model fragment.

Amador and Weld discuss population abstraction in (Amador & Weld 1990). Amador and Weld model population systems on three levels: the individual level, the aggregate level, and the macro level. The aggregate level contains a collective description of individual level properties. Applying statistical operators, such as `Summation`, `Mean`, `Max`, and `Min`, to the attribute values of individuals in the population generates probabilistic descriptions of the individuals. The modeler uses these statistical operators on all the attributes of all the members of a population to create a representative model fragment. For instance, the attribute values of a representative may be the summation of the attribute values for its individual members.

## Example

We illustrate our abstraction framework with a simple example of model fragments representing a hot air balloon. Figure 1 gives the formal specification for the model fragment `hot-air-balloon`.

The hot air balloon is composed of an `envelope` (the balloon), `burner` (to heat the air in the envelope), `basket`, `temperature-sensor1`, `temperature-sensor2`, `pressure-sensor1`, and `pressure-sensor2`. The sensors measure the conditions within the envelope. The quantities `temperature1`, `temperature2`, `pressure1`, and `pressure2` indicate the readings from the respective sensors. Table 2 specifies abstraction methods that are performed on `hot-air-balloon` and the abstracted representational element. For each abstraction, the table indicates the portion of the original representation that is abstracted and specifies its abstract representation. The abstract model fragments are named to correspond with the graph of model fragments in Figure 2. The graph illustrates the relationship between the original and abstract representations of `hot-air-balloon`. The `hot-air-balloon` is an aggregate model fragment, representing the collection of its component model fragments.

```
(defModelFragment hot-air-balloon
   :quantities
        (temperature1, temperature2,
         pressure1, pressure2,
         balloon-temp, balloon-pres,
         balloon-vol, balloon-mass,
         compressibility-factor,
         universal-gas-const)
   :conditions
        ((< (Abs (- temperature1 373K))
              1e-5)
         (< (Abs (- temperature2 373K))
              1e-5)
         (< pressure1 101.3kPa)
         (< pressure2 101.3kPa))
   :consequences
        ((= balloon-temp
         (Avg temperature1 temperature2))
        (= balloon-pres
         (Avg pressure1 pressure2))
        (= balloon-vol
         (/ (*  compressibility-factor
                universal-gas-const
                balloon-temp
                balloon-mass)
                balloon-pres))))
```

Figure 1: The `hot-air-balloon` Model Fragment

## Discussion

The framework categorizes abstraction relations among model fragments according to the type of transformation performed on one model fragment to produce another model fragment.

It would be ideal if one could make general statements about the representational consequences of applying these transformations on model fragments using such general theories of abstraction as the one proposed by Giunchiglia and Walsh or the one proposed by Struss. Unfortunately, it is impossible to simply apply either of these general theories to abstractions of

| Abstraction Type | Original Representional Elements | Abstract Representational Elements | Abstract Model Fragment Name |
|---|---|---|---|
| Quantity Elimination | `temperature1`<br>`temperature2` | `temperature1` | **balloon-with-single-temp-sensor** |
| Quantity Aggregation | `temperature1`<br>`temperature2` | `avg-temp` | **balloon-with-sensor-averager** |
| Condition Elimination | `(< pressure1 101.3kPa)`<br>`(< pressure2 101.3kPa)` | `(< pressure1 101.3kPa)` | **balloon-with-single-pres-cond** |
| Condition Approximation | `(< (Abs`<br>`(- temperature1 373K)`<br>`1e-3))` | `(= temperature1 373K)` | **balloon-with-approx-temp-cond** |
| Condition Aggregation | `(< pressure1 101.3kPa)`<br>`(< pressure2 101.3kPa)` | `(< (+ pressure1`<br>`pressure2)`<br>`202.6kPa)` | **balloon-with-agg-pres-cond** |
| Consequence Approximation | `(= balloon-vol`<br>`(* compressibility-factor`<br>`universal-gas-const`<br>`balloon-temp`<br>`balloon-mass`<br>`(inv balloon-pres)))` | `(= balloon-vol`<br>`(* quantity-contained`<br>`ideal-gas-const`<br>`balloon-temp`<br>`(inv balloon-pres)))` | **balloon-assuming - ideal-gas** |
| Model Fragment Aggregation | `envelope`<br>`burner`<br>`basket`<br>`temperature-sensor-t1`<br>`temperature-sensor-t2`<br>`pressure-sensor-p1`<br>`pressure-sensor-p2` | `hot-air-balloon` | **original-hot-air-balloon** |

Table 2: Abstraction of the `hot-air-balloon` Model Fragment

model fragments. Giunchiglia and Walsh's theory compares two logical theories and does not easily extend to comparisons among model fragments. Performing the types of transformations discussed in this paper on a model fragment produces a new set of model fragments. A knowledge base in the compositional modeling framework is a set of model fragments and does not constitute a consistent logical theory. A knowledge base can contain logically inconsistent model fragments though different subsets of the knowledge base may give rise to logically consistent models that are useful in different situations. Struss' theory also does not serve our purpose since his theory presumes a complete model, and in our case, we need a way to characterize differences among potentially relevant fragments of models before they are assembled into a complete model.

Nevertheless, if we changed our problem and made strong assumptions that (1) we are starting from a set of model fragments that constitute a complete and logically consistent model, and (2) an abstraction operation performed on one model fragment will be performed consistently throughout the set, we could make the following general observations:

- Elimination generally results in TD abstraction. It falls in the category of simplification as defined by Struss but is not necessarily a representational transformation[6] since the result may no longer be a model (Struss 1991).
- Approximation is neither TD nor TI. It is a representational transformation, and is a type of simplification as defined by Struss.
- Aggregation is TD. It is a representational transformation.

[6]Intuitively, a transformation is a representational transformation if it results in a model that covers all the situations covered by the original model; however, it may make different kinds of distinctions from the original.

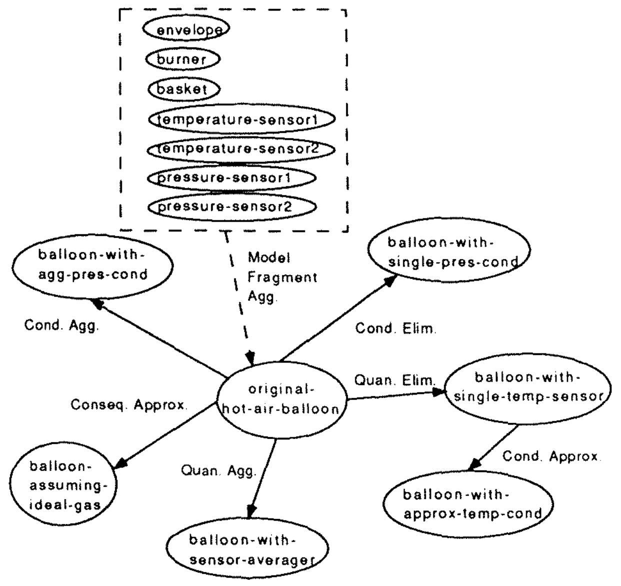

Figure 2: Graph of Model Fragments

Our immediate motivation for this work is to expand our model formulation method by using this framework to precisely characterize a broader class of abstraction relations among model fragments. Our next step is to analyze various types of assumptions underlying the transformations discussed in this paper. We must then decide how to represent such assumptions, what additional information can be specified about the query, and how to reason with them in order to choose an appropriate level of abstraction for each model fragment.

Further work is needed to predict the computational consequences of abstraction in terms of the effects they have on the results of behavior prediction. One observation that can be made is that abstraction of the representation as discussed in this paper does not necessarily result in a simpler prediction. In particular, condition abstraction generally results in a weaker condition, which could result in more model fragments becoming activated during simulation.

Though our immediate goal is to incorporate the abstraction framework into the model formulation mechanism and use further analysis results to guide model fragment selection, this framework is an important step towards the automatic generation of abstract model fragments. The proposed framework has the advantage of being easy to operationalize once one selects the aspect of the representation to abstract. We plan to explore this possibility of automatic abstraction of model fragments in the future.

## Acknowledgments

The first author was supported by the National Physical Science Consortium Fellowship.

This research was sponsored by the Advanced Research Projects Agency, ARPA Order 8607, monitored by NASA Ames Research Center under grant NAG 2-581; and by NASA Ames Research Center under grant NCC 2-537.

## References

Amador, F. G., and Weld, D. 1990. Multi-level modeling of populations. In *Proceedings of the Fourth International Workshop on Qualitative Physics.*

Amarel, S. 1981. On representations of problems of reasoning about actions. In Webber, B. L., and Nilsson, N. J., eds., *Readings in Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann.

Falkenhainer, B., and Forbus, K. 1991. Compositional modeling: Finding the right model for the job. In *Artificial Intelligence*, Vol. 51, pp. 95–143.

Farquhar, A.; Bobrow, D.; Falkenhainer, B.; Fikes, R.; Forbus, K.; Gruber, T.; Iwasaki, Y.; and Kuipers, B. 1993. A compositional modeling language. Knowledge Systems Laboratory Technical Report KSL-93-53, Stanford University, Stanford, California.

Fikes, R. E.; Hart, P.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. In *Artificial Intelligence*, 251–288.

Giordana, A., and Saitta, L. 1990. Abstraction: A general framework for learning. In *Working Notes of AAAI-90 Workshop on Automatic Generation of Approximations and Abstractions.*

Giunchiglia, F., and Walsh, T. 1992. A theory of abstraction. To appear in the Journal of Artificial Intelligence.

Iwasaki, Y., and Levy, A. 1994. Automated model selection for simulation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

Iwasaki, Y., and Simon, H. 1994. Causality and model abstraction. In *Artificial Intelligence.* to appear.

Knoblock, C. A. 1989. A theory of abstraction for hierarchical planning. In Benjamin, P., ed., *Proceedings of the Workshop on Change of Representation and Inductive Bias.* Boston, Mass: Kluwer.

Knoblock, C. A. 1990. Learning abstraction hierarchies for problem solving. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann.

Levy, A. Y. 1993. *Irrelevance Reasoning in Knowledge Base Systems.* Ph.D. Dissertation, Stanford University, Stanford, CA.

Minton, S. 1988. *Learning Search Control Knowledge: An Explanation Based Approach.* Academic Publishers, Hingham, MA.

Plaisted, D. 1981. Theorem proving with abstraction. In *Artificial Intelligence*, Vol. 16, pp. 47–108.

Plaisted, D. 1986. Abstraction using generalization functions. In *Proceedings 8th Conference on Automated Deduction*, 365–376.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. In *Artificial Intelligence*, Vol. 5, pp. 115–135.

Simon, H. A., and Ando, A. 1961. Aggregation of variables in dynamic systems. In *Econometrica*, volume 29.

Struss, P. 1991. A theory of model simplification and abstraction for diagnosis. In *Working notes of the 5th International Workshop on Qualitative Reasoning*.

T. M. Mitchell, R. M. Keller, S. K.-C. 1990. Explanation-based generalization: A unifying view. In Shavlik, J. W., and Dietterich, T. G., eds., *Readings in Machine Learning*. San Mateo, CA: Morgan Kaufmann. 435–451.

Williams, B. C. 1991. Critical abstraction: Generating simplest models for causal explanation. In *Working notes of the 5th International Workshop on Qualitative Reasoning*.