International Institute for
Applied Systems Analysis
IIASA www.iiasa.ac.at

Dimensional Consistency Analysis in Complex Algebraic Models

Nastase, V., Makowski, M. and Michalowski, W.

IIASA Interim Report
2007

International Institute for
Applied Systems Analysis
Schlossplatz 1
A-2361 Laxenburg, Austria

Tel: +43 2236 807 342
Fax: +43 2236 71313
E-mail: publications@iiasa.ac.at
Web: www.iiasa.ac.at

Interim Report **IR-07-029**

Dimensional Consistency Analysis in Complex Algebraic Models

Vivi Nastase (vnastase@site.uottawa.ca)
Marek Makowski (marek@iiasa.ac.at)
Wojtek Michalowski (wojtek@telfer.uottawa.ca)

Approved by

Leen Hordijk (hordijk@iiasa.ac.at)
Director, IIASA

December 2007

Foreword

Research described in this report was conducted as part of the activities covered by the Memorandum of Understanding signed between the International Institute for Applied System Analysis and the University of Ottawa, Canada and partially funded by the ICRA Grant from the University of Ottawa. Through such international collaborative effort the researchers were able to propose novel methodology to supporting development of complex numerical equations involving multiple units of measurement.

This research builds on the OntoEng methodology for engineering mathematics that assists in developing semantically consistent complex formulae. It focuses on a method of representing knowledge about a specific model structure enhanced with the ability to account for non-standard representations. Such methodological research should help IIASA researchers involved in the development and use of the SMT Modeling System to handle in an easier manner semantic consistency of large-scale environmental models.

This collaborative research involved scientists from the School of Management and Telfer School of Information Technology and Engineering at the University of Ottawa, and the Atmospheric Pollution and Economic Development Program and the Integrated Modeling Environment Project at the International Institute for Applied Systems Analysis.

Abstract

Relations in complex algebraic models include numerous variables and parameters that capture the physical dimensions of the objects represented in models (such as *mass*, or *volume* of an object). A model developer must ensure the semantic correctness of the model, which includes consistency across physical dimensions and their units of measure in the model relations. Such dimensional consistency analysis is the subject of the research described in this paper.

We propose a new methodological framework for this type of analysis which comprises:

- a two-level structure for representing knowledge about physical dimensions and units of measure; and
- the dimensional analysis algorithm that uses this structured knowledge for the verification of consistency.

The proposed methodology allows us to resolve issues related to handling complex non-decomposable units of measure and the situation when instances of the same physical dimension are associated with different physical quantities. We illustrate the proposed methodological framework using mathematical relations from a comprehensive environmental model developed at IIASA.

Keywords: analysis of algorithms, algebraic models, dimensional analysis, units of measure, structured modeling.

Acknowledgments

The research described in this paper was supported by the Collaborative Research Grant from the University of Ottawa while Dr. Vivi Nastase was a post-doctoral fellow at the University.

The authors wish to thank Dr. Fabian Wagner of IIASA for providing detailed information about the new versions of the RAINS model, and for sharing his ideas about the topics related to the research described in this paper. The authors would also like to thank Dr. Szymon Wilk, and two anonymous reviewers for their comments and suggestions.

About the Authors

Vivi Nastase is currently a research scientist at EML Research gGmbH, Heidelberg, Germany. Vivi's research interests include computational linguistics, applications of machine learning and data mining techniques to language and interactions in negotiations and numerical analysis. She has published papers in journals including Group Decision and Negotiation, and computational linguistic and negotiation conferences. She was a guest editor for two journal special issues, for Computational Intelligence and Group Decision and Negotiation. Vivi is currently working on incorporating encyclopedic knowledge in natural language processing systems.

Marek Makowski is a leader of IIASA Integrated Modeling Environment Project. His research interests focus on model-based support for solving complex problems, which incorporates three interlinked areas. First, integration of interdisciplinary knowledge and its representation by mathematical models. Second, creation of knowledge by comprehensive model analysis, including multicriteria methods. Third, tailoring the modeling process to meet the needs of decision-making processes.

Thus Marek's research interests cover a cluster of areas relevant to the adaptation (whenever possible) or development (when needed) of methodology, algorithms, and software for model-based decision-making support. This includes more specific topics in Operations Research (OR) such as: multicriteria problem analysis, large scale optimization, optimization of badly conditioned problems, use of database management systems for complex models, decision analysis and support, user interfaces in decision support systems, effective treatment of uncertainty and risk.

Marek has published over 80 papers and book-chapters, coordinated or led several scientific projects, and has been twice guest editor of the European Journal of Operational Research.

Wojtek Michalowski is University Research Chair in Health Informatics and Decision Support at the Telfer School of Management, adjunct professor in the Faculty of Medicine, University of Ottawa, member of the Ottawa-Carleton Institute for Computer Science, and adjunct research professor in the Eric Sprott School of Business, Carleton University. He is past director of the Master of Health Administration program. During 1997/1998 academic year he was senior research scholar at the International Institute for Applied Systems Analysis.

Wojtek's research interests include health informatics and specifically ubiquitous and mobile clinical decision support systems, health care management, and data mining. He has written over seventy refereed papers and has published articles in some thirty journals, including Management Science, Naval Research Logistics, Operations Research, Journal of Optimization Theory and Applications, IEEE Systems, Man and Cybernetics, International Journal of Medical Informatics, IEEE In-

telligent Systems, Decision Support Systems, Methods of Information in Medicine, Healthcare Management Science, and European Journal of Operational Research.

Wojtek is a Principal Investigator of the MET Research Program aiming at developing clinical decision support environment for the point-of-care applications.

Contents

List of Tables

List of Figures

Dimensional Consistency Analysis in Complex Algebraic Models

Vivi Nastase[*] *(vnastase@site.uottawa.ca)*
Marek Makowski[**] *(marek@iiasa.ac.at)*
Wojtek Michalowski[***] *(wojtek@telfer.uottawa.ca)*

1 Introduction

Dimensional consistency analysis is a conceptual framework for the verification of correctness of mathematical relations from the point of view of the physical dimensions and units of measures associated with the relations' variables and parameters. (Bridgman, 1922) wrote that such analysis *is routinely used by physical scientists and engineers to check the plausibility of derived equations and computations. It is also used to form reasonable hypotheses about complex physical situations that can be tested by experiment or by more developed theories of the phenomena.*

Dimensional consistency analysis has its formal roots in the works of Fourier from the 1820's (see (Macagno, 1971) for a historical overview), and by the beginning of the 20th century, it had established solid foundations through the works of Bridgman (1922) and Buckingham's π-theorem (Buckingham, 1914). Dimensional consistency analysis has found numerous applications in very diverse fields (see (Sonin, 2001) for an overview). This paper describes an application of dimensional consistency analysis during development of complex algebraic models. In particular, we demonstrate how to verify the consistency of a model's relations, from the point of view of units of measure associated with the variables and parameters appearing in these relations.

Algebraic models are abstract representations of real life objects and phenomena that use entities (constants, variables, parameters) representing physical quantities and various types of relationships among them (functions, equalities, inequalities). Algebraic model complexity comes from two sources: the number of entities and relationships between them, and the complexity of the entities' physical dimensions in terms of the corresponding units of measures. An additional difficulty is introduced by the fact that models are often developed by interdisciplinary teams, working remotely on different model's components, making it more difficult to maintain semantic consistency. Consistency in the use of units of measure is critical for a model's usability, but maintaining it is a tedious task that is difficult to perform manually. Moreover, such consistency has to be assured at each stage of the modeling process, including:

[*]School of Information Technology and Engineering, University of Ottawa, Canada.
[**]Integrated Modeling Environment Project, IIASA.
[***]Telfer School of Management, University of Ottawa, Canada.

1. Symbolic model specification, involving declarations of model entities (parameters, variables), and relations among them;

2. Preparation and verification of data to be used for defining model parameters;

3. Generation and analysis of model instances (composed of the model specification and a specific set of data);

4. Interpretation of the results of model analysis.

In this research we focus on a methodological framework that supports the first two stages of modeling process, where the dimensional consistency issues that must be addressed are difficult and no practical solution is yet available for models having many complex units of measures.

We stress that we have not attempted to develop any new language nor an ontology for supporting dimensional consistency analysis. The experience with development of interdisciplinary models motivated one of the authors to develop the Structured Modeling Technology (SMT) (Makowski, 2005), which allows to create model specification without knowledge of any modeling language. Therefore we propose to follow a similar approach while defining the measurement units. After a set of measurement units is specified by modelers, the dimensional consistency will be used to check the syntax correctness of model relations. In the paper we introduce some formalism in order to present in a concise manner the related concepts to be used in the implementation.

The paper aims at reaching two goals:

- First, to provide the modelers with effective tools for reconciling complex measurement units without a requirement to know any specific modeling language. It proposes a theoretical framework allowing modelers to easily define appropriate set of the units and structure them implicitly using template-like or inheritance mechanism.

- Second, to provide an overview of diverse approaches to handling measurement units and dimensional consistency, and to propose an approach that can be implemented in modeling systems in a way that is transparent to modelers. Most of the widely used modeling systems do not support dimension consistency analysis (cf Section 2.4). Therefore the proposed approach may be of interest also to the developers of modeling environments.

The remaining part of the paper is organized as follows: Section 2 summarizes the modeling context, and provides an overview of previous research on dimensional consistency analysis, also known as *unit of measure reconciliation*. Sections 3 and 4 are the core of the paper and introduce proposed knowledge representation for PDs and UMs, and the algorithm that uses this knowledge for dimensional consistency analysis. We justify the need for, and illustrate the application of proposed methodology in Section 5, using the environmental RAINS model as a case study. In Section 6 we discuss issues of potential implementation of the proposed solution. Section 7 wraps up the paper with conclusions and lessons learned.

2 Modeling context

The modeling process of a complex decision problem is very different from traditional approaches to mathematical modeling, when a person (or a small team) develops a model. In the traditional approach it is possible to assure consistency of relations and data by a proper communication between team members. However, complex problems are modeled by multidisciplinary teams often working at distant locations, on different submodels that are tested separately before being amalgamated. These submodels are developed by people with diverse professional backgrounds (e.g., in engineering, economics, atmospheric chemistry, environment, health) and describe different problem domains. In each of these domains typically different units of measure are used to represent models' entities. Moreover, data used for instantiating model parameters is coming from diversified sources (often being generated by solving other models) thus input data is also recorded using a variety of units of measure. Large models are typically composed of submodels developed by several teams, therefore assuring semantic consistency of all relations is no longer as easy as it was for small models developed by a team. Moreover, large models use millions of data items, typically available from diverse sources and requiring conversions to make the measurement units of original data to become consistent with the units used in the model symbolic specification.[1]

Therefore, especially for complex models it is essential to assure:

1. Semantic correctness of relations defined in model specification, which includes consistency of the units of measure;

2. Consistency between units used in model specification, and those used in input data.

Modeling processes supporting policy making have to meet the strong requirements of credibility, transparency, replicability of results, integrated model analysis, controllability[2] of the modeling process, quality assurance, documentation, controllable sharing of modeling resources through the Internet, and efficient use of resources on computational grids. These requirements cannot be met when models are developed using general-purpose modeling tools. This was also the situation faced by the developers of a family of Regional Air Pollution Information and Simulation (RAINS) models (see e.g., (Amann and Makowski, 2000)) created to support international negotiations on air pollution and policy development. Therefore, to support the modeling process for RAINS and other complex models, a specialized Web-based modeling environment called Structured Modeling Technology (SMT) (Makowski, 2005) was developed. SMT employs a methodology proposed by Geoffrion, (see (Geoffrion, 1987)) that relies on the concept of **entity** to manipulate various model elements (parameters, constants, variables, relations). Such an approach makes it relatively easy to handle units of measure associated with parameters and variables, because they are represented as one of the features each entity has.

Although the SMT provides a conceptual framework to deal with units of measure in a conceptually simple way, it is actually challenging to apply it in practice, mainly because

[1] An illustration of complex measurement units is provided in Section 5. A more detailed discussion of issues related to the development of large and/or complex models can be found in (Makowski, 2005).

[2] This term covers several interlinked elements of modeling process, including modification of the model specification and the data defining the model parameters, various views on data (often having complex indexing structure), and interactive analysis of results (obtained for different model instances and/or representations of user preferences).

entities in large models have rather complex measurements. Thus, a rational approach to handling units of measure requires a systematic methodology and this requirement constitutes the main motivation behind the research reported in this paper.

2.1 Dimensions and units of measure

Entities in a model – variables, constants, parameters, relations – represent measures of **physical quantities** (PQs) of objects and phenomena of the world, for example the earth's diameter, a person's age, etc. The measure of a physical quantity is a **physical dimension** (PD) – length, mass, etc. – expressed as a numerical value, or **magnitude** measured with a help of some **unit of measure** (UM). Normally, a certain set of PDs and their UMs are chosen as the set of **base** or primary PDs or UMs, in terms of which all the other PDs and UMs – **derived** PDs and UMs – are defined. Usually, the base PDs and UMs are the fundamental dimensions and their units are defined by the International System of Units (SI)[3] – length, mass, time, electric current, thermodynamic temperature, amount of substance, luminous intensity, angle, solid angle. Sometimes, for convenience or to emphasize certain properties of a model, a different set of base PDs and UMs can be chosen. However, considering that a physical dimension can be measured in many different ways, we say that all units of measure for a given physical dimension form an **equivalence class** (for example: g, $\pounds$, oz and their multiples represent *mass*). Each UM in an equivalence class is related to some base unit for the associated dimension through a **scaling factor**.

Throughout the paper, we call **decomposable** PDs and UMs the derived PDs and UMs, that for the purpose of dimensional consistency analysis must be represented in terms of their base components. **Non-decomposable** PDs and UMs are those which despite being derived must be treated as base PDs and UMs, respectively. When referring to a PD throughout this paper we assume that its associated UM is defined. Particular features of the UM (scaling factors for example) are explicitly mentioned whenever relevant.

2.2 Physical dimensions in complex algebraic models

Algebraic models often combine data from diverse fields and different PDs customarily used in these fields are associated with the model's entities. Although they are derived from the SI system, the diversification of PDs makes it difficult to check if all components of each model relation are specified consistently, and if the data used for instantiation of parameters of these relations is correct from the point of view of the UMs. This situation creates a need for automatic analysis to ensure that the model is dimensionally consistent and the data used to instantiate the model matches the entity declarations.

A good example of difficulties in ensuring dimensional consistency comes from air pollution modeling where, for example, the nitrogen-oxides ozone transfer coefficient is measured in:

$$\frac{mg \times hours}{m^3 \times kt}.$$

Despite the fact that a mass quantity appears both in the numerator and the denominator it cannot be reduced because one expresses the mass of ozone, and the other the mass of

[3] `http://www.bipm.fr/en/si/`.

nitrogen oxides. This exemplifies a situation where similar PDs are differentiated by the PQ they apply to. Another example is the general PD representing *pollutant emissions*; it is defined for each pollution type: sulphur oxides (SO_x), nitrogen oxides (NO_x) and ammonia (NH_3). These emissions interact during their travel from the emission sources to deposition places (conventionally called grids) and the resulting environmental effects are measured by various indicators in each grid. In air-quality models different entities represent different elements of the problem, for example, emission levels (of the considered types of pollution), transfer coefficients, air-quality indicators, relations between the emissions and the indicators, to name a few. Each of these entities is described by a PD. Semantic correctness of the model requires that each relation including these entities is consistent from the point of view of the physical dimensions of its terms.

Another dimensional consistency issue is related to non-decomposable PDs. For example, $force$ is a derived PD ($force = mass \times acceleration$) and its UM $newton$ (N) can be expressed using base UMs as $N = kg \times m \times s^{-2}$. However, there are models where such a decomposition of N is not feasible. We need to stress that a distinction between decomposable and non-decomposable PDs can only be made by a modeler based on his/her knowledge about the content in which the PD is used. Therefore a modeling system should provide the corresponding functionality.

Considering the often complicated specification of PDs in algebraic models, it is essential to support model developers in defining them.

2.3 General framework

We propose a general methodological framework comprised of a method of representing knowledge about PDs and the dimensional analysis algorithm that exploits this structured knowledge as a way to verify the dimensional consistency of relations in algebraic models.

We propose to represent knowledge about PDs on two levels. The first level, called *generic*, is used to define PDs. The second level, called *specific*, contains instances of the PDs from the generic level augmented with specific properties needed for defining features of model entities. For example, a PD called *pollution emission* belongs to the generic level, while emissions of specific pollutants, such as *sulphur oxides, nitrogen oxides* and *ammonia* belong to the specific level. Splitting PD information along these two levels facilitates PD definition, operation on PDs with multiple instances associated with different PQs, and performing dimensional analysis, as we will demonstrate later.

The **generic level** forms the "core" of a knowledge representation. It includes each generic dimension relevant to a model[4] together with the corresponding PDs. Having defined such generic level we impede redundant definitions. In the discussion that follows we will call PDs from generic level as the *gPDs* (generic PDs). The gPDs are linked with each other according to their definitions. A simple example is shown in Figure 1.

The **specific level** contains instances of gPDs that express dimensions of objects in the model. Because the gPD is already defined, obtaining an instance (iPD) requires only adding information specific to the iPD in the form of:

attribute – which associates the iPD with a specific world object. For example, by adding an attribute *sulphur oxides* to the gPD *pollution emission*, we obtain the iPD *sulphur*

[4]By *generic* we mean that there are no distinctions related to particular world objects the dimension may refer to, or scaling factors within its unit of measure.

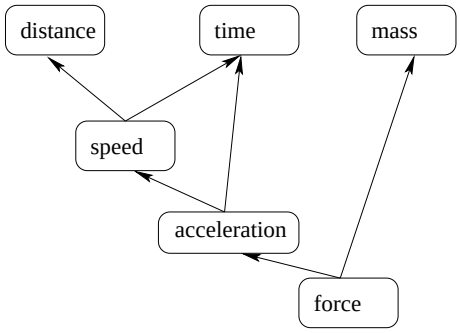

Figure 1: Example of gPDs. A directed edge from gPD_i to gPD_j signifies that gPD_j appears in gPD_i's definition.

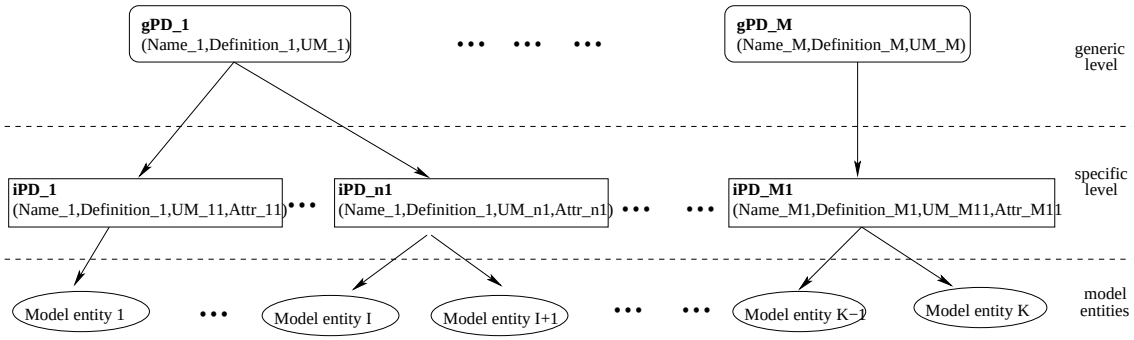

Figure 2: Example of knowledge representation of physical dimensions on two levels, with model entities linked to dimensions from the specific level

oxides emissions;

scaling factor – which is used to adjust the UM of a particular iPD, to reflect the UMs used in practice.

Figure 2 illustrates the proposed two-level knowledge representation, where each gPD has several possible instantiations, each with different attributes, possibly different UMs (from the same equivalence class); iPD instances are then captured as dimensions of the model entities.

Considering that knowledge of the PDs and UMs in the domain to be modeled sits with the domain experts, the representation we propose should be, and indeed can be, easily edited by the model developers. From this point of view, structuring the knowledge of PDs on two levels, accomplishes the following:

1. Reduces the amount of work necessary for defining PDs for a model:
   - gPDs are defined using previously defined gPDs;
   - gPD definitions are reused for each iPD specification. This process is described in detail in Section 3.
2. Enables reuse of knowledge about PDs. This knowledge can be structured as a specialized resource that can be shared at different levels: for closely related models,

the entire structure can be reused (generic level, specific level and entity definitions), while for more loosely coupled models only knowledge from the generic level can be reused.

The proposed two-level knowledge representation facilitates:

1. Distinction of the iPDs associated with different objects by assigning different attributes (see the example on pollution modeling in Section 2.2).

2. Distinction between decomposable and non-decomposable PDs: non-decomposable PDs have a null definition in terms of simpler dimensions.

3. Projection of all PDs onto the set of base PDs specific for a model.

Ultimately it facilitates the application of the proposed dimensional analysis algorithm which is based on (Bridgman; Uschold et al.; Novak, Jr., 1922; 1998; 1995).

2.4 Related work

We review the literature from the perspective of dimensional consistency analysis. We look first at stand-alone methods that solve the UM reconciliation problem, and then describe solutions incorporated within programming languages. We show how these solutions do not address all PD consistency issues that are present in complex algebraic models. Finally we discuss approaches that incorporate dimensional consistency analysis within modeling environments, and those that use an external knowledge representation, like we do.

An example of a stand-alone method for handling UMs is COMET written in APL, which performs conversions between the British and the metric measurement systems (Schulz, 1990). Also Bhargava (1993) proposed a very interesting solution, independent of a particular system or programming environment, which transforms dimensional consistency analysis into numerical analysis. In this approach each base PD is coded through a prime number, and consistency check is done by comparisons of the corresponding products of prime numbers.

A lot of effort has gone into incorporating knowledge about UMs into programming languages. Karr and Loveman (1978) discuss methods of unit conversion, dimensional consistency analysis and language syntax issues related to incorporating units into programming languages. Hilfinger (1988) describes a method for implementing a package for dimensional analysis in ADA, using the language's facilities for abstraction and extension. He explains the use, implementation and potential efficiency of a package for four basic PDs, but the method is not flexible enough to accommodate domain-specific UMs. Cunis (1992) describes an implementation in Lisp for converting units, where formula verification and conversions are performed at runtime. Novak, Jr. (1995) proposes an implementation of UMs as part of data types, also in Lisp, accompanied by algorithms to perform conversions and simplifications of combinations of units.

Kennedy (1994) presents an extension of a strongly-typed programming language, ML with the notion of dimension type. Expressions are checked in a manner similar to detecting programming errors, before runtime, and the type of an expression can be inferred, using term and relation unification. Denis (2001) describes Unum, a dimensional

consistency solution implemented in Python. Allen et al. (2004) present a method of incorporating UMs in object-oriented programming languages.

The approaches summarized above, while providing suitable methodological background, are separate from model development environments, i.e., are not integrated in general purpose modeling languages and the corresponding tools, such as GAMS (Bisschop and Meeraus, 1982), or AMPL (Fourer et al., 1990). These environments support the modeling process, thus freeing the users from writing problem-specific model generators and low-level interactions with optimization solvers. Each tool has a specific modeling language that supports model specification and data handling. However these tools do not effectively support handling of UMs. They assume that it is the responsibility of model developers to use consistent UMs for all model entities, and typically do not support specification of the units and checking consistency of relations from the UM/PD perspective. In a model using entities with simple units this is doable but for complex models, delegating the dimensional consistency verification to a model developer results in a time-consuming and error prone task.

Among the widely used modeling environments, AIMMS (Bisschop and Roelofs, 2006) is the only one that provides extensive support for handling UMs. The type of dimensional consistency analysis implemented in AIMMS can be termed as unit reconciliation – it is based on analysis of the UMs associated with entities, as opposed to analysis of the PDs. AIMMS has a list of built-in PDs and their associated UMs. During model specification, the model developer must a priori declare the PDs that will be used and the chosen UM. If a standard PD is used with a derived UM (for example km instead of the base m), the user must define the scaling factor to transform the derived UM into the base one, because the system maintains a representation of UMs in terms of base units. When writing a new relation, the system checks if the UMs agree. AIMMS allows the model developer to define her own dimensions and units, but it does not automatically recognize standard dimensions, their UMs and scaling factors. Therefore all PDs used in a model developed with AIMMS must be defined as a part of the model specification.

When developing complex models a possibility of reusing a repository of previously defined PDs would ease the work load of a model developer. *OntoEng* (Gruber, 1995), (Gruber and Olsen, 1994), illustrates the use of an external resource for dimensional consistency analysis, as it has an ontology of PDs and their UMs. The ontology was designed as an external resource for engineering mathematics to support formula verification from the point of view of UMs. The UMs sub-ontology was subsequently revised and extended (Pinto and P. Martins, 2001). Uschold et al. (1998) describe an application using the UM sub-ontology of *OntoEng* for solving a panel layout problem and explore the cost of reusing *OntoEng* and enhancing it with the necessary functionality suitable for such optimization problems.

Having a representation of PDs as an external knowledge representation embedded in a model development environment provides a solution that greatly eases the development process. The external resource includes knowledge about PDs that can be reused to avoid repeated definitions of the same dimensions and units. The representation of knowledge should be flexible enough to distinguish similar PDs pertaining to different objects, and to allow for PDs that are both decomposable and non-decomposable.

3 Knowledge representation of physical dimensions and their units of measure

3.1 Requirements for representation of physical dimensions

In general, there are three important requirements to be satisfied by a representation of knowledge about PDs. We discuss each requirement together with the proposed solution.

Requirement 1:

Ability to distinguish between PDs describing the same type of physical phenomenon or quantity, while using knowledge about their commonalities. For example, emissions of NO_x and SO_x are both instances of *pollution emissions* that must be distinguishable in a model.
Proposed solution: Have a two-level representation of knowledge about PDs:

- The **generic level** contains definitions of distinct generic PDs. A gPD is represented as the tuple $(Name, UM, Definition)$. For the example above, the gPD is labeled as *pollution emission*.
- The **specific level** contains instances of gPDs. The iPDs inherit the definition from the gPD they specialize. iPDs also have an attribute field, which serves to distinguish it from any of the other iPDs of the same gPD. An iPD is represented as the tuple:
  (Name,UM,Definition,Attribute).
  The name of an iPD can be different than its gPD's name, to reflect better the iPDs characteristics. In the example above, NO_x *emission* is an instance of the *pollution emission* with the attribute NO_x.

In the proposed methodology we rely on the generic level of knowledge representation and gPD definitions as the basis for the dimensional consistency analysis. UMs are automatically generated using these definitions. The dimensional analysis algorithm operates on generic representations based on PDs and their definitions.

Requirement 2:

Ability to easily define derived PDs.
Proposed solution: A new gPD (denoted by $PD.G$) is defined as a function of previously defined gPDs:

$$PD.G = f(PD_1, \ldots, PD_k)$$

where f is an algebraic function composed of multiplication and division operators. Considering that division is the inverse of multiplication, we can write an equivalent definition of $PD.G$[5]:

$$PD.G = PD_1^{exp_1} \times \ldots \times PD_k^{exp_k}$$

[5]In the following discussion we use the following abbreviations: exp to refer to the $Exponent$, Sf to refer to the $ScalingFactor$, $attr$ to refer to the $Attribute$.

where $exp_i \in R$ are real numbers. If

$$\{PD_{B1}, \ldots, PD_{Bn}\}$$

is the set of base PDs chosen for the model, then by replacing gradually each PD_i with it's simpler components, we arrive at an expression of $PD.G$ in terms of base PDs:

$$PD.G = PD_{B1}^{exp_{B1}} \times \ldots \times PD_{Bn}^{exp_{Bn}}$$

The definition of $PD.G$ can be written as a vector:

$$< (PD_{B1}, exp_{B1})(PD_{B2}, exp_{B2}) \ldots (PD_{Bn}, exp_{Bn}) >$$

where the exponents of PDs that are not required for in $PD.G$'s definition are 0. In this vector notation, PD_x stands for the name of the x-th PD.

In order to completely capture UM information in a PD definition, we need to include an extra component – a scaling factor – that allows for the specification of the correct unit of measure. The scaling factor is an absolute number (e.g., 1000 to indicate tons when kilograms is the chosen UM for mass, or 0.7503 to indicate Euros when the chosen UM for a cost is US Dollars, if 0.7503 is the applied exchange rate).

The revised definition of $PD.G$ becomes:

$$PD.G =< (PD_{B1}, exp_{B1}, Sf_{B1})(PD_{B2}, exp_{B2}, Sf_{B2}) \ldots (PD_{Bn}, exp_{Bn}, Sf_{Bn}) >$$

Knowing the UMs associated with each PD_{Bi}, the UM for $PD.G$ is:

$$UM.A = UM_{B1}^{Sf_{B1} \times exp_{B1}} \times \ldots \times UM_{Bn}^{Sf_{Bn} \times exp_{Bn}}$$

Returning back to $PD.G$'s original definition, it can be rewritten as:

$$PD.G =< (PD_1, exp_1, Sf_1)(PD_2, exp_2, Sf_2) \ldots (PD_k, exp_k, Sf_k) >$$

where exp_i combines the exponents of the base PDs that participate in the definition of PD_i. The same is true for the Sf_i.

If $PD.I$ is an instance of the generic $PD.G$, $PD.I$ inherits $PD.G$'s name, UM, definition, and is augmented by an extra field of attributes. The name and UM can be changed to reflect specific characteristics of the $PD.I$.

Following the representation of $PD.G$ as a vector, $PD.I$ is defined as:

$$PD.I =< (PD_1, exp_1, Sf_1, attr_{S1})(PD_2, exp_2, Sf_2, attr_{S2}) \ldots (PD_k, exp_k, Sf_k, attr_{Sk}) >$$

where a vector element $(PD_i, exp_i, Sf_i, attr_{Si})$ represents an instance of $(PD_i, exp_i, Sf_i$ in the definition of the generic PD. The attribute $attr_{Si}$ is used to distinguish different instances of the same gPD, and to constrain compatible combinations of iPD in a relation. For example, an attempt to combine an iPD with the attribute SO_x with an iPD with the attribute NH_3 is semantically incorrect, and therefore should not be allowed.

The model developer should provide the attribute, or it can be automatically inherited from instances of components of the gPD. We explain this process later in this section.

The model developer can also modify the scaling factors, to reflect the UM requirements associated with each PD instance.

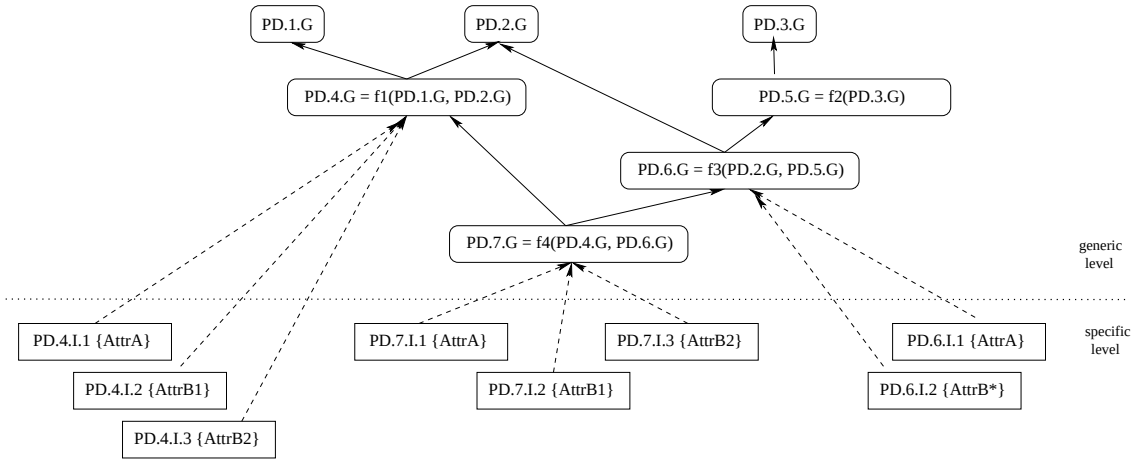

Figure 3: Representation of generic and specific levels of knowledge

Requirement 3:

A derived PD can be treated as decomposable and non-decomposable.
Proposed solution: Duplicate the entry for such a gPD in the knowledge representation, with the non-decomposable PD having a similar name as the decomposable one, but a null definition.

Thus, if PD_{dual} is a PD that can be either decomposable ($PD_{dual}1$) or non-decomposable ($PD_{dual}2$) depending on the circumstances, the representation contains the following entries:

$$\begin{array}{lll} (PD_{dual}1 & UM & < (PD_1, exp_1, Sf_1) \ldots (PD_k, exp_k, Sf_k) > \quad) \\ (PD_{dual}2 & UM & \text{—} \qquad\qquad\qquad\qquad\qquad\quad) \end{array}$$

Note that the UM is the same for both PDs. A null definition for $PD_{dual}2$ makes it automatically non-decomposable and part of the set of base dimensions. Base dimensions, together with PD definitions are the essential elements of the dimensional consistency analysis algorithm.

3.2 Knowledge representation

The two-level representation of knowledge is illustrated in Figure 3. The generic PDs ($PD.i.G$) form the generic level, $i \in \{1, \ldots, N\}$, and are represented by their name and definition in relation to other gPDs in rounded rectangles. Their instances ($PD.i.I.j$) form the specific level, and are represented by their name and (optional) attributes[6] (in curly brackets) in regular rectangles. A dimension $PD.i$ has a projection on the generic level, $PD.i.G$ and as many as the model requires on the specific level, $PD.i.I.j$.

As a model representation of a real world object or phenomenon each model entity has a physical dimension which is represented by an iPD composed of the definition of the corresponding physical dimension, the UM, and iPD's attributes.

A gPD is defined using previously defined gPDs. If these previously defined dimensions already have instance PDs, they are automatically combined to generate instances for the newly defined gPD. Attributes of iPDs determine which combinations are valid.

[6]The symbol $*$ is a wild card, and stands in for any other symbol or combination of symbols.

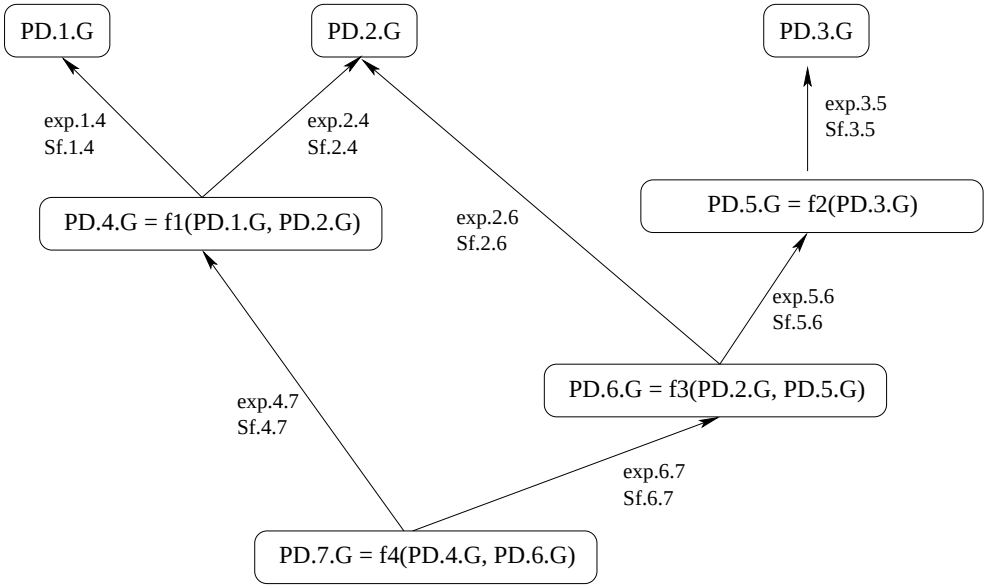

Figure 4: Directed graph representing the connectivity between generic PDs

In the example presented in Figure 3, both $PD.4.G$ and $PD.6.G$ have instances with different attributes. $PD.7.G$ combines $PD.4.G$ and $PD.6.G$. More precisely:

$$PD.7.G = f_4(PD.4.G, PD.6.G) \rightarrow < (PD.4.G, exp_4)(PD.6.G, exp_6) > .$$

When creating a new PD from existing ones, we create instances for this new PD using attributes to control the process. In the case of creating instances for $PD.7.G$, we use the instances for $PD.4.G$ and $PD.6.G$ (see Figure 3):

$$PD.7.I.* \rightarrow < (PD.4.I.*, exp_4, Sf_4, attr_{4.*})(PD.6.I.*, exp_6, Sf_4, attr_{6.*}) >$$

There are six possible instances for $PD.7.I.*$, corresponding to the six pairs of $(PD.4.I.*, PD.6.I.*)$:

$PD.7.I.1 \rightarrow < (PD.4.I.1, exp_4, Sf_4, AttrA)(PD.6.I.1, exp_6, Sf_6, AttrA) >$
$PD.7.I.2 \rightarrow < (PD.4.I.2, exp_4, Sf_4, AttrB1)(PD.6.I.2, exp_6, Sf_6, AttrB*) >$
$PD.7.I.3 \rightarrow < (PD.4.I.3, exp_4, Sf_4, AttrB2)(PD.6.I.2, exp_6, Sf_6, AttrB*) >$
$PD.7.I.4 \rightarrow < (PD.4.I.1, exp_4, Sf_4, AttrA)(PD.6.I.2, exp_6, Sf_6, AttrB*) >$
$PD.7.I.5 \rightarrow < (PD.4.I.2, exp_4, Sf_4, AttrB1)(PD.6.I.1, exp_6, Sf_6, AttrA) >$
$PD.7.I.6 \rightarrow < (PD.4.I.3, exp_4, Sf_4, AttrB2)(PD.6.I.1, exp_6, Sf_6, AttrA) >$

Assuming that attributes $AttrA$ and $AttrB*$ do not match, only the first three combinations are feasible, as constrained by the attribute values.

The definition of PD captures the interconnectivity between complex and base PDs. Thus, it is possible to visualize the relations between the gPDs as a directed acyclic graph (see Figure 4). The directed edges connect a complex PD with its simpler components. We can attach information to the edge (such as exponent) to represent how a gPD and its components are connected. If $PD.G = f(PD_1, \ldots, PD_n) = PD_1^{exp_1} \times \ldots \times PD_n^{exp_n}$, exp_i and/or Sf_i can be associated with the edge connecting $PD.G$ and PD_i.

In the next section we discuss the role of the attributes as part of the vector representation of PDs in dimensional consistency analysis.

4 Dimensional consistency analysis

Dimensional consistency analysis of an algebraic model involves two steps:

1. Consistent definitions of complex PDs.

2. Checking dimensional consistency of model relations.

We propose to implement these two steps using a common methodological framework.

From the model developer perspective, consistency analysis involves the UMs associated with model's entities. However, it is possible to approach the analysis from the more general perspective of PDs, as this allows for an easier implementation of a dimensional analysis algorithm. This is the approach we adopt.

Bridgman (1922) proposed a method of obtaining a dimensional formula of any PD in terms of PDs that are chosen as base, and shows how to change from a system of units with one set of base PDs, to a system of units with a different set of base PDs. This research inspired Novak, Jr. (1995) and Uschold et al. (1998) to use a vector representation of derived PDs and to perform dimensional consistency analysis using vector operations. The set of base PDs, $\{PD_{Bi}, i = 1, n\}$ is fixed. Derived PDs are defined in terms of ever simpler dimensions until the level of base PDs is reached. Each position in the vector representation of a derived PD is a tuple (exp_{Bi}, Sf_{Bi}) representing the exponent corresponding to the base dimension PD_{Bi}, and the scaling factor relative to the base UM for PD_{Bi}. An entity in a model is expressed through its corresponding PD, and the scaling factor Sf with respect to the base UM for that PD: $< PD, Sf >$. Projecting this onto the vector of n base PDs, the entity is represented by the vector

$$< (exp_{B1}, Sf_{B1})(exp_{B2}, Sf_{B2})...(exp_{Bn}, Sf_{Bn}) >$$

For example, if there were 4 PDs: $length, mass, time, money$, a length of km is expressed as a vector: $< (1, 10^3)(0, 0)(0, 0)(0, 0) >$, when m is considered the base UM for $length$[7]. The acceleration of m/s^2 is represented as $< (1, 1)(0, 0)(-2, 1)(0, 0) >$ if m is the base unit for length, and s for time.

Each base PD has a specific position (index) in the vector representation. For instance, index 1 in the example above is assigned to length, index 2 to mass, and so on. This works well in situations when the base PDs are known in advance. However, when a non-decomposable PD with a complex UM must be treated as a base PD, then the vector representation has to be dynamic because the list of base dimensions is known only after completing the model definition.

Considering the requirements outlined in Section 3, the iPD is represented as:
(Name, Exponent, ScalingFactor, Attributes).

Having PD_{name} ensures that vector operations are performed correctly even when the list of dimensions is known only at runtime. The values of *Attributes* constrain operations to feasible combinations of compatible entities. The exponent and the scaling factor serve for consistency verification, as described in (Novak, Jr., 1995) and (Uschold et al., 1998).

[7]Note that we don't specify a number of km, as the amounts corresponding to model entities do not impact dimensional analysis

This representation is further transformed, using PD's definition, into a vector representation containing only base PDs:

$$(PD_{name}, exp, Sf, attr) \rightarrow < (PD_{B1}, exp_{B1}, Sf_{B1}, attr_{B1}) ... (PD_{Bn}, exp_{Bn}, Sf_{Bn}, attr_{Bn}) >$$

where PD_{Bi} are base PDs necessary for the definition of PD_{name}.

Definition of derived PDs (i.e., composed of several base PDs to which either multiplication or division operators are applied) requires a multiplication operator.

In order to simplify the notation we denote by $PDInfo_x$ the tuple $(PD_x, exp_x, Sf_x, attr_x)$ for the dimension PD_x. Then the **multiplication** operator is defined as:

$$\begin{array}{rl} & < PDInfo_{i1}, PDInfo_{i2}, ..., PDInfo_{in} > \\ \times & < PDInfo_{j1}, PDInfo_{j2}, ..., PDInfo_{jm} > \\ \hline = & < PDInfo_{k1}, PDInfo_{k2}, ..., PDInfo_{kp} > \end{array}$$

The result of the multiplication of two PDs is defined in two steps:

1. $PDInfo_k$ is composed of elements of $PDInfo_x$ and $PDInfo_y$
2. If definitions of two PDs of $PDInfo_k$ contain the same dimension PD_k with the same scaling factor Sf_k and attribute $attr_k$, then such two elements are replaced by one with the exponent being the sum of PD_k's exponents. If the resulting exponent is equal to zero, then the corresponding element is eliminated from the $PDInfo_k$, (see the example in Section 5.2).

The described methodology provides consistent definition of complex PDs that are often composed of other PDs combined with multiplication and/or division operators. Now, we will show how the same approach can be used for the analysis of dimensional correctness of model relations.

Model relations can be considered as composed of terms (expressions) to which the following operators are applied: $=$, $+$, $-$, $\leq$, $<$, $\geq$, $>$. In other words, a term is composed of entities (parameters and/or variables) to which either the multiplication or the division operators is applied. Thus, determining the PD of a term is equivalent to defining a derived PD from its simpler components.

The dimensional consistency of a relation requires that all terms of a relation have the same PD. This can be easily verified by comparing the vector representations of the term PDs. An example of such consistency analysis is presented in Section 5.2.

5 RAINS model

To illustrate the application of the proposed methodology, we use selected relations from the RAINS models developed at the International Institute for Applied Systems Analysis (IIASA)[8]. These models combine information on economic and energy development, emission control potentials and costs, atmospheric dispersion characteristics and environmental sensitivities towards air pollution (Amann et al., 2004), (Schöpp et al., 1998).

[8]The name *RAINS* covers a family of algebraic models. The example discussed here comes from the Greenhouse Gas and Air Pollution Interactions and Synergies (GAINS) model.

PD	Unit	Type
mass	kg	generic
time	s	generic
length	m	generic
substance	mol	generic
area	$m2$	generic
emission	kt/year	generic
particulate emission	kt/year	instance
SO_x emission	kt/year	instance
NO_x emission	kt/year	instance
NH_3 emission	kt/year	instance
Equivalence coefficient	mol of charge	generic
eq($H+$)	mol of charge	instance
eq($N-$)	mol of charge	instance
pollution level	eq(_)/(ha*year)	generic
eutrophication level	eq(N^-)/(ha*year)	instance
acidification level	eq(H^+)/(ha*year)	instance
transfer coefficient	eq(_)/(ha*year*kt/year)	generic
SO_x transfer coefficient	eq(H^+)/(ha*year*kt/year)	instance
NO_x transfer coefficient	eq(N^-)/(ha*year*kt/year)	instance
NH_3 transfer coefficient	eq(N^-)/(ha*year*kt/year)	instance

Table 1: Examples of complex PDs and their UMs

One of the reasons for developing the SMT system was to support checking semantic consistency of complex models, and the complexity of the RAINS models illustrates the need of such support. The entities in RAINS models have complex, and often non-standard, PDs. A sample of these PDs is presented in Table 1.

5.1 Describing physical dimensions and units of measure

In order to define the PDs given in Table 1 in such a way that model entities can be associated with them, a model developer should be able to define first the base PDs *mass, time, length, substance,* and then combine them to define the derived gPDs: *particulate emission* and *equivalence coefficient*. Finally, one can define *pollution level* and *transfer coefficient*. A schematic view of the relations between the PDs from Table 1 is presented as a directed acyclic graph in Figure 5[9].

A new gPD is defined using existing PDs. For example, the transfer coefficients k_{ij} (from i-th source to j-th receptor) are defined by:

$$k_{ij} = \frac{\partial x_j}{\partial y_i},$$

where x and y stand for the corresponding pollution level, and pollution emission, respectively.

[9] In order to make the figure easier to read, we include only the names of the PDs.

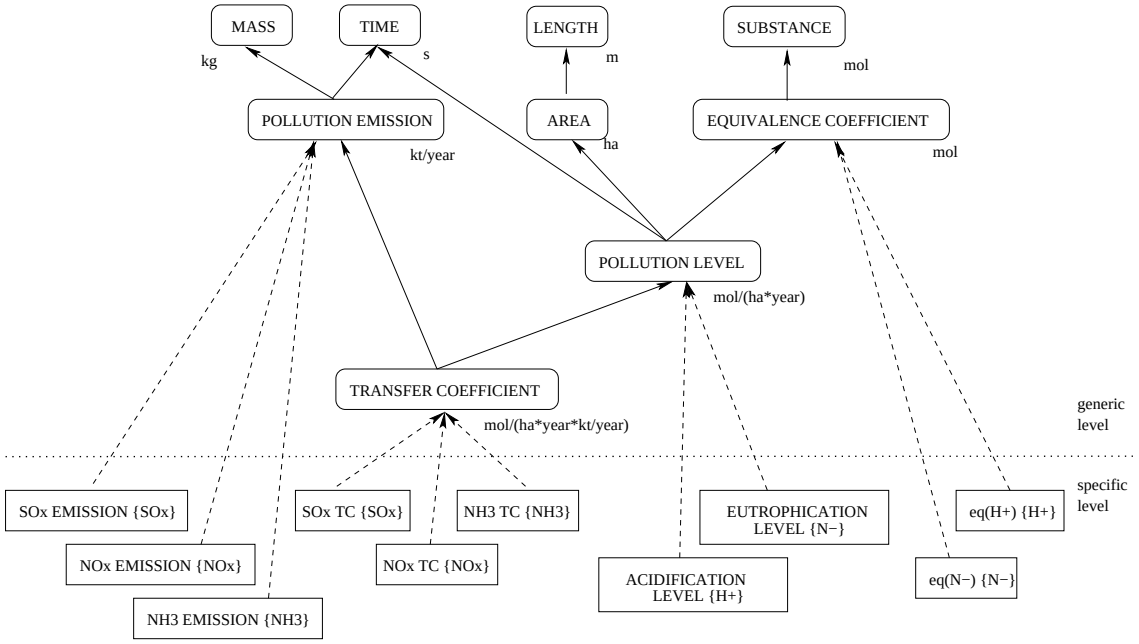

Figure 5: Directed graph representing gPDs and iPDs

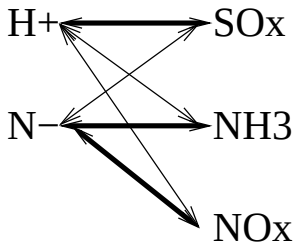

Figure 6: Illustration of compatibilities for transfer coefficients.

The corresponding knowledge representation contains the following information:

$$\begin{bmatrix} \text{Name} & \text{: transfer coefficient} \\ \text{UM} & \text{:} mol \times ha^{-1} \times year^{-1} \times kt^{-1} \\ \text{Definition:} & \text{pollution level} \times \text{ pollution emission}^{-1} \end{bmatrix}$$

The UM is formed using PD's definition, by combining the UM's of PD's components, as shown in Section 3.

In order to use a transfer coefficient in a model, the model developer first defines instances of transfer coefficient. For the example shown in Figure 5 and Table 1, the relevant instances are NO_x *transfer coefficient*, SO_x *transfer coefficient* and NH_3 *transfer coefficient*.

As explained in Section 3, attributes are used to constrain the generation of iPDs from their corresponding gPD definition. In the case study, the *pollution level* has two PD instances – eutrophication (with attribute N^-) and acidification (with H^+); *pollution emission* has three PD instances with the attributes SO_x, NO_x and NH_3 respectively.

In order to automatically obtain the three transfer coefficients for the three gases the model developer defines the following compatibilities: (N^-, NO_x), (N^-, NH_3), (H^+, SO_x), see Figure 6. Now we can use the definition of the *transfer coefficient* in terms of *pollution level* and *pollution emission,* and the attributes of instances of *pollution level* and *pollution emission* to determine the instances of *transfer coefficient*:
transfer coefficient = pollution level $\times$ pollution emission^{-1}

The compatible attribute pairs allow three instances for the *transfer coefficient* gPD: NO_x *transfer coefficient,* NH_3 *transfer coefficient* and SO_x *transfer coefficient*.

Formally, the external knowledge representation is specified as described in Table 2 in the Appendix. The model developer can expand this description, keeping in mind that when defining a new iPD, the corresponding gPD must be defined first.

5.2 Dimensional consistency analysis algorithm

We illustrate the operation of the dimensional analysis algorithm using a simple relation from RAINS models:

$$somo35_k = \sum_{i \in I} tno_{ik} * n_i + \sum_{i \in I} tvo_{ik} * v_i + ko_k, \qquad k \in IR \tag{1}$$

where k is the grid index, IR is the set of grids (where the environmental impact is measured), and I is the set of indices associated with countries/regions.

Relation (1) combines the following entities:

- $somo35_k$: measures ozone (O_3) impact in k-th grid. The UM of $somo_k$ is defined as $mg \times hours \times m^{-3}$, and it has the following vector representation in terms of base PDs[10]:

$$< (mass, 1, 10^{-3}, [O_3])\ (time, 1, 3600, [])(length, -3, 1, []) > \tag{2}$$

- tno_{ik}: measures the O_3NO_x transfer coefficient in $mg \times hours \times m^{-3} \times kt^{-1}$. Its vector representation in terms of base PDs is:

$$< (mass, 1, 10^{-3}, [O_3])\ (time, 1, 3600, [])\ (length, -3, 1, [])\ (mass, -1, 10^{9}, [NO_x]) > \tag{3}$$

- n_i: measures the annual emission of NO_x, in kt. Its vector representation in terms of base PDs is:

$$< (mass, 1, 10^{9}, [NO_x]) > \tag{4}$$

- tvo_{ik}: measures the O_3VOC transfer coefficient in $mg \times hours \times m^{-3} \times kt^{-1}$. Its vector representation in terms of base PDs is:

$$< (mass, 1, 10^{-3}, [O_3])\ (time, 1, 3600, [])\ (length, -3, 1, [])\ (mass, -1, 10^{9}, [VOC]) > \tag{5}$$

- v_i: measures the VOC emission in kt. Its vector representation in terms of base PDs is:

$$< (mass, 1, 10^{9}, [VOC]) > \tag{6}$$

[10] In the vectors that follow, the symbol [] denotes an empty list of attributes.

- ko_k: measures background O_3 accumulation in $mg \times hours \times m^{-3}$. Its vector representation in terms of base PDs is:

$$< (mass, 1, 10^{-3}, [O_3]) \ (time, 1, 3600, []) \ (length, -3, 1, []) > \qquad (7)$$

These entities form four terms in relation (1). It is easy to show that all these terms have the same PDs:

- $somo35_k$: defined by (2);
- $tno_{ik} \times n_i$: defined by multiplication of PDs (3) and (4); note that

  $< (mass, 1, 10^9, [NO_x]) > \times < (mass, -1, 10^9, [NO_x]) > = < (mass, 0, 10^9, [NO_x]) >$

  therefore this sub-term does not contribute to the term's PD;
- $tvo_{ik} \times v_i$: defined by multiplication of PDs (5) and (6).
- ko_k : defined by (7).

The above demonstrates that relation (1) is dimensionally consistent.

6 Implementation issues

The proposed methodology for representing and using knowledge about PDs and UMs can be implemented as an SMT resource, consisting of a structured knowledge repository that supports the model developer in extending the set of commonly used PDs by the PDs specific to a model (or a set of models from an application domain). New PDs are defined in terms of existing ones, and knowledge of UMs and scaling factors is automatically expanded to cover newly defined PDs.

6.1 Building a repository of knowledge about dimensions and units

An important implementation issue is related to creating and editing knowledge about PDs and their UMs. From a user perspective, this process should be easy and as error-free as possibly. In particular, many typing errors in PD definitions can be avoided if a model developer follows a structured process in which the PD components of new definitions are chosen from automatically generated lists.

Because of the two-level organization of the knowledge representation, there are two steps of PD definitions: one for gPDs, and one for the iPDs.

Defining a generic PD

A definition of a gPD can be expressed as a product of previously defined PDs with real-number exponents. Therefore adding a PD definition is easy. We rely on the vector representation of a PD:

$PD.G \leftarrow < (PD_1, exp_1) \ ... \ (PD_n, exp_n) >$

Such a vector can be defined from pairs $(PD_i, exp_i), i = 1, \ldots, n$ provided by the model developer. One can also change the UM associated with PD_i so that the final UM

obtained as a product of individual UMs with the appropriate exponents reflects the model developer's preferences for a particular dimension.

The knowledge is initialized with information about the base PDs and UMs defined in the SI. UM equivalence classes are associated with their corresponding PDs, and each UM in the class has an associated scaling factor that relates it to the base UM for the class.

When defining a PD, the model developer chooses a previously defined PD from a list, inputs a new integer exponent, and chooses a UM from the equivalence class for this PD. After this, one can continue the process by adding more PDs to the definition, or close the definition.

Adding an instance to a gPD

The model developer first selects a gPD from a list. By such a choice the gPD is defined (and cannot be changed). The corresponding UM however can be changed by adding scaling factors, and attributes that constrain possible iPD combinations and distinguish between them.

As described in Section 3, PD instances can be automatically created using PD definitions, thus greatly reducing the burden on the model developer.

Homonymous attributes for different iPDs match by default. Attributes which have different names, but should be considered compatible within a certain model (such as H^+ and SO_x) must be defined as compatible by a model developer.

Specifying model entities

Physical dimension and the corresponding UM of each model entity is defined by a corresponding PD instance. Selection of the desired iPD can be easily implemented, for example through browsing a list. A structured knowledge representation can help this process by allowing a hierarchical access to the PD repository: the user can select first from a (smaller) list of gPDs, and then choose one of its instances. This knowledge representation can be managed through a separate interface, supporting grouping of PDs into semantically-motivated clusters (e.g. economic, financial, environmental) to further ease the definition process when the list of generic PDs becomes too large.

6.2 Dimensional consistency analysis algorithm

The dimensional consistency algorithm, as opposed to the knowledge repository, is hidden from the model developer. The dimensional analysis is incorporated into the model parser which processes the model specification. Therefore, the dimensional inconsistencies are treated in a similar way as other semantic errors in the model specification. SMT has an efficient model parser therefore model developers are supported in interactive model specification and analysis. The approach proposed in this paper builds on an efficient vector representation of PDs, therefore dimensional analysis can be done quickly and requires only a small part of the (small) computational resources needed for parsing model specification.

6.3 Object-oriented implementation

SMT is implemented using the object-oriented programming paradigm. Therefore, the dimensional consistency analysis can be implemented by two new classes, and extensions of the two existing classes.

The new **gPD class** implements the structure and methods for handling generic PDs. The structure of an object contains fields specific for dimensions, including *Name, Definition, UM*. The methods in this class are designed to:[11]

- manage the corresponding gPD definition,
- build the UM from the definition,
- build the vector representation.

The new **iPD class** implements the structure and methods for handling PD instances. The structure of an object in this class follows PD definition and includes: *Name, Definition, UM, Attribute*. The methods in this class are designed to:

- make the link to objects of the gPD class such that the definition, UM and vector representation can be automatically acquired from the corresponding object,
- interact with the model developer to acquire iPD's attributes,
- interact with the model developer to allow for changing of scaling factors, as necessary,
- recreate the vector representation to incorporate the new scaling factors and attributes.

The **Entity class** handling the data structure and methods for model entities now manages also the corresponding iPD, which in turn handles all necessary information about the dimension and the corresponding units of measure.

The **Relation class** (inherited from the *Entity class*) manages the data structure and methods for handling model relations. The relation class handles information about the entities involved, and the relations between them. A new method in this class implements the dimensional consistency analysis algorithm. This ensures that the defined relation is consistent from the point of view of the PDs (and therefore also UMs) it combines. Any discovered inconsistencies are treated similarly to handling other syntax or semantic errors in a model specification.

7 Conclusions

We have presented a methodological framework for addressing two specific issues related to dimensional consistency in complex algebraic models: (i) existence of complex physical dimensions which can be decomposable or non-decomposable depending on the modeling context; (ii) existence of physical dimensions of the same type associated with different objects.

[11] The actual implementation may reveal other necessary methods.

The proposed framework is composed of a structured knowledge representation describing physical dimensions and their associated units of measure that the model developer can create and edit in the course of model creation, and an algorithm for performing dimensional consistency analysis that uses this structured knowledge. The essential elements of the knowledge representation include the definition of physical dimensions and their units of measure in terms of simpler elements, and specification of the attributes associated with specific dimensions. These elements enable representation of any physical dimension as a vector, wherein each element represents the contribution of a base dimension to the definition of the new one, including exponent, scaling factor and attributes. The attributes capture information about the specific object to which a physical dimension is associated, allowing consistent dimensional simplifications. The vector representation supports the easy implementation of dimensional consistency verification.

We illustrated the proposed framework using examples from the RAINS family of models and showed how it applies to the analysis of consistency of complex dimensions and units of measure. Our goal is to implement this framework within the SMT modeling environment and assess its performance and usability not only for RAINS models, but also for other complex algebraic models.

References

Allen, Eric, David Chase, Victor Luchangco, Jan-Willem Maessen, Jr. Guy L. Steele. 2004. Object-oriented units of measurement. *OOPSLA '04: Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. Vancouver, BC, Canada, 384–403.

Amann, Markus, Janusz Cofala, Chris Heyes, Zbigniew Klimont, Reinhard Mechler, Max Posch, Wolfgang Schöpp. 2004. The RAINS model. IIASA – Documentation of the model approach prepared for the RAINS peer review 2004.

Amann, Markus, Marek Makowski. 2000. Effect-focused air quality management. A. Wierzbicki, M. Makowski, J. Wessels, eds., *Model-Based Decision Support Methodology with Environmental Applications*. Series: Mathematical Modeling and Applications, Kluwer Academic Publishers, Dordrecht, 367–398.

Bhargava, Hemant K. 1993. Dimensional analysis in mathematical modeling systems: A simple numerical method. *ORSA Journal on Computing* **5** 33–39.

Bisschop, Johannes, Alexander Meeraus. 1982. On the development of a general algebraic modeling system in a strategic planning environment. *Mathematical Programming Study* 1–29.

Bisschop, Johannes, Marcel Roelofs. 2006. *AIMMS Language Reference*. Lulu Press, The Netherlands.

Bridgman, Percy William. 1922. *Dimensional Analysis*. Yale University Press.

Buckingham, Edgar. 1914. On physically similar systems: Illustrations of the use of dimensional analysis. *Physical Review* 345–376.

Cunis, Roman. 1992. A package for handling units of measure in lisp. *ACM Lisp Pointers* **5**.

Denis, Pierre X. 2001. Unum, numbers with units in python. *Data Systems In Aerospace, DASIA - 2001*. home.tiscali.be/be052320/Unum.html.

Fourer, Robert, David Gay, Brian W. Kernighan. 1990. A mathematical programming language. *Management Science* **36** 519–554.

Geoffrion, Arthur. 1987. An introduction to structured modeling. *Management Science* **33** 547–588.

Gruber, Thomas. 1995. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* **43** 907–928.

Gruber, Thomas, Greg Olsen. 1994. An ontology for engineering mathematics. *Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, Bonn, Germany, 258–269.

Hilfinger, Paul N. 1988. An ada package for dimensional analysis. *ACM Transactions of Programming Languages Systems* **10** 189–203.

Karr, Michael, David B. Loveman. 1978. Incorporation of units into programming languages. *Communications of the ACM* **21** 385–391.

Kennedy, Andrew. 1994. Dimension types. Donald Sannella, ed., *Programming Languages and Systems, 5th European Symposium on Programming (ESOP '04)*, vol. 788. Springer, Edinburgh, U.K., 348–362.

Macagno, Enzo O. 1971. Historico-critical review of dimensional analysis. *Journal of the Franklin Institute* **292** 391–40.

Makowski, Marek. 2005. A structured modeling technology. *European Journal of Operations Research* **166** 615–648.

Novak, Jr., Gordon S. 1995. Conversion of units of measurement. *IEEE Transactions on Software Engineering* **21** 651–661.

Pinto, Helena Sofia, Jo ao P. Martins. 2001. Revising and extending the units of measure "subontology". *IJCAI-01 Workshop on the IEEE Standard Upper Ontology*. AAAI Press, Seattle, Washington, USA, 43–50.

Schöpp, Wolfgang, Markus Amann, Janusz Cofala, Chris Heyes, Zbigniew Klimont. 1998. Integrated assessment of european air pollution emission control strategies. *Environmental Modelling and Software* **14** 1–9.

Schulz, Charles A. 1990. Writing applications for uniform operation on a mainframe or pc: A metric conversion program. *APL Quote Quad* **20**.

Sonin, Ain A. 2001. *The Physical Basis of Dimensional Analysis*. 2nd ed., http://web.mit.edu/2.25/www/pdf/DA_unified.pdf.

Uschold, Mike, Mike Healy, Keith Williamson, Peter Clark, Steven Woods. 1998. Ontology reuse and application. *Proceedings of the International Conference on Formal Ontology and Information Systems - FOIS'98*. IOS Press, Amsterdam, NL, 179–192.

A A sample of PDs and UMs of the RAINS model entities

Level	PD	UM	Definition	Generic PD	Attributes
generic	mass	kg	–	–	–
generic	time	s	–	–	–
generic	length	m	–	–	–
generic	substance	mol	–	–	–
generic	area	m^2	$< length^2 >$	–	–
generic	volume	m^3	$< length^3 >$	–	–
generic	pollution emission	kt	$< mass >$	–	–
generic	equivalence coefficient	mol	$< substance >$	–	–
generic	pollution level	$mol \times ha^{-1} \times year^{-1}$	$< equivalence\ coefficient, area^{-1}, time^{-1} >$	–	–
generic	transfer coefficient	$mol \times ha^{-1} \times year^{-1}$	$< pollution\ level, pollution\ emission^{-1} >$	–	–
generic	concentration	$mg \times m^{-3}$	$< mass, volume^{-1} >$	–	–
generic	accumulation	$mg \times m^{-3} \times hours$	$< concentration, time >$	–	–
generic	O_3 transfer coefficient	$mg \times m^{-3} \times hours \times kt^{-1}$	$< accumulation, particulate\ emission^{-1} >$	–	–
...					
specific	SO_x emission	kt	$< mass >$	particulate emission	$[SO_x]$
specific	NO_x emission	kt	$< mass >$	particulate emission	$[NO_x]$
specific	NH_3 emission	kt	$< mass >$	particulate emission	$[NH_3]$
specific	VOC emission	kt	$< mass >$	particulate emission	$[VOC]$
specific	$eq(H^+)$	mol	$< substance >$	equivalence coeff.	$[H^+]$
specific	$eq(N^-)$	mol	$< substance >$	equivalence coeff.	$[N^-]$
specific	eutrophication level	$mol \times ha^{-1} \times year^{-1}$	$< equivalence\ coefficient, area^{-1}, time^{-1}, pollution\ emission^{-1} >$	pollution level	$[N^-]$
specific	acidification level	$mol \times ha^{-1} \times year^{-1}$	$< equivalence\ coefficient, area^{-1}, time^{-1}, pollution\ emission^{-1} >$	pollution level	$[H^+]$
specific	O_3 accumulation	$mg \times m^{-3} \times hours$	$< concentration, time >$	accumulation	$[O_3]$
specific	$O_3 NO_x$ transfer coefficient	$mg \times m^{-3} \times hours \times kt^{-1}$	$< accumulation, particulate\ emission^{-1} >$	O_3 transfer coeff.	$[O_3, NO_x]$
specific	$O_3 VOC$ transfer coefficient	$mg \times m^{-3} \times hours \times kt^{-1}$	$< accumulation, particulate\ emission^{-1} >$	O_3 transfer coeff.	$[O_3, VOC]$
...					

Table 2: Description of PDs and their UMs in a RAINS model