GaeaPN: A Petri Net Model for the Management of Data and Metadata Derivations in Scientific Experiments [*]

Nabil I. Hachem, Ke Qiu, Nina Serrao, and Michael A. Gennert
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA
e-mail: hachem,qiu,michaelg@cs.wpi.edu

Abstract

One important aspect of scientific data management is metadata management. Metadata is broadly defined as information about data (e.g., content, source, processing applied, precision). One kind of metadata which needs special attention is the data derivation information, i.e., how data are generated.

In our application domain of geographical information systems (GIS) and global change research, we view scientific objects according to three different extents: spatial, temporal, and derivation. Extents are dimensions along which domain users "naturally" access and select information from the database. While the spatial and temporal extents have been studied and formal semantics to those extents proposed, derivation semantics have been ignored.

In this paper, we present our own view of what metadata is and propose extensions to current database models to include implicit metadata management. We formulate a definition of extents as dimensions of metadata, and specifically discuss derivation metadata and the semantics of such specialized relationships. The proposed model, GaeaPN, is based on a variation of Petri Nets and extends current models such as the object-oriented data model. Petri nets are interpreted as a model to capture and manage data derivation relationships between scientifc data as well as the procedures and algorithms to derive data. We formulate the basic model, study its closure with respect to a "metadata space," as well as its closure with repect to database states.

The important contributions of this work are... ... We believe that this framework, useful for GIS and global change studies, generalizes well to other scientific fields.

[*]This work is supported by the National Science Foundation under Contract IRI-9116988.

1 Introduction

[I have a feeling that metadata discussion stuff should be moved to their own section or merged with the discussion of Gaea.]

There are several issues in scientific databases which make conventional database techniques insufficient to achieve the goals of data integration and data sharing [10, 16, 45]. One of the most important issues is metadata management.

1.1 What is Metadata?

For scientists, metadata is the information required to identify data of interest based on content, validity, sources, preprocessing, or other selected properties. In scientific databases, with many kinds of data stored, the associated metadata must be preserved and accessible so that the data can be meaningfully processed later.

Metadata includes such information as: Who did what and when, device characteristics, transform definitions, documentation and citations, structure and format description, precision and accuracy, etc. The list can go on like this without end. So how can scientific databases capture and manage the metadata necessary for a specific application?

First let us present our view of what is metadata? The definition widely used is – "metadata is the data about data." After looking up the defintion of data in a dictionary; that is "data is something known or assumed; facts from which conclusions can be inferred," we can say that metadata is something we know or facts about the data.

Metadata is a relative concept. It depends on what your object is or where your interest are. For example, assume we have rainfall data, and we are interested in the amount of rainfall. Then how the rainfall data is collected, where and when it is measured is metadata because this information is used for the interpretation of the rainfall measurement. However, consider for example a student database. A student record may include a social security number, name, age, major, home address, etc. Among all this information, it is difficult to say what is data and what is metadata. Furthermore, metadata for database management systems (DBMS) are quite different from those for an application. For example, in relational systems, metadata includes the number of columns in a relation, the type of each attribute, which is primary key, etc. Furthermore, for a DBMS, there is no difference between the management of data and metadata of a specific application.

[needs some work on wording]

In view of the above, one should look at metadata according to different views. For example, for the database administrator metadata is represented by the schema definition, integrity constraints, and other database structures. At the system level, metadata is represented by indexing information, and other information on data procesing algorithms and their implementations. For a scientific user, the metadata of an application appears as different relationships between the interested data and other data in the system. For example, in a computational chemistry database [8], the metadata of experimental data is represented as relationships with other data such as who performs the experiment, which molecule is the subject, what program package it used, etc.

1.2 Motivation

It seems there is nothing special about metadata management if we can create a standard set of metadata that is believed to be essential to understanding a specific database. Apparently, such metadata will vary from domain to domain. However, just storing and retrieval of metadata is not enough. A good database management system should automatically create and maintain the metadata as scientific databases are dynamic and evolutionary.

In order to see how a system can automatically create and maintain metadata, let's look at the development of a temporal database [?]. In a conventional DBMS time is not supported. Although in most cases date is supported as a data type, it is usually represented as a text string. When temporal information is needed in a application such as medical history, the programmer has to take care of the temporal "extension" of the application because a query cannot be specified declaratively to include time. For example, one can't ask the following: "find the patient's record surgery between 1970 and 1980" or "find the patient's temperature and blood pressure two day's ago." essentially the database only reflects the current status of the world, and history information is not available. In this case, temporal information is metadata.

In a temporal database, the time when the data is input into the database (transaction time) and when the data is valid (real time) are stored in the database. As the status of the database changes, those information are modified automatically. The user can make the previous queries on a temporal database [?].

1.3 The Gaea Approach

One objective of the Gaea project [?, ?, ?] is to develop a metadata manager for generic metadata, intrinsically managed by the system. Obviously, extensibility is also required for the system to be tunable to different application domains. Essentially, we promote some of the important relationships in data modeling and implement them in a DBMS environment.

In scientific databases, there are many different metadata that are very important and worth database management system support. For example, one kind of metadata which needs special attention is the data derivation information, i.e., how data are generated. We have detailed a framework for this in [Hachem93 *et al.*]. This paper focuses on the model used for that framework.

In Gaea such implicitely managed metadata are called extents. Extents represent essential information implicitely and naturaly used by the domain scientist to identify the data objects of interest. We are currently interested with three kinds of extents:

1) Temporal, 2) Spatial, and Derivation extents. We provide, in Gaea, a mechanism to automatically keep track of those kinds of information as data are created and stored in the database system.

Specifically for derived data and metadata, data is classified into two categories: base data and derived data. By base data, we mean those data obtained from well known sources outside the system. Base data are well understood and accepted by most scientists. Base data may be provided by a variety of standard agencies, government departments, research institutions, or generated by the scientists themselves. By derived data, we mean data obtained by scientists in their research by applying some algorithms on base data[1]. Unlike base data, derived data are not well understood. One important objective for the efficient management of scientific information is to be able to build on pre-existing knowledge, by sharing both base and derived data.

Consider the following simple scenario: two scientists are working on detecting the changes in vegetation index in Africa between 1988 and 1989. One may subtract the NDVI[2] of 1988 from that of 1989, while another divides the NDVI of 1989 by that of 1988. In this case, if only the resultant images are stored (as in common GIS such as IDRISI and GRASS [13, 40]), there is no way to share and compare the produced data unless the derivation procedures are known to both scientists.

[this will be replaced with what we contribute and a breakdown of the paper. What is now here is the VLDB paper stuff].

In this paper, we investigate this problem and propose a framework for the management of derived data. This framework is being implemented in the Gaea kernel, a spatio-temporal DBMS for global change research [20]. We focus on how Gaea handles metadata, and provide a general framework for the management of scientific experiments and procedures. Our contribution parallels other efforts such as [5, 8, 36], while addressing limitations of current systems such as [13, 40]. We propose to extend current semantic modeling and object-oriented technology with special constructs: concepts, processes[3], and tasks. Concepts are used to capture entity sets with imprecise definitions. A process captures the derivation procedure of a specific object class, while a task is the instance representing the derivation of a specific scientific data object. We believe that this framework, useful for GIS and global change studies, generalizes well to other scientific fields.

[need to link to other sections]

[1]One user's derived data may be another's base data. For example, cloud cover maps may be derived data for a satellite imagery scientist, but base data for a climatologist.

[2]NDVI is the normalized difference vegetation index. It is a qualitative measure of vegetation derived from AVHRR satellite imagery data.

[3]Here we use the term process to refer to its general definition as understood in the scientific community and not necessarily as perceived in the field of Computer Science.

2 The Gaea System

We provide first an overall description of the Gaea architecture which is being implemented in Phase 2 of our project, then present and discuss in some detail the different levels of management that Gaea provides for experiments in Earth Science applications. We concentrate on the different constraucts and specifically on our view of how to represent and manage derived data.

2.1 The Gaea Architecture

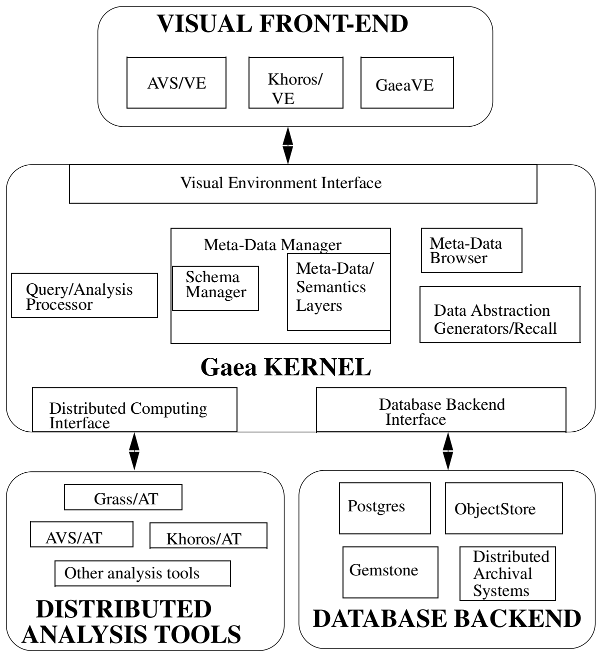


Figure 1: The Architecture of Gaea

The Gaea system architecture is designed to meet the needs of scientific research. Our view of a scientific data management and analysis environment can be layered along three levels (Figure 1): 1) The visual frontend, which allows the user to pose visual queries, apply analysis operators to data, and visualize data, including analysis results; 2) the Gaea Kernel, which provides support for meta-data, that is data about the data, and converts simple queries from the visual frontend into a complex series of database accesses and operations; and 3) The Database Backend, which actually stores the data, providing network and archiving functions. We describe each subsystem in turn.

The **Visual Frontend** mediates all interaction with the user. Our objective is to provide sufficient flexibility so that a variety of popular visual environments can be interfaced to the Gaea Kernel. There exists many such packages, either commercial (e.g., AVS [?]) or publicly available (e.g., Khoros [?]). These visual environments come with complete analysis subsystems; we would like to make use of the frontends and analysis operators separately, as shown in Figure 1. In addition, we have written our own visual frontend tailored to the Gaea Kernel [?]. One challenge on which we are currently working is the definition of a query and analysis language in which any visual query can be expressed. When that language is defined and implemented, any visual environment may be incorporated into Gaea by converting commands into the common query and analysis language.

The most important function of the **Gaea Kernel** is the management of meta-data and the semantics of derived data. This semantics is elaborated upon in Section 2.2. Users can query meta-data to obtain the meaning of derived data. Furthermore, capturing a data object's derivation process information enables the user to repeat that process and derive new data, given different input data. The kernel will include a schema manager which manages the meta-data and the associated derivation semantics and analysis operators (Figure 1). The Query/Analysis Processor (QAP) is responsible for processing queries, deriving new data whenever necessary, and using meta-data. The kernel includes a semantic and meta-data browser to allow a user to find relevant data without knowing specific file and path names. There is also a Data Abstraction Generation and Recall module which allows previously generated data to serve as a template for additional queries, i.e., queries can be abstracted. Finally, generic interfaces to the frontend visual environments and backend distributed computing and distributed databases and archives are provided.

The **Backend System** consists of distributed and archive databases such as Postgres, Object Store and Gemstone ([?], pp. 34–93). The distributed computing environment consists of scientific analysis operators which are available within commercial or public domain software systems. Examples are the analysis tools available within AVS, Khoros, and GRASS. These tools may be imported into Gaea because the meta-data manager will have registered information about analysis operators, their domains of application, data types and formats they apply to, among other meta-data. The Gaea kernel will be able to chose from these available tools and use them to provide a seamless integration between analysis and data management for scientific environments.

2.2 Meta-Data Management in Gaea

We view scientists as manipulating objects in three orthogonal extents: space, time, and data derivation (object classes). For example, in global change studies, objects have spatial as well as temporal extents. Although these two extents may be correlated, scientists retrieve and manipulate "scientific objects" by viewing those extents as orthogonal. The semantics of the spatial [14, 15, 18, 30] and temporal [2, ?, 27, 37, 39, 41] dimensions have been the subject of much research over the last decade. The third dimension, which has not received much attention so far, is the data derivation dimension, dealing with the derivation procedure followed in the

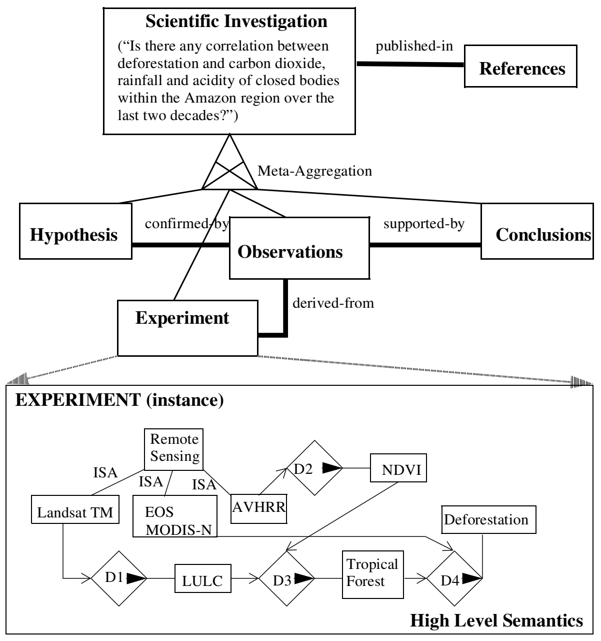


Figure 2: Generic Semantic Model for Scientific Investigations

generation of new or existing complex objects.

Meta-data management in Gaea extends semantic modeling technology [22, 31] with the following constructs [Hachem93 *et al.*]: concepts, processes, and tasks. Concepts are used to capture entity sets with imprecise definitions. A process captures the derivation procedure of a specific object class, while a task is the instance representing the derivation of a specific scientific data object. We believe that this framework, useful for GIS and global change studies, generalizes well to other scientific fields. The actual meta-data are viewed by the system at three semantic levels (Figures 2 and 3):

2.3 High Level Semantics and the Experiment Level

This level records the information that is necessary for the understanding of a specific experiment. In global change research, it is difficult to agree on carefully designed experiments. The Gaea kernel supports experiments through the experiment manager module of the metadata manager. The experiment manager is capable of manipulating conventional semantic modeling constructs [22]. In addition, we introduce the notion of *concepts*, which may either be base data or data derived from other data according to any of several well-defined algorithms.

A general definition of a **concept** is a representation of a spatio-temporal entity set, extended with an imprecise definition. Concepts are very common in scientific databases. In the context of geographical information systems and global change research, one can effectively cite many examples of concepts.

PERSON is an entity set as defined by the ER model [7], and may be considered a concept with a well defined and agreed upon meaning. But can we define what a DESERT or DESERTIC REGION is? According to [6], an acceptable definition of a desert must include consideration of the following factors: the amount of precipitation received, the distribution of this precipitation over a calendar year, the amount of evaporation, the mean temperature during the designated period, and the amount and utilization of the radiation received. Furthermore, every one of those factors may have different metrics: for example dryness, related to precipitation, can be measured by the Aridity Index, a Quotient of Dryness or the Radiational Index of Dryness [6]. So a DESERTIC REGION is an entity set whose definition may differ from one user to another.

The full semantic model for experiment management is based on *concepts* and the specialized relationship *derived-by*. Derived-by relationships connect concepts to those that are potentially used to derive them. Concepts are arranged in hierarchies, which map into sets of classes at the derivation semantics layer (Figure 3). We are in the process of extending this view to include specialized semantic entities such as scientific investigations, experiments, observations, hypotheses, conclusions, and others (Figure 2). A *scientific investigation* describes the case study being performed. In the described scenario, it captures the essence of the header and is a meta-level aggregation of entities, concepts and relationships such as: *hypothesis* confirmed by *observations* based on *experiments*. *Conclusions* are drawn from *observations*. An *experiment* is the set of tasks that are performed on data representing concepts to derive new data of other concepts. The specialized relationship *derived-by* maps into a set of processes. As there is one process that maps to a specific derived class in the derivation semantics layer, a *derived-by* relation is associated with one concept at the high level semantics layer (Figure 3). The model will include other conventional semantic constructs [22, 31], such as conventional entities, ISA hierarchies, associations and aggregation. This high-level model will provide a desktop manager for scientific investigations, based on a schema-centric view similar to [**?**]. Our long term objective is to provide an interface so that other high level managers could be integrated.

2.4 Derivation Semantics Layer

This layer provides for the management of (scientific) derivations of data. Concepts map to a set of object classes in the derivation semantics layer. Each class represents a different definition of a concept, based on a specific derivation procedure.

The derivation semantics layer records the derivation relationships among classes of data. Such relationships can also be used for the generation of new data objects for a class. Typically, when data are not stored in the database, we generate the needed data with the help of such derivation relationships. The basic constructs used are: 1) a **Process**, which captures the description of a scientific procedure used for the generation of new concepts from other concepts and 2) a **Task**, which is the instantiation of a process with input data objects. Every task will generate a set of

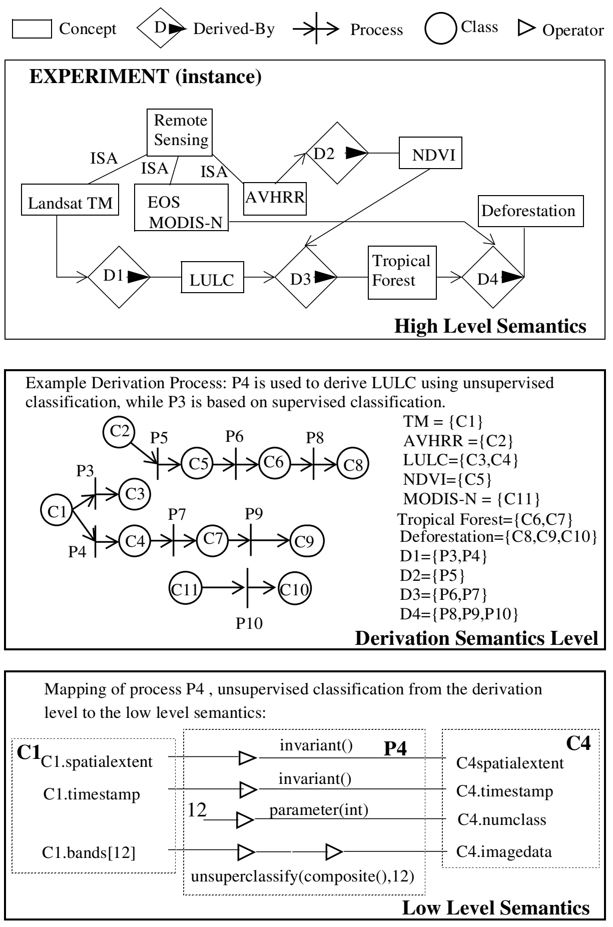


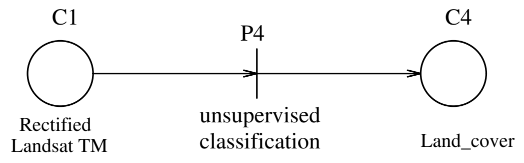
Figure 3: Derivation Management Layers

objects (most commonly just one) for the output class.

An example of a process for the derivation of LULC is illustrated in Figure 3. Process P4 derives class C4 which has four attributes: the spatial extent `C4.spatialextent`, the temporal extent `C4.timestamp`, the number of land cover classes `C4.numclass`, and raster image data `C4.data`. The extents are invariantly transferred from the input classes, while the image data is derived using the functional application of the image operators: `unsuperclassify()` and `composite()`[12]. The assertions using the rule `common()` make sure that the spatio-temporal extents of the input classes are the same or overlap. The process definition in Gaea is illustrated in Figure 4.

2.5 System-Level Semantics or the ADT Level

This layer is used to manage the abstract data type (ADT) view of the system. The mapping between the derivation semantics layer and the system layer consists of the



```
DEFINE PROCESS    P4
OUTPUT   C4
ARGUMENT  (  bands  SETOF   C1  )
TEMPLATE {
  ASSERTIONS:
          card ( bands ) = 3;        // need three bands
          common ( bands.timestamp );
          common ( bands.spatialextent );
  MAPPINGS:
          C4.spatialextent = ANYOF bands.spatialextent;
          C4.timestamp = ANYOF bands.timestamp;
          C4.numclass = 12;
          C4.data = unsuperclassify ( composite ( bands ),  12 );
}
```

Figure 4: Derivation Process for Unsupervised Classification

mapping of a process as a transformation of a set of input classes to an output class using operators that are applied to primitive classes. This is captured using a data flow network of functional operators that are applied on primitive classes, such as spatial coordinates, temporal attributes, and raster images. The mapping of Process P4 is illustrated in Figure 3.

The mapping we just described captures the structural aspect of the scientific procedure used to derived a concept (LULC in the example). Gaea provides for the actual application of such derivation procedure by dynamically interpreting each operation at run time. Whenever a user requests an LULC output, the Gaea kernel parses the query using the mapping from the top layer to the system level layer. Within that layer, operators are classified according to multiple taxonomies as illustrated in Figure 5. A user can specifically select an operator based on the function it performs; for example a classifier which performs clustering, or a multispectral image function similar to principle component analysis [35]. The specific operator can be implemented in different GIS systems such as Grass and IDRISI, or analysis software such as AVS and Khoros. Furthermore, such operators may have binaries for different architectures. Once a process is edited and specific operators selected, the user does not have to worry about the low level details of which version is used and on which platforms it is applied. The Gaea kernel through its meta-data manager and QAP takes care of this task.

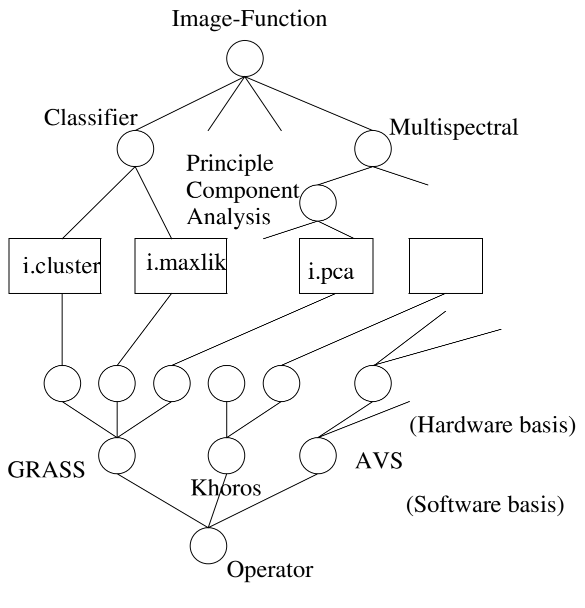


Figure 5: Operator taxonomies

We are currently designing and implementing the low level operator taxonomies in Gaea; GRASS operators are now accessible from within Gaea. To be able to use the taxonomies for distributed data management and analysis, the following types of meta-data must be included:

- Input and output data object types.
- Programming language used, necessary to determine the appropriate run-time environment.
- Experimental/computational. Is the input to the operator intended to be observed data (experimental operator) or a process model (computational operator).
- Hardware platform.
- Software environment needed. Library or other executable packages needed for proper execution.
- Invocation mode. Flags, switches, etc. must be supplied.
- Type of parallelism. SIMD, SISD, MIMD.
- Timing information. Allow query optimization based on expected time to complete an operation.

3 GaeaPN: A Model for Managing Data Derivation Semantics

The core of the Gaea system is the metadata manager. We focus here on the presentation of a model for managing the semantics of data derivations. Specifically, the petri net model we describe is implemented has been the Gaea system prototype for earth science applications. We start by motivating the use of PNs then present GaeaPN, our network model used in the Gaea system. We then discuss the closure of the model with respect to "metadata space" and the database states, followed by semantics of queries and updates. Finally we provide a brief discussion of our implementation and point out how the model can be applied in other scientific domains.

3.1 Petri Nets

Petri nets (PN) have been used extensively for representing and studying concurrent systems [**?**, **?**, **?**, **?**]. We have proposed to represent data derivation processes with PNs
[Hachem93 *et al.*]. The advantages of using PNs to model the derivation process in Gaea are [1, 9, 23, 33]:

- The graphical representation of Petri nets is not only easy to understand but has a well-defined semantics which, in an unambiguous way, defines the behavior of the system.

- PN have proven to be very useful to describe pieces of intended system behavior where process synchronization is of utmost importance and the behavior of the system needs to be analyzed.

- PN can be used to represent systems in a top-down fashion at various levels of abstraction, i.e. they can be used to model a system hierarchically.

- PN are uninterpreted models. Hence they can be used in many different environments by using appropriate interpretations.

Informally, a Petri net is an abstract model of flow of information and control of actions in a system. A Petri net structure consists of places, transitions and input and output functions [33].

Definition 1 *A Petri net is defined as the four-tuple $C = (P,\ T,\ I,\ O)$ where*
$P = \{p_1, p_2,\ \ldots\ p_m\}$ *is a set of places.*
$T = \{t_1, t_2,\ \ldots\ t_n\}$ *is a set of transitions.*
The relationship between the places and the transitions is defined using the input function I, and the output function O.
I defines for each transition t_j, the set of input places for the transition $I(t_j)$.
O defines for each transition t_j, the set of output places for the transition $O(t_j)$ [33].

The two components of a Petri net are places, represented using circles and transitions, represented using vertical lines (Figure 6). Arrows interconnect places and transitions. Tokens (black tiny circles) move from place to place according to a specific rule.

Definition 2 *A marking μ of a Petri net is an assignment of tokens to the places in that net. A Petri net $C = (P,\ T,\ I,\ O)$ with a marking μ becomes the marked Petri net, $M = (P,\ T,\ I,\ O,\ \mu)$.*

Definition 3 *The firing rules for a Petri net are as follows:*

- *A transition is said to be* **enabled** *when all of its input places have a token in them.*

- *Only one of the enabled transitions can fire at a time.*

- *A transition fires by removing the enabling tokens from their input places and generating new tokens which are deposited in the output places of the transition.*

- *The number of tokens in each place always remains nonnegative when a transition is fired* [33].

Tokens are moved by firing of the transitions. In Figure 6, for example, transition P1 is enabled since its input places A and B have at least one token in them. Transitions P2 and P3 are not enabled as their respective input places do not have any tokens in them. Transition P1 fires by removing one token from each of A and B and then deposits one token in C. The effect of firing a transition can be seen in Figure 7.

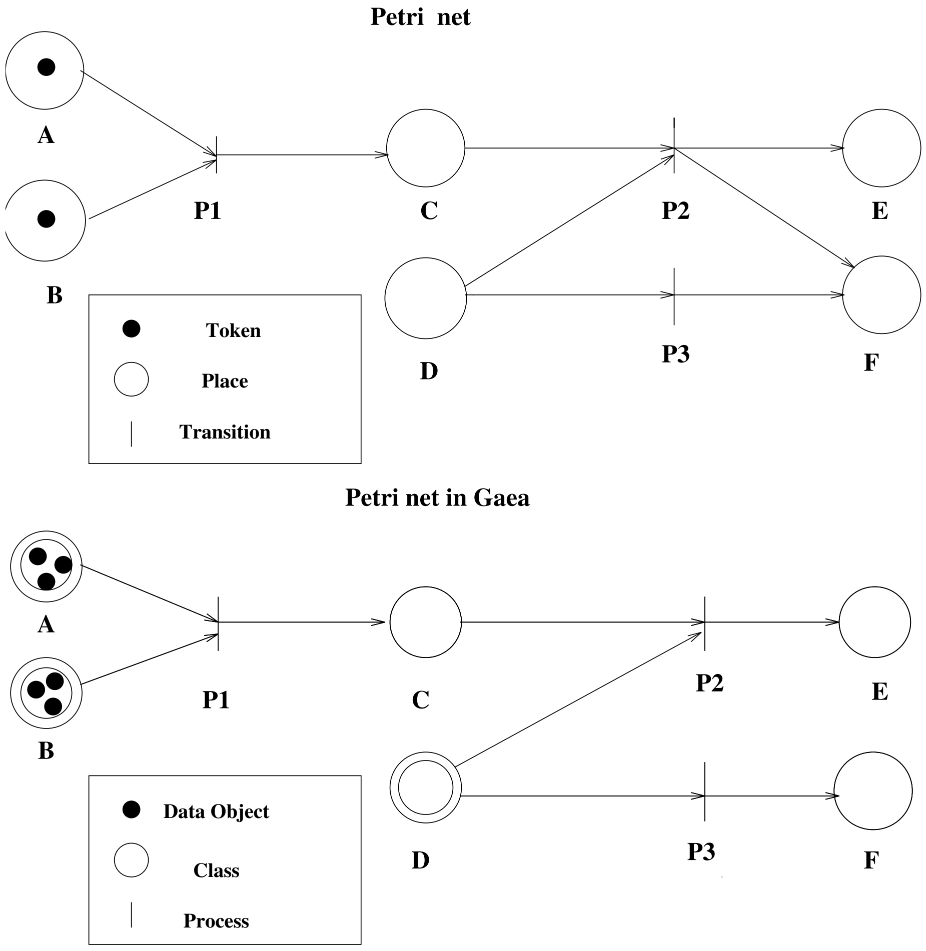


Figure 6: Petri Net and Gaea Petri Net

Many different interpretations of PN have been applied in various areas. Some of these areas are computer hardware, distributed databases, operating systems, compilers [?, 23], logic, and in hypertext systems [?]. Extensions to Petri nets have been proposed in their capacity as modeling tools and decision making tools. For example, a finite state system is a Petri net which has very high decision making capability but the modeling capabilities are limited. The latter is due to the fact that the reachable states in a finite state system is finite. Moreover cooperation of parallel sequences cannot be described using state machines. Turing machines on the other hand have very good modeling capabilities, but decision making power is limited since most problems are undecidable in case of Turing machines. In fact PN evolved to overcome the limitations of finite state machines [33]. One of the extensions was called *Generalized Petri nets* where multiple arcs between places and transitions were allowed signifying the number of input tokens used and output tokens generated. PN cannot model ordering of events, hence the PN were extended to include *inhibitor arcs* [?]. Another extension to PN is a *marked graph* where each place has exactly one input transition and one output transition. *Free choice nets* in which each arc from a place is either the unique output of the place, or the unique input to a transition is another subclass of Petri nets [33].

High-level Petri nets like Colored Petri nets (CP-net) have provided methods for organizing concepts in a hierarchy. Moreover CP-net provides a formalism for relating and analyzing the individual Petri nets. CP-net consists of three different parts: *net structure, the declarations and the net inscriptions.* The *net structure* is a directed graph with two kinds of nodes, places and transitions, interconnected by arcs. The latter is prevalent in low-level Petri nets. The *declarations* describe the different types of data and the variables being used in the the description of any process using a CP-net. The *net inscriptions* consist of names for the places, transitions and arcs, the types of data (color sets) and initialization expressions attached to a place, the *guards* attached to a transition and the *arc expressions* attached to an arc. Some of these features were incorporated into the variant of Petri nets proposed in Gaea, but use of CP-nets to model derivation semantics was not found to be significantly beneficial.

The different extensions to Petri nets were found to be inadequate for the modeling and decision support capabilities desired by the Gaea System. An interpretation and extension of Petri nets for the derivation semantics layer of the Gaea System is given below.

3.2 Interpretation of Petri Nets in Gaea: GaeaPN

Petri nets are uninterpreted models as mentioned earlier and they can exhibit complex behavior. A meaning or interpretation can be assigned to the different entities in the net namely, the places, transitions and tokens. Thus Petri nets can be used in Gaea to model the derivation process with appropriate interpretation.

Definition 4 *The Gaea Petri net (GaeaPN) is defined as the six-tuple* $G = (C, P, I, O, A, m_0)$ *where*
$C = \{c_1, c_2, \ldots c_m\}$ *is a set of classes (places) where a class encapsulates the structure of the different types of data.*
$P = \{p_1, p_2, \ldots p_n\}$ *is a set of processes (transitions) where a process consists of the*

procedures applied to the data in the classes.
The relationship between the classes and the processes is defined using the input function I, and the output function O.
I defines for each process p_j, the set of input classes for the process I(p_j).
O defines for each process p_j, the output class for the process O(p_j). Each process has an unique output class.
A is an assertion function. It is defined from process P into expressions such that A(p_j) is true.
m_0 is the initial marking of GaeaPN. It is the set of instances of data-objects in the base classes. A class c_i is a base class if it is not an output class to a process. A marking of a GaeaPN is an assignment of tokens to the classes in that net.

A **token** in GaeaPN is the **instance of a class** (data object)[4]. A class can have more than one token as illustrated for classes A and B of the GaeaPN of Figure 6. Also, from Figure 6 one can observe that in GaeaPN a process has only one output class hence process P2 does not have an arrow to class F in the GaeaPN.

3.3 Execution Rules for Marked GaeaPN

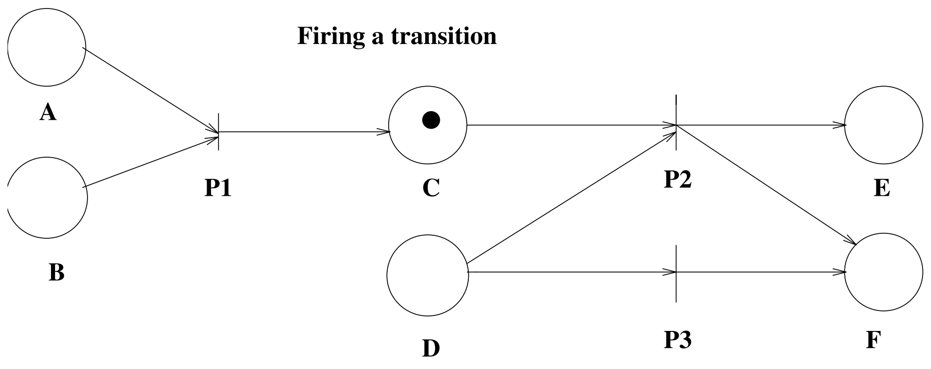
In Gaea tokens are not created and consumed hence they are always present in the system as can be seen in Figure 7. There is only one token in a class in case of a PN but many tokens in case of a GaeaPN. The state of a Petri net is defined by its marking.

Definition 5 *A marking in a GaeaPN is modified according to the following firing rules:*

- *A process p is said to be instantiatable (enabled), if there exists a set of tokens in each input class c_i of process p such that A(p) (the assertions for process p) is true (are satisfied).*
- *Each of the instantiatable processes can fire at any time.*
- *Instantiating (firing) a process p, does not remove any tokens from each input class c_i of process p, but adds one token to the output class c of p.*

Therefore, the equivalent to firing of a transition in a PN is instantiating a process in a GaeaPN. The only difference between the two being, that the latter does not remove a data object from its input class when it generates a new data object in the output class.

[4]Although it is not discussed here, the instance of a class (data object) in Gaea is a spatio-temporal object i.e. it has an intrinsic spatial and temporal attribute [?].



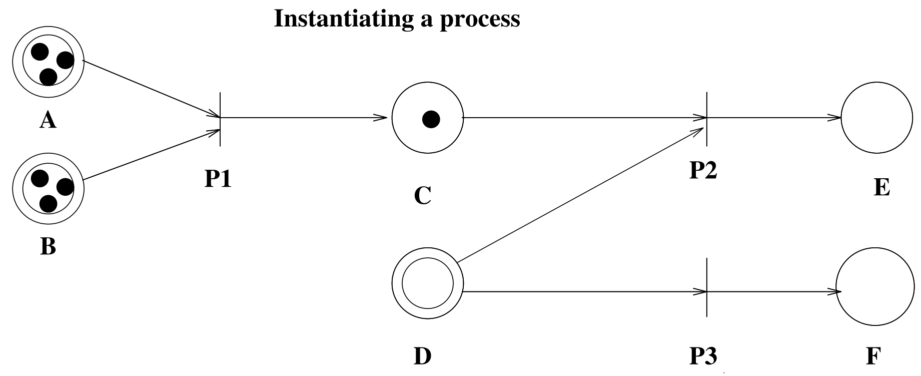


Figure 7: Results of Firings in a Basic Petri Net and a Gaea Petri Net

Definition 6 *Given an initial marking, a marking in a GaeaPN is said to be a* **legal marking** *if it is a result of successive and/or simultaneous instantiations of processes from the initial marking.*

Definition 7 *Given an initial marking the union of all legal markings is called a* **final marking**.

Every legal marking includes the initial marking m_0 and the final marking includes every other legal marking. Since the number of tokens in a class is bounded (finite) over the set of all markings, there is a finite number of markings for a GaeaPN. A final marking is achievable as the cardinality of the initial marking is finite and only an enumerable number of tokens can be generated from the initial marking.

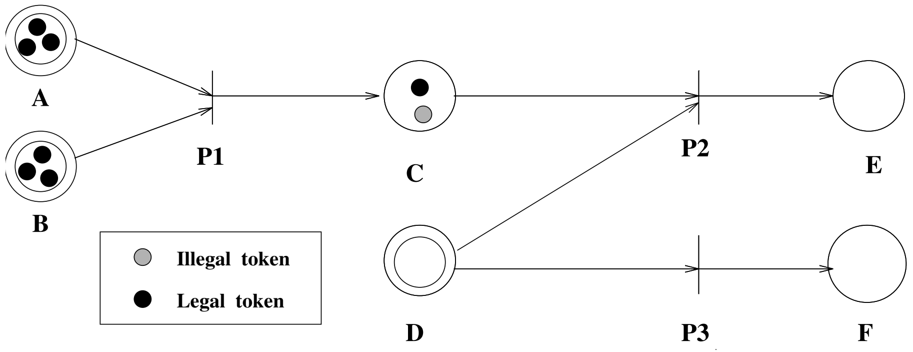


Figure 8: Illegal Marking in a GaeaPN

Illustration: Consider the GaeaPN in Figure 6. The initial marking, m_0 consists of the set of data objects in classes A and B i.e., a set of three data objects from each of the classes making it a total of a set of six data objects. From the legal marking m_0 if P1 is instantiated, generating an object in class C the resultant marking m_1, is a legal marking as illustrated in Figure 7. Assume a new data object is added to class C as shown in Figure 8 i.e., without instantiation of process P1. Then the marking $m_1^{'}$ (which includes m_0) is NOT considered to be a legal marking. Consider the GaeaPN in Figure 9 the initial marking, m_0 is the set of data objects in classes A, B and D. The marking m_1 is the result of instantiating process P1 from the initial marking m_0, generating an object in class C. The marking m_2 is the result of instantiating process P2 from marking m_1 generating an object in class E. The marking m_3 is the result of instantiating process P3 from marking m_0 generating an object in class F. Assuming

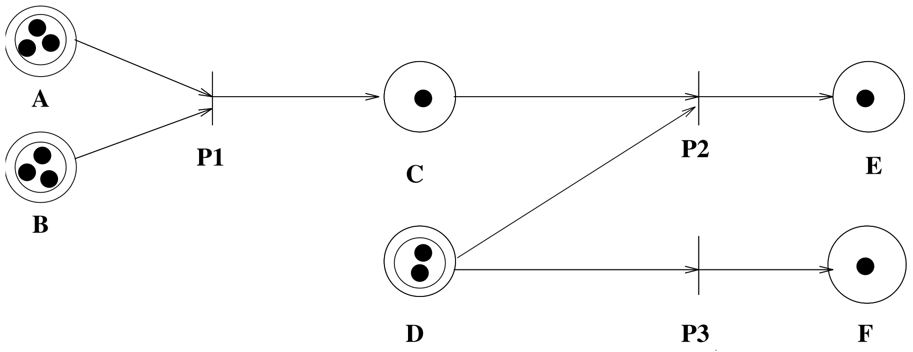


Figure 9: GaeaPN with a Final Marking

for this example that it is not possible for other instantiations to take place. Thus, the final marking, m_f for the GaeaPN of Figure 9 is $m_0 \cup m_1 \cup m_2 \cup m_3$.

Proposition 1 *The set of legal markings forms a partially ordered set (poset), ordered by the operation of set inclusion. The least upper bound (lub) of the set is the final marking and the the greatest lower bound (glb) is the initial marking. Hence it follows that the set of legal markings forms a* **lattice***.*

Illustration: The initial marking m_0 is the glb as it is contained in every marking and the final marking m_f is the lub. The set of legal markings $\{ m_0, m_1, m_2, m_3 \}$ forms a lattice. m_0 is contained in m_1 and m_1 is contained in m_2. Similarly m_0 is contained in m_3. However m_2 and m_3 are not comparable.

Petri nets in Gaea have another salient feature, called *assertions* which is similar to *guards* in CP-nets [23]. In CP-nets the *guard* of a transition is a boolean expression which must be fulfilled before the transition can fire. In GaeaPN, the assertions are a part of the process definition. Assertions are used to provide the following features:

- Guarantee the integrity of data.
- When an assertion is specified, the system tests it for validity. For a process to be instantiated the input data needs to satisfy the assertions. Hence the assertions are validated as part of the process of instantiation.
- the capture and expression of relationship between classes as pertaining to a specific derivation process.

3.4 Observations

Particularly, very useful in Gaea is that Petri nets can be used to represent systems in a top-down fashion at various levels of abstraction. The latter is most useful in Gaea when the operators that comprise a process need to be depicted as illustrated by process P7 in Figure ?? (Refer Section ??).

The problems of *conflict* which arise in PN do not arise in GaeaPN as tokens are not removed from the input places [?, 33]. Since firing of one of the transitions in the conflict does not disable the other i.e. more than one process can be instantiated with the same data object. Nevertheless the problems of *deadlock* may arise in Gaea. A process is said to be *dead* in a marking if there is no sequence of process instantiations that can instantiate it. In other words a dead process is one which is not only uninstantiatable but a process which cannot become instantiatable. The latter may arise when base classes do not have any data, hence processes directly or indirectly dependent on that base class will be dead. For example, in Figure 6 process P3 is said to be dead as there are no tokens (data-objects) in its input class. Process P2 is also dead as it needs input data from classes C and D. D being a base class and as it does not have any data causes P2 to be dead, even though class C also does not have any data. The latter is due to the fact that the data in class C can be generated eventually by instantiating process P1 but data in base classes cannot be generated. Furthermore the state of *nondeterminism* does not occur in the GaeaPN. Since when more than one transition is enabled in Gaea all of them can fire based on some order.

Logic and GaeaPN: One can draw a correspondence between logic and GaeaPN. The initial marking corresponds to the *extensional* set of facts of a database (EDB) and the *intensional* set of rules of the database (IDB) correlates with the set of processes [?]. The final marking in GaeaPN corresponds to the closed world assumption (CWA).

3.5 Analysis of Petri Nets

Different properties of PN that have been investigated as analysis tools are *boundedness, conservation of tokens, safe nets, liveness of transitions* [33]. The property of *conservation of tokens* is not important in GaeaPN as tokens are not created and consumed. Similarly the property of a *safe net and boundedness* is irrelevant since there is no bound on the number of tokens in any class (place) of the net. The implication of *liveness* can be different for different systems modeled using PN. The concept of *liveness* is reducible to the *reachability problem* in PN and can be used to analyze a PN [33]. Reachability is defined in GaeaPN with three different interpretations of the PN, namely the graph based, the class based and the object based interpretation.

3.5.1 Reachability

In Petri nets a *reachability set* is defined as the set of all states into which the Petri net can enter by any possible execution. The *reachability problem* is as follows: Given a marked Petri net (with marking M) and a marking M', is M' reachable from M [33]?

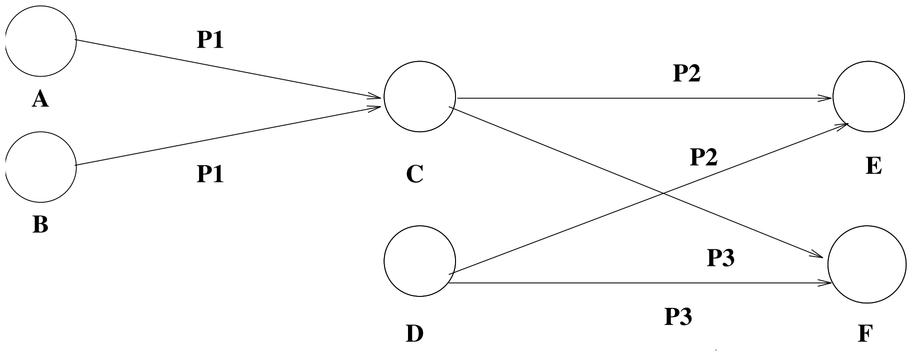


Figure 10: Graph Representation of the Petri Net in Figure 6

A Petri net is also defined as a bipartite directed graph as illustrated in Figure 10 [?]. The correspondence between PN and graphs is so similar that most often they are considered as different representations for the same concept [33]. In graph terminology a node n_1 is reachable from a node n_2 if n_1 equals n_2, or there is a path from n_2 to n_1 [?]. Therefore a graph based definition of reachability is as follows:

Definition 8 Class reachability (graph based) *A class* **b** *is said to be reachable if it is a base class or a class derived by a process* **p** *such that the set of input classes* $\{a_i\}$ *of process p are reachable. The set of input classes* $\{a_i\}$ *are said to be reachable if each of the members of the class are reachable. The* **reachability set** *of a class is the pseudo-transitive closure of the set of reachable classes from a given set of classes.*

The GaeaPN can be used to determine if a class is reachable or an instance of a class (data object) is reachable. Hence the definition of reachability of an object is different from the reachability of a class. In order to define class and object reachability one needs to define the *reachability path* of a class. The definitions of class reachability and object reachability are based on the definition of a **reachability path**.

Definition 9 *The* **Reachability path** *of a class a_i is the set of processes $\{p_i\}$, that should be instantiated from a given set of classes $\{a_j\}$ to reach the class a_i.*

Definition 10 Class reachability (PN based): *Given a set of input classes $\{a_i\}$ with data objects (instances), a class b is said to be reachable if there exists a subset of objects from the set of input classes $\{a_i\}$ that instantiate all processes in the reachability path of class b.*

Definition 11 Object reachability: *Given a set of input classes $\{a_i\}$ with a set of data objects (instances or tokens), an object b in class* **c** *is said to be reachable if the given set of data objects from the set of input classes $\{a_i\}$ instantiates all processes in the reachability path of class c and generates the object b.*

Illustration: Consider the GaeaPN of Figure 6. The Petri net in Gaea has tokens in classes A and B. These tokens form the initial marking. Based on the **graph based** definition of class reachability, class E is said to be reachable from class A. The **reachability path** of class E is the set of processes {P1, P2}. Based on the **class reachability** definition class E is not reachable, since there are no tokens in class D, hence process P2 cannot be instantiated to obtain any new data-object in class E. Again based on the definition for **object reachability**, class E is not reachable from class A. However class C is said to be reachable from both the graph based view and by the definition for class reachability. Although a new object is generated in class C as a result of instantiation of process P1, the latter may not be the object desired by the user. Therefore, from the **object reachability** point of view an object in class C is said to be reachable if the new object generated is the one desired by the user.

In addition to *class reachability, object reachability, and graph based reachability,* **reverse reachability** can be defined for a class and an object. Intuitively, reverse reachability is the ability to determine the source data provided the target data exists. As opposed to the earlier definitions of reachability, new data is not generated as a result of **reverse reachability**. It is only the ability to determine the source from which the target data was generated. Therefore the definitions for **reverse reachability** are as follows:

Definition 12 Reverse class reachability: *Given a non base-class the ability to recursively determine the set of input classes, that had the set of data objects to generate at least one new data object in the given non base-class.*

Definition 13 Reverse object reachability: *An object b is said to be reverse reachable if the set of objects in the initial marking that were used to generate the object b can be determined. Object reachability is the ability to determine the set of data objects that were used to generate the object b.*

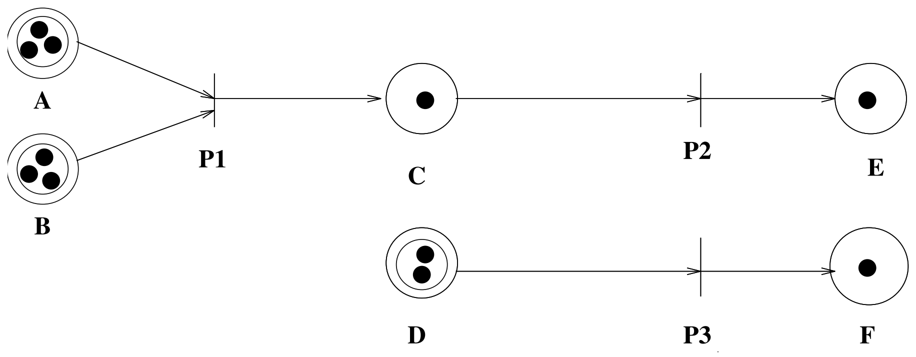


Figure 11: Reverse Class and Object Reachability

Illustration: Consider the Petri net in Gaea as shown in Figure 11. Classes E and F have a token (data object), assume it is the final marking. The initial marking for those tokens in classes E and F is a token in class A, class B and class D. Therefore, the classes E and F are said to be reverse reachable. Similarly if the object (token) b in class E is the final marking then the initial marking for the marking in class E is in class A and class B. However one needs to note that reverse reachability gets the initial marking. If the user knows the structure of the network then he/she can determine from which class in the network the marking needs to be obtained.

In addition to the property of finding the reachability set of a Petri net given an initial marking, in Gaea one would be able to find the initial marking for a desired legal marking. Moreover it would generate the data objects in the legal marking. The latter is due to the property of *reversibility*.

3.5.2 Reversibility

In Gaea a process is said to be reversible, if for a given object in the output class of process p, the corresponding object in the input class of process p can be found from the mappings defined as part of the process definition. The syntax and semantics of the Gaea language construct *define process* is given in Section **??**. An example of a process that is reversible is as follows:

Example:
```
    Define process
    output o1
    arguments ( a of in1)
```

```
template
mappings:
o1.timestamp = a.timestamp;
o1.spatialextent = a.spatialextent;
```

Assume the granularity of time is the same in the input and output class. The functions defined by the two mappings specified in the process are bijective mappings. Therefore, if the user desires information for "15 Nov 1992" and for the town of "Worcester", the information can be retrieved from the input class and assigned to the output class. Hence it can be said that the process p is reversible. An example of a process that is not reversible is as follows:

Example:

```
Define process p2
output o2
arguments ( a of in2)
template
mappings:
o2.timestamp = a.timestamp;
o2.spatialextent = a.spatialextent;
o2.data = gIMaxlik(gICluster(gMkGroup(a.filename), 12));
```

In the above process definition *gIMaxlik, gICluster, gMkGroup* are operators in Gaea borrowed from GRASS which perform the functions of *Maximum Likelihood, Clustering*, and *making a group* respectively. In the above process if an output object is specified as "15 Nov 1992" and for the town of "Worcester", the input object cannot be found as there is no indication as to which input object needs to be used. Moreover even if objects can be found to have the timestamp as "15 Nov 1992" and for the town of "Worcester", they may not all be usable. On the other hand, if objects can be found to have the timestamp "15 Nov 1992" and for the town of "Worcester" they may generate more than one object. Thus it is observed that precisely one desired object cannot be generated. Therefore the process is not reversible.

Definition 14 Reversibility *is the process of generating the desired set of data in class b by instantiating processes in the reachability path of class b provided*

- *all processes in the GaeaPN are reversible and*
- *a set of classes with the set of data objects that can be used to obtain the desired set of data in class b can be found.*

Difference between reachability and reverse reachability:

- Reachability generates data if it does not exist.
- Reverse reachability does not generate any data, it only helps you find the source of the target data. In order to do reverse reachability the target data should have been generated earlier.

Difference between reversibility and reverse reachability:

- For reversibility the target data need not exist. Reversibility generates the target data when the target data does not exist after determining which is the source data that can be used to generate the target data.
- For reverse reachability the target data has to exist.

Uses of Reachability and Reversibility: It automates the process of data derivation in Gaea, thus making the process of derived data retrieval transparent to the user.

4 Relationships to Other Work

[extend with new stuff from Nina's and Additional on Moose and others from Ke...]

In this Section, we review other proposed mechanisms that relate to our work, and make some comparisons.

4.1 Related Work in Conceptual Modeling

Markowitz [25] uses the extended E-R approach to model both the functional and structural components of an information system. The basic idea is to represent a process as a relationship and apply existential constraints to express the partial order implied in a process. However, we do not believe that the E-R approach is sufficient to represent derivation relationships among data classes for the following reasons:

1. An E-R diagram is basically a network structure, while the derivation relationship actually defines a hierarchical structure among data classes, which is not obvious in an E-R diagram representation.
2. Derivation relationships are different from other kinds of relationships in an E-R model. The input data classes and output class of a derivation relationship cannot be directly mapped into the E-R model. Furthermore, the constraints involved in a derivation relationship cannot be expressed in the E-R model.

3. Compared with the E-R diagram, the PN we propose to use expresses more semantics for a derivation relationship. It shows not only the input and output classes but also the constraints on a derivation procedure. Those constraints are in the form of guard rules that need to be satisfied for a derivation to be applicable. We have briefly shown how PNs can also be used to generate derived data automatically. Furthermore, PNs can be used to capture the control flow of the scientific computation on hand.

4.2 Derivation Management vs. Functional Modeling

One may find similarities between our work and functional modeling in the system analysis stage of business database applications. However they are different in their purpose and the methods used.

Usually an information system is described by two components: structure and function. In the structural component, entities and their relationships are identified. This is also called a static view of the database and forms the basis for schema definition. The dynamic view (behavior) of the database is described in the functional component, which forms the basis for application programs.

One popular method for functional modeling is Data Flow Analysis [26]. In data flow analysis, an information system is considered as a process that maps input data to output data, and can be represented as a data flow diagram. Then the transformation process is further decomposed into subprocesses until each is basic enough to be implemented with a piece of simple program.

Although functional analysis is also concerned with a process, the purpose is different from that of derivation management in scientific databases. A process in functional analysis is used to develop application programs, while a process in our work is used to define derivation relationships among data classes. In addition, a task, the instantiation of a process, is of no interest in functional analysis, while it plays an important role in data derivation management. It is an individual task that defines the derivation relationship among a set of data objects.

In summary, functional analysis is concerned with how to transform input data to output data, i.e., how to accomplish the task; while data derivation management is concerned with how the data were and will be generated, i.e., how the task was and will be accomplished.

4.3 Related Research in Scientific Databases

Experiment management is also the goal in [8]. The problem is to model experiments in computational chemistry, and the approach followed is based on the object-oriented paradigm. Cushing *et. al.* derived a model that captures the interrelationships between the data, its source, methods and instruments used, and other information relevant to the generation of the data. They provide a mechanism for managing the definition, preparation, monitoring and interpretation of computational experiments. We address the same problem, but identify differences between experiment management and data derivation management. By using different formalisms to model them, we have introduced more semantics into our system.

Semantic networks are an appropriate tool to capture the relationships among a set of data objects. This formalism has been used in the USD system [36]. Although their intention was to make use of the flexibility of semantic networks to represent unstructured data, it can also be adequately used to model an experiment. The problem with semantic networks is that they might become too complex with a large database system. In addition, data derivation relationships are not explicitly represented in the network.

[add some of Nina's stuff.]

5 Conclusions and Future Work

[bla bla bla]

References

[1] T. Agerwala, "Putting Petri Nets to Work," IEEE Computer Magazine, pp. 85-94, Dec. 1979.

[2] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," CACM, Vol. 26, No. 11, pp. 832-843, Nov. 1983.

[3] M. Atkinson, F. Bancilhon, D. DeWitt, D. Maier, and S. Zdonik, "The Object-Oriented Database System Manifesto," Proc. Int. Conf. on Deductive and Object-Oriented Databases, pp. 40–57, 1989.

[4] J.-L. Baer, "Modeling Architectural Features With Petri-Nets," Lecture Notes in Computer Science, Springer-Verlag, No. 255, pp. 258-277, 1986.

[5] A. Beller, "Spatial/Temporal Events in GIS," GIS/LIS 91, V57, N4, pp. 407–411, 1991.

[6] G. Bender, "Reference Handbook on the Deserts of North America," Greenwood Press, p. 594, 1982.

[7] P.P. Chen, "The Entity-Relationship Model: Towards a Unified View of Data," ACM Trans. on Database Systems, Vol. 1, No. 1, pp. 9–36, 1976.

[8] J.B. Cushing, et. al. "Object-Oriented Database Support for Computational Chemistry," Proc. SSDM'92, pp. 58–76, 1992.

[9] A.M. Davis, *A comparison of techniques for the specification of external system behavior*, Communications of the ACM, Sept. 1988, Vol.31, No.9, pp.1098-1115.

[10] J. Dozier, "Access to data in NASA's Earth Observing System," Keynote Address, Proc. ACM SIGMOD Intern. Conf. on Management of Data, San Diego, 1992.

[11] J.R. Eastman, "Time Series Map Analysis Using Standard Principle Components," ASPRS/ACSM/RT 92, pp. 195–205, 1992.

[12] J.R. Eastman and J.McKendry, "Change and Time Series Analysis in GIS," UNITAR, 1991.

[13] J.R. Eastman, "IDRISI – A Grid-Based Geographic Analysis System (User's Manual)," Clark University, Worcester, MA, Nov. 1990.

[14] M. Egenhofer and J. Herring, "A Mathematical Framework for the Definition of Topological Relationships," Proc. 4th Intl. Symp. on Spatial Data Handling, pp. 803-813, Zurich, Switzerland, 1990.

[15] A.U. Frank, "Properties of Geometric Data: Requirements for Spatial Methods," In Advances in Spatial Databases, 2nd Symp., SSD'91, Spring-Verlag, Zurich, Switzerland, Aug. 1991.

[16] J.C. French, A.K. Jones, and J.L. Pfaltz, "Summary of the Final Report of the NSF Workshop on Scientific Database Management," SIGMOD Rec. **19**-4, pp. 32–40, 1990.

[17] H.J. Genrich "Predicate/Transition Nets," Lecture Notes in Computer Science **254**, pp. 207–247 , Springer-Verlag, 1986.

[18] O. Guenther and A. Buchmann, "Research Issues in Spatial Databases," SIGMOD Rec. **19**-4, pp. 61–68, 1990.

[19] N.I. Hachem, "Petri-Net Driven Knowledge Base System for Automated Microcode Generation in VLSI," in Proc. of the IASTED Int. Symp. MODELING and SIMULATION, Calgary, Canada, July 1991.

[Hachem93a *et al.*] N. I. Hachem, K. Qiu, M. Gennert, M. Ward, *Managing Derived Data in the Gaea Scientific DBMS*, Proceedings of the VLDB Conference 1993, pp.1-13.

[Hachem93 *et al.*] N. I. Hachem, M. Gennert, M. Ward, *Distributed Database Management for Scientific Data Analysis*, Proc. Int. Wkshp. on Global GIS, Int. Soc. Photogrammetry and Remote Sensing WG IV/6, Tokyo, Japan, pp. 85–93, Aug. 1993.

[20] N.I. Hachem, M.A. Gennert, and M.O. Ward, "A DBMS Architecture for Global Change Research," Proc. ISY Conf. on Earth and Space Science, Pasadena, CA, pp. 186–194, Feb. 1992.

[21] M.A. Holliday and M.K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," IEEE Trans. on Soft. Eng., Vol. 13, Dec. 1987.

[22] R. Hull and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues," ACM Computing Surveys, Vol. 19, No. 3, pp. 201–260, 1987.

[23] K. Jensen, *Coloured Petri Nets: A High Level Language for System Design and Analysis*, EATCS Monographs on Theoretical Computer Science, 1992, pp.342-416.

[24] G. Kappel and M. Schrefl, "Object/Behavior Diagrams," Proc. of Intl. Conf. on Data Engineering, pp. 530–539, 1991.

[25] V.M. Markowitz, "Representing Processes in the Extended Entity-Relationship Model," Proc. Intl. Conf. Data Engineering, pp. 103–110, 1990.

[26] J. Martin and C. McClure, "Diagraming Techniques for Analysis and Programming," Prentice-Hall, New Jersey, 1985.

[27] L.E. McKenzie, Jr. and R. Snodgrass, "Evaluation of Relational Algebras Incorporating the Time Dimension in Databases," ACM Computing Surveys, Vol. 23, No. 4, pp. 501-543, Dec. 1991.

[28] T. Muck and G. Vinek, "Modeling Dynamic Constraints Using Augmented Place Transition Nets," Information Systems, **14**-4, pp. 327–340, 1989.

[29] F.J. Monkhouse and J. Small, "A Dictionary of the Natural Environment," Halsted Press, p. 320, 1978.

[30] B.C. Ooi., "Efficient Query processing in Geographic Information Systems," In Lecture Note in Computer Science, Spring-Verlag, 1990.

[31] J. Peckham and F. Maryanski, "Semantic Data Models," ACM Computing Surveys, Vol. 20, No. 3, pp. 153–189.

[32] J.L. Peterson, "Petri Net Theory and the Modeling of Systems," Prentice Hall, 1981.

[33] James L. Peterson, *Petri Nets*, Computing Surveys, Vol.9, No.3, September 1977, pp.223-252.

[34] C.V Ramamoorthy and G.S. Ho, "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," IEEE Trans. on Soft. Eng., Vol. 6, Sep. 1980.

[35] J.A. Richards, "Multispectral Transformations of Image Data," Chapter 6 in Remote Sensing Digital Image Analysis, Springer-Verlag, pp 127–145, 1986.

[36] R. R. Johnson, M. Goldner, M. Lee, K. McKay, R. Shectman, and J. Woodruff, "USD - A Database Management System for Scientific Research," Video presentation at the ACM SIGMOD Int. Conf. on Management of Data, San Diego, 1992.

[37] K. Qiu, N.I. Hachem, M.O. Ward, and M.A. Gennert, "Providing Temporal Support in Data Base Management Systems for Global Change Research," Proc. SSDM '92, Switzerland, 1992.

[38] H. Sakai, "A Method for Entity-Relationship Behavior Modeling," Entity-Relationship Approach to Software Engineering, North-Holland, pp. 111–129, 1983.

[39] A. Segev and A. Shoshani, “Logical Modeling of Temporal Data, ” Proc. ACM SIGMOD Conf., pp. 454–466, 1987.

[40] M. Shapiro, et. al., “GRASS 4.0 Programmer’s Manual (Draft),” U.S. Army Construction Engineering Research Laboratory, April 1992.

[41] R. Snodgrass, I. Ahn, “Temporal Databases,” IEEE Computer, Vol. 19, No. 9, pp. 35–42, Sept. 1986.

[42] M. Stonebraker, L.A. Rowe, and M. Hirohima, “The Implementation of POSTGRES,” IEEE Trans. Knowledge and Data Eng. **2**-1, pp. 125–142, 1990.

[43] L. Vincent and P. Soille, “Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations,” IEEE Trans. on PAMI, Vol. 13, No. 6, pp. 583–598, 1991.

[44] Y. Zhang, M.O. Ward, N.I. Hachem, and M.A. Gennert, “A Visual Programming Environment for Supporting Scientific Data Analysis,” Proc. of the Int. Workshop on Visual Prog. Languages, August 1993. (extended version as WPI-CS-TR 93-01, March 1993)

[45] Y. Zhou, M. Gennert, N. Hachem, and M. Ward, “Requirements of a Database Management System for Global Change Studies,” ASPRS/ACSM/RT 92, pp. 186–194, 1992.