18TH INTERNATIONAL COMMAND AND CONTROL RESEARCH AND TECHNOLOGY SYMPOSIUM

C2 IN UNDERDEVELOPED, DEGRADED AND DENIED OPERATIONAL ENVIRONMENTS

MULTI-ENTITY BAYESIAN NETWORKS LEARNING IN PREDICTIVE SITUATION AWARENESS

Topic 3: Data, Information and Knowledge

Cheol Young Park* [STUDENT]
Kathryn Blackmond Laskey
Paulo Costa
Shou Matsumoto

The Sensor Fusion Lab & Center of Excellence in C4I
The Volgenau School of Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030-4444
(703) 332-9921

cparkf@masonlive.gmu.edu, [klaskey, pcosta]@gmu.edu, smatsum2@masonlive.gmu.edu

Point of Contact: Cheol Young Park
cparkf@masonlive.gmu.edu and/or (703) 332-9921

ABSTRACT

Over the past two decades, machine learning has led to substantial changes in Data Fusion Systems throughout the world. One of the most important application areas for data fusion is situation awareness to support command and control. Situation Awareness is perception of elements in the environment, comprehension of the current situation, and projection of future status before decision making. Traditional fusion systems focus on lower levels of the JDL hierarchy, leaving higher-level fusion and situation awareness largely to unaided human judgment. This becomes untenable in today's increasingly data-rich environments, characterized by information and cognitive overload. Higher-level fusion to support situation awareness requires semantically rich representations amenable to automated processing. Ontologies are an essential tool for representing domain semantics and expressing information about entities and relationships in the domain. Probabilistic ontologies augment standard ontologies with support for uncertainty management, which is essential for higher-level fusion to support situation awareness. PROGNOS is a prototype Predictive Situation Awareness (PSAW) System for the maritime domain. The core logic for the PROGNOS probabilistic ontologies is Multi-Entity Bayesian Networks (MEBN), which combines First-Order Logic with Bayesian Networks for representing and reasoning about uncertainty in complex, knowledge-rich domains. MEBN goes beyond standard Bayesian networks to enable reasoning about an unknown number of entities interacting with each other in various types of relationships, a key requirement for PSAW. The existing probabilistic ontology for PROGNOS was constructed manually by a domain expert. However, manual MEBN modeling is labor-intensive and insufficiently agile. To address this problem, we developed a learning algorithm for MEBN-based probabilistic ontologies. This paper presents a bridge between MEBN and the Relational Model, and a parameter and structure learning algorithm for MEBN. The methods are evaluated on a case study from PROGNOS.

1 INTRODUCTION

Over the past two decades, machine learning has led to substantial changes in Data Fusion Systems throughout the world [White, 1988; Endsley, 1988; Steinberg et al., 1998; Endsley et al., 2003; Llinas et al., 2004; Linggins et al., 2008]. One of the most important application areas for data fusion is Situation Awareness (SAW) to support command and control (C2). Systems to support SAW provide information regarding the present or future situation. This information supports situation assessment (SA) and is exploited for C2 decision making.

According to the most common cited definition, SAW is composed of three processes; perception of elements in the environment, comprehension of the current situation, and projection of the future status [Endsley, 1988; Endsley et al., 2003]. Breton and Rousseau classified 26 SAW definitions and identified a set of common elements of SAW. They identified two distinct varieties, which they termed State- and Process-oriented SAW. In their definition, Process-oriented SAW focuses on the link between the situation and the cognitive processes generating SAW, while State-oriented SAW focuses on the link between the situation and an internal representation of elements present in the situation [Breton & Rousseau, 2001].

In contrast to traditional SAW, Predictive Situation Awareness (PSAW) emphasizes the ability to make predictions about aspects of a temporally evolving situation [Costa et al., 2009;

Carvalho et al., 2010]. Traditionally, decision makers are responsible for the higher-level data fusion in which they use the results of low-level fusion to estimate and predict the evolving situation. PROGNOS is a prototype system intended to address the need for higher-level data fusion [Costa et al., 2009; Carvalho et al., 2010]. PROGNOS provides higher-level fusion through state-of-the-art knowledge representation and reasoning.

The PROGNOS probabilistic ontologies employ Multi-Entity Bayesian Networks (MEBN) which combines First-Order Logic with Bayesian Network for representing and reasoning about uncertainty in complex, knowledge-rich domains [Laskey, 2008]. MEBN goes beyond standard Bayesian networks to enable reasoning about an unknown number of entities interacting with each other in various types of relationships. A PSAW system must aggregate state estimates provided by lower level information fusion (LLIF) systems to help users understand key aspects of the aggregate situation and project its likely evolution. A semantically rich representation is needed that can capture attributes of, relationships among, and processes associated with various kinds of entities. Ontologies provide common semantics for expressing information about entities and relationships in the domain. Probabilistic ontologies (PR-OWL) augment standard ontologies with support for uncertainty management [Costa, 2005]. PR-OWL 2 extends PR-OWL to provide better integration with OWL ontologies [Carvalho, 2011]. MEBN is the logical basis for the uncertainty representation in the PROGNOS Probabilistic ontologies.

1.1 MEBN for PSAW

Figure 1 shows a simplified illustrative example of a problem in PSAW. Our goal is to estimate a vehicle type (e.g., tracked and wheeled) of a target object and a degree of danger (e.g., high and low) of a specific region. Figure 1 depicts a specific situation of interest.

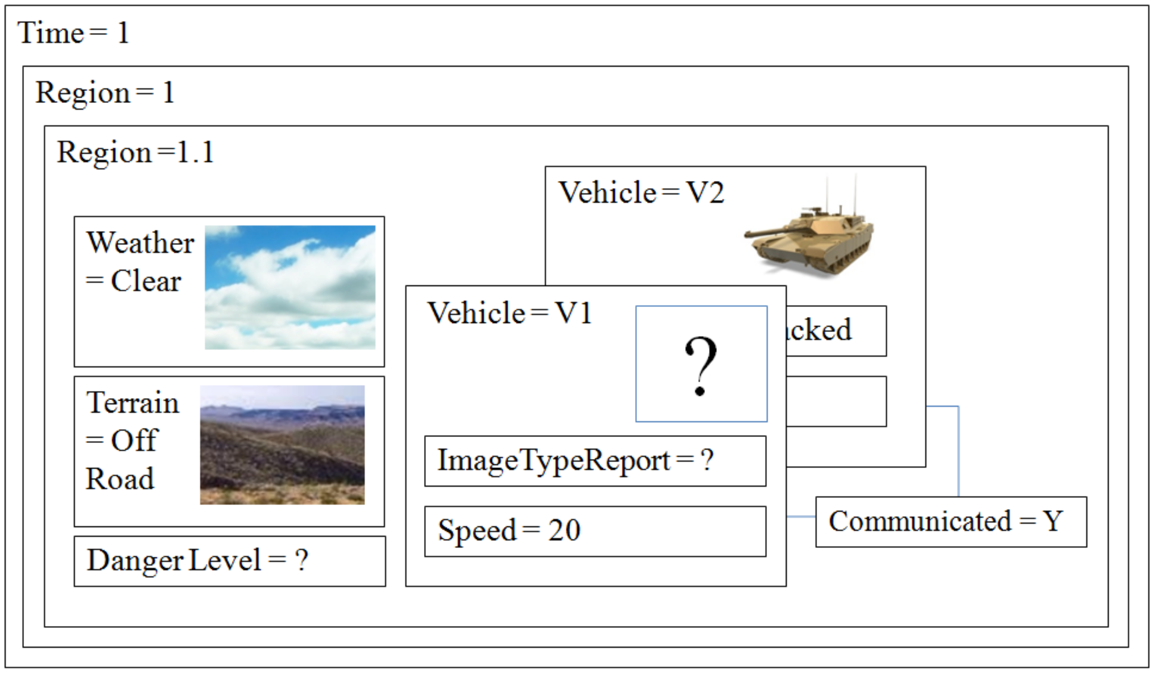


Figure 1. Vehicle Identification Context in PSAW

The rectangles in Figure 1 mean instances of entities. Figure 1 expresses two relations among entities. An inner rectangle which is shown within an outer rectangle means a part entity of an entity represented by the outer rectangle, so it means composition or aggregation. The rectangle described by "Communicated = Y" specifies an interconnected relation.

Our system has been provided with the following evidence. At Time 1, a weather sensor has reported clear weather for Region 1.1. A geographic information system has reported that Region 1.1 is off-road terrain. Two vehicle objects, V1 and V2, have been detected by an imaging system, which has reported that V2 is tracked and has failed to report a type for V1. An MTI sensor indicates that both vehicles are traveling slowly. A COMINT report indicates communications between V1 and V2. Given this evidence, we want to know the object type of both vehicles and the danger level of the Region 1.1.

We might consider using a Bayesian network (BN) [Pearl, 1988] to fuse these reports from multiple sources and answer the queries of interest. Figure 2 shows a Bayesian network we might use for this problem.

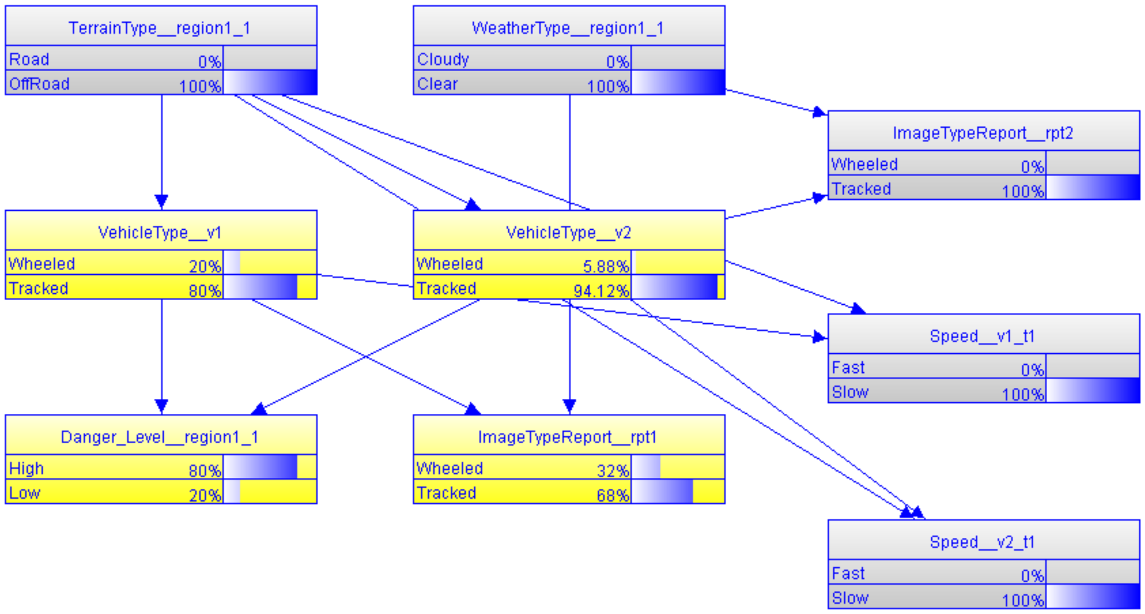


Figure 2. Bayesian Network for Vehicle Identification Context

Each box in the figure depicts a random variable (RV), or node. A label at the top of the box gives a name for the RV, the labels inside the boxes indicate its possible states, and the numbers indicate the probability of the state given our current evidence. For example, the RV *VehicleType_v2* denotes the type of vehicle 2. It can have value either *Wheeled* or *Tracked*. Arcs represent direct dependence relationships. For example, *ImageTypeReport_rpt1*, the type recorded on imaging sensor report *rpt1*, depends on *VehicleType_v2*, the actual type of *v2*, the vehicle being observed by the sensor. RVs for which we have evidence are shown in gray and probabilities are set to 100% for the value that was actually observed. For example, recall that Region 1.1 was off-road terrain; thus, evidence for *OffRoad* is applied to the node *Terrain-Type_region1_1*. Given all the evidence we have acquired, we assign 80% probability that V1 is tracked, 94% probability that V2 is tracked, and 80% probability that the danger level in Region 1.1 is high.

Manual construction of a BN like Figure 2 is feasible, but what about situations containing hundreds of vehicles and reports? For such situations, MEBN allows us to build up a complex BN out of modular pieces. Figure 3 shows a MEBN model, called an MTheory, that expresses our domain knowledge using modular components, called MFrags, that can be composed into larger models. For example, the *ImageTypeReport* MFrag expresses knowledge the reported type from an imaging sensor. The green pentagons are *context RVs* that express conditions under which the MEBN fragment is valid: *obj* is a vehicle located in region *rgn*, and *rpt* is a report about *obj*. The gray trapezoid *input RVs* have their distributions defined in other MFrags. The yellow oval *resident RV*, *ImageTypeReport(rpt)* in this case, has its distribution defined in this MFrag, and its distribution depends on the vehicle type of *obj* and the weather of *rgn*.

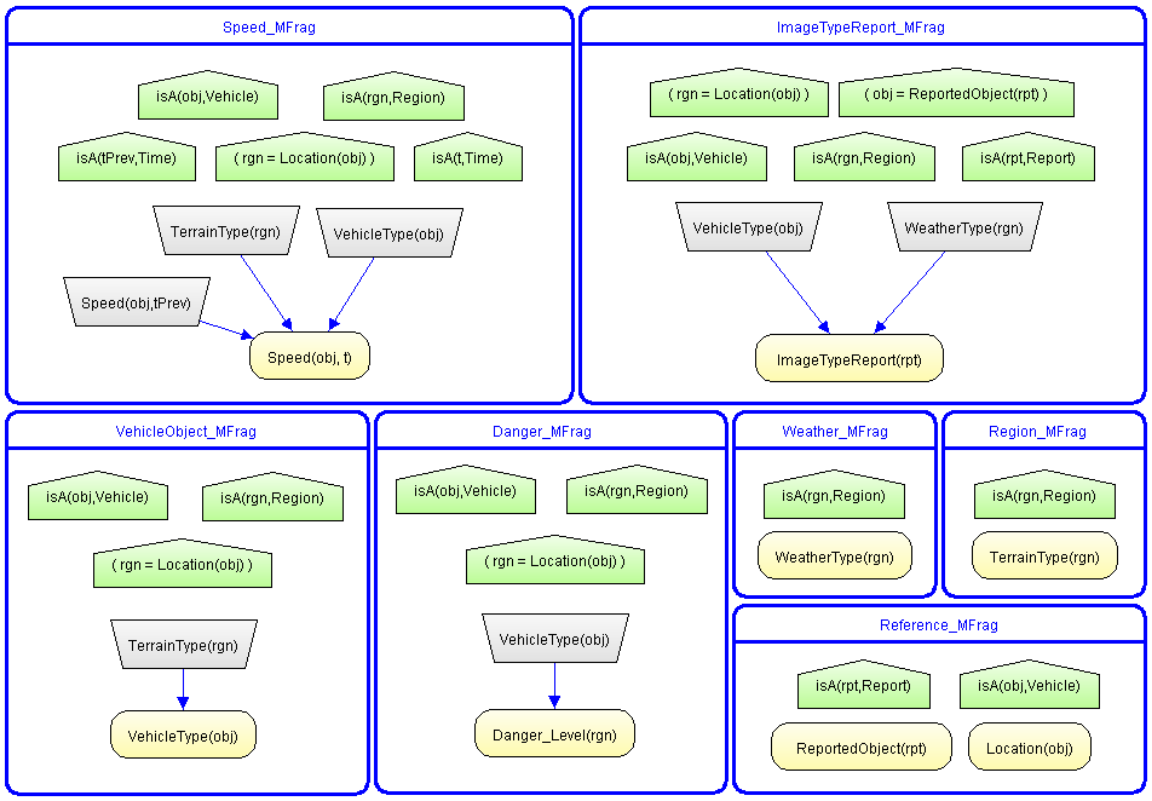
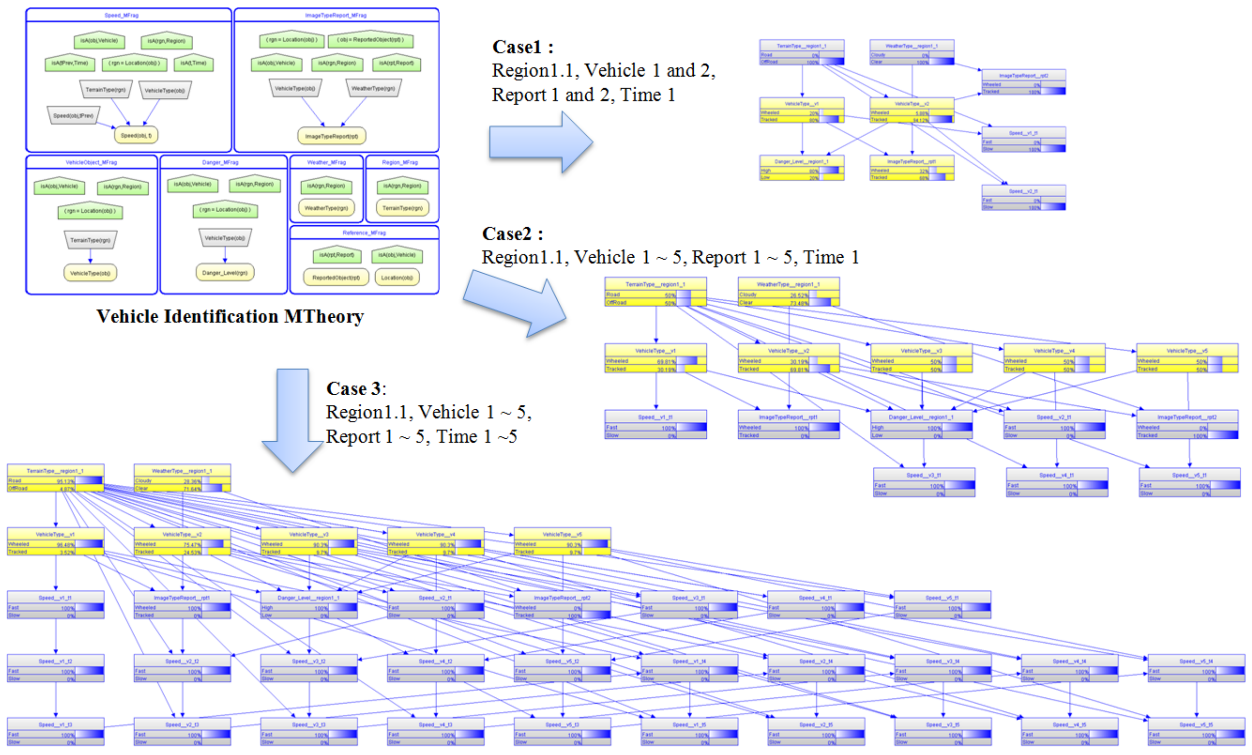


Figure 3. ***Vehicle Identification*** **MTheory**

In the *Vehicle Identification* MTheory in Figure 3, there are 7 MFrags such as *Speed*, *ImageTypeReport*, *VehicleObject*, *Danger*, *Weather*, *Region*, and *Reference* MFrag. The MTheory can generate many different BNs specialized to different situations, as depicted in Figure 4 below. Case 1 is the BN of Figure 2, representing two vehicles with two reports in a single region at a single time. Case 2 represents five vehicles with five reports in a single region at a single time. Case 3 represents are five vehicles with five reports in a single region at 5 time steps.



Vehicle Identification MTheory
Case1 :
Region1.1, Vehicle 1 and 2,
Report 1 and 2, Time 1
Case2 :
Region1.1, Vehicle 1 ~ 5, Report 1 ~ 5, Time 1
Case 3:
Region1.1, Vehicle 1 ~ 5,
Report 1 ~ 5, Time 1 ~5

Figure 4. Generated SSBNs from *Vehicle Identification* MTheory

MEBN has been applied to situation assessment [Laskey, 2000; Wright et al., 2002; Costa et al., 2005]. Its increased expressive power over ordinary BNs is an advantage for situation assessment:

"Military situation assessment requires reasoning about an unknown number of hierarchically organized entities interacting with each other in varied ways [Wright et al., 2002]."

1.2 Problem Statement

In previous applications of MEBN to situation assessment, the MTheory was constructed manually by a domain expert using the MEBN modeling process called Uncertainty Modeling Process for Semantic Technologies (UMP-ST) [Carvalho, 2011]. Manual MEBN modeling is a labor-intensive and insufficiently agile process. This paper addresses the question of how to move beyond this manual process. In particular, we focus on machine learning methods in which a MEBN theory is learned from observations on previous situations.

We assume the availability of past data from similar situations. Typically, such data are stored in relational databases. Therefore, we consider the problem of how to use data stored in a relational database for learning an MTheory. We take the standard approach of decomposing the learning problem into parameter and structure learning, treating each of these in turn.

1.3 Scope

This paper presents a basic structure and parameter learning algorithm for MEBN theories and illustrates the method on synthetic data generated from the *PROGNOS* Simulaton. We assume:

1. The data for learning are stored in a relational database.
    a. There is a single centralized database rather than multiple distributed databases.
    b. We do not consider learning from unstructured data.
2. The database contains enough observations for accurate learning.
3. There is no missing data.
4. All RVs are discrete. Continuous RVs are not considered.
5. Learning is in batch mode. We do not consider online incremental learning.
6. We do not consider the problem of learning functions for aggregating influences from multiple instances of the parents of an RV.

These assumptions will be relaxed in future work.

2 MULTI-ENTITY BAYESIAN NETWORK AND RELATIONAL MODEL

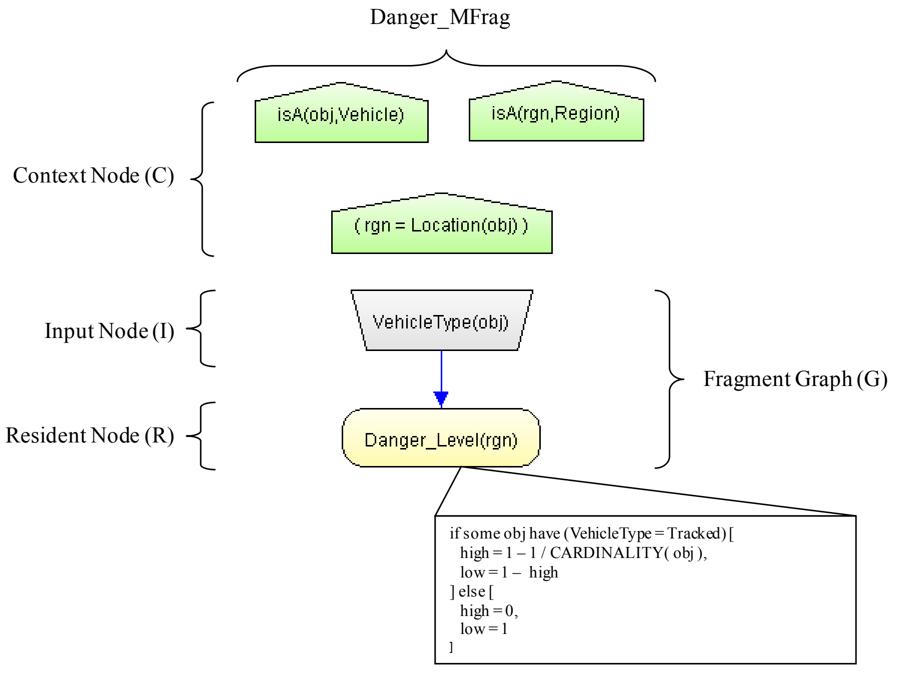
This section defines Multi-Entity Bayesian Networks (MEBN) and the Relational Model (RM). In Section 3, we present the MEBN-RM Model, a bridge between MEBN and RM that will allow data represented in RM to be used to learn a MEBN theory.

2.1 Multi-Entity Bayesian Network

MEBN represents domain knowledge as a collection of MFrags. An MFrag (see Figure 6) is a fragment of a graphical model that is a template for probabilistic relationships among instances of its random variables. Random variables in an MFrag can contain ordinary variables which can be instantiated for different domain entities. We can think of an MFrag as a class which can generate instances of BN fragments, which can then be assembled into a Bayesian network.

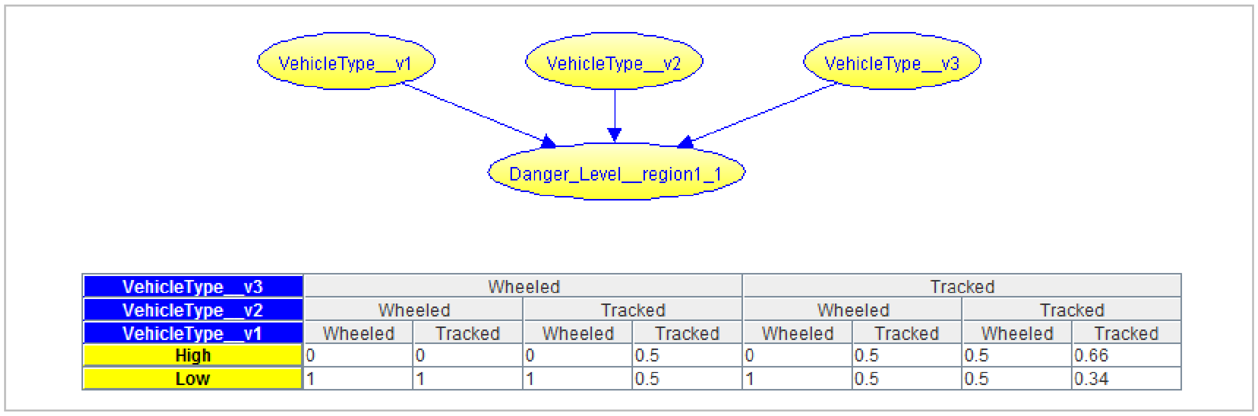
The following definition of MFrags is taken from [Laskey, 2008]. An MFrag can contain three kinds of nodes: context nodes which represent conditions under which the distribution defined in the MFrag is valid, input nodes which have their distributions defined elsewhere and condition the distributions defined in the MFrag, and resident nodes with their distributions defined in the MFrag. Each resident node has an associated local distribution, which defines its distribution as a function of the values of its parents. The RVs in an MFrag can depend on *ordinary variables*. We can substitute different domain entities for the ordinary variables to make instances of the RVs in the MFrag.

Figure 6 shows the *Danger* MFrag of the *Vehicle Identification* MTheory. The *Danger* MFrag represents probabilistic knowledge of how the level of danger of a region is measured depending on the vehicle type of detected objects. For example, if in a region there is a large number of tracked vehicles (e.g., Tanks), the danger level of the region will be high. The context nodes for this MFrag (shown as pentagons in the figure) show that this MFrag applies when a Vehicle entity is substituted for the ordinary variable *obj*, a Region entity is substituted for the ordinary variable *rgn*, and a vehicle *obj* is located in region *rgn*. The context node *rgn* = *Location*(*obj*) constrains the values of *obj* and *rgn* from the possible instances of vehicle and region. For example, suppose v1 and v2 are vehicles and r1 is a region in which only v1 is located. The context node *rgn* = *Location*(*obj*) will allow only an instance of (v1, r1) to be selected, but not (v2, r1), because r1 is not the location of v2. Next, we see the input node *VehicleType*(*obj*), depicted as a trapezoid. Input nodes are nodes whose distribution is defined in another MFrag. In Figure 6, the node *Danger_Level*(*rgn*) is a resident node, which means its distribution is defined in the MFrag of the figure. This node *Danger_Level*(*rgn*) might be an input node of some other MFrag, where it would appear as a trapezoid. Like the graph of a BN, the fragment graph shows statistical dependencies. The local distribution for *Danger_Level*(*rgn*) describes its probability distribution as a function of the input nodes given the instances that satisfy the context nodes. In our example, the argument, *rgn*, is the region variable. If the situation involves two regions, r1 and r2, then *Danger_Level*(*r1*) and *Danger_Level*(*r2*) will be instantiated. The local distribution is defined in a language called Local Probability Description (LPD) Language. In our example, the probabilities of the states, high and low, of the *Danger_Level*(*rgn*) RV are defined as a function of the values, high and low, of instances *rgn* = *Location*(*obj*) of the parent nodes that satisfy the context constraints. For the high state in the first if-scope in the LPD Language, probability value is assigned by the function described by "1 – 1 / CARDINALITY(*obj*)". The CARDINALITY function returns the number of instances of *obj* satisfying the if-condition. For example, in the LPD expression of Figure 6, if the situation involves three vehicles and two of them are tracked, then the CARDINALITY function will return 2. We see that as the number of tracked vehicles becomes very large, the function, "1 – 1 / CARDINALITY (obj)", will tend to 1.



```
if some obj have (VehicleType = Tracked) [
  high = 1 – 1 / CARDINALITY( obj ),
  low = 1 –  high
] else [
  high = 0,
  low = 1
]
```

Figure 6. *Danger* MFrag



VehicleType__v3	Wheeled				Tracked			
VehicleType__v2	Wheeled		Tracked		Wheeled		Tracked	
VehicleType__v1	Wheeled	Tracked	Wheeled	Tracked	Wheeled	Tracked	Wheeled	Tracked
High	0	0	0	0.5	0	0.5	0.5	0.66
Low	1	1	1	0.5	1	0.5	0.5	0.34

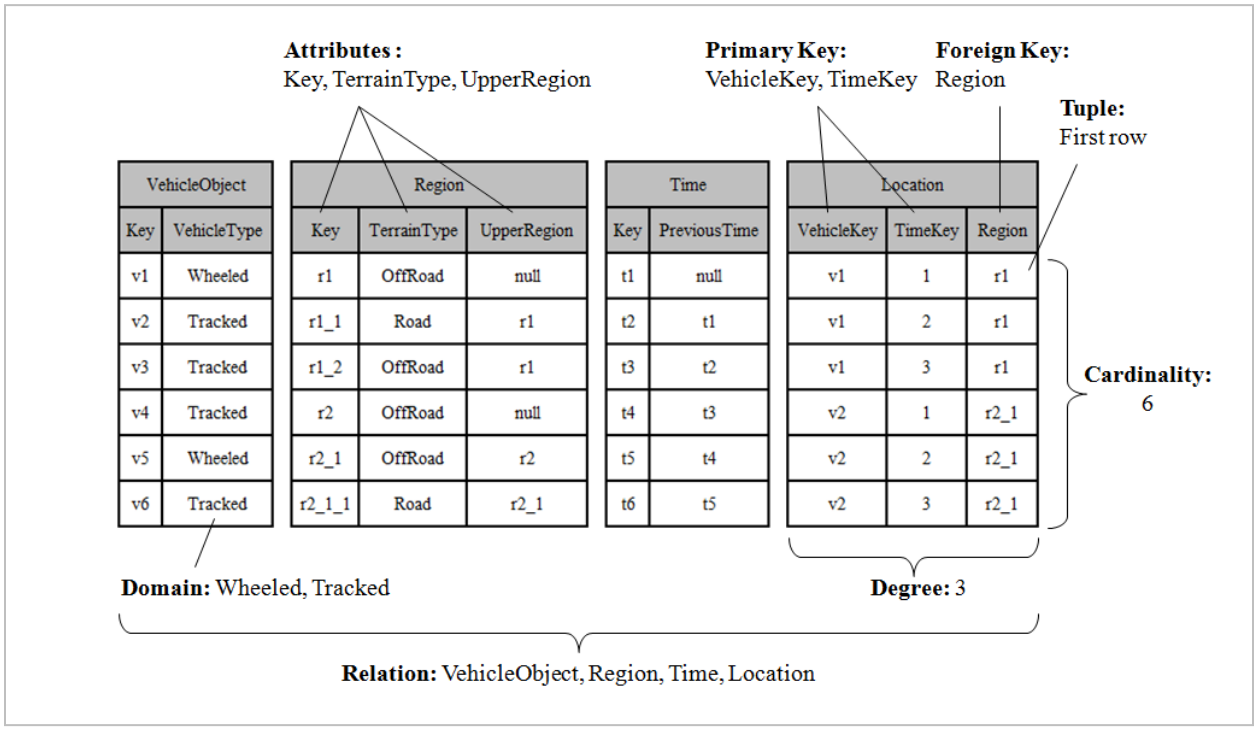
Figure 7. SSBN of *Danger* MFrag (given v1, v2, and v3 as vehicle, and region1_1 as region)

From this *Danger* MFrag, diverse situation-specific Bayesian Networks (SSBN) can be generated depending on the specific entities involved in the situation. For example, a single region entity called region1_1 and three vehicle entities called v1, v2, and v3 will give rise to the SSBN in Figure 7, with the conditional probability table (CPT) for *Danger_Level_region1_1* as shown.

An MTheory is a collection of MFrags that defines a consistent joint distribution over random variables describing a domain. The MFrags forming an MTheory should be mutually consistent. To ensure consistency, conditions must be satisfied such as no-cycle, bounded causal depth, unique home MFrags, and recursive specification condition [Laskey, 2008]. No-cycle means that the generated SSBN will contain no directed cycles. Bounded causal depth means that depth from a root node to a leaf node of an instance SSBN should be finite. Unique home MFrags means that each random variable has its distribution defined in a single MFrag, called its home MFrag. Recursive specification means that MEBN provides a means for defining the distribution for a RV depending on an ordered ordinary variable from previous instances of the RV. The *Vehicle Identification* MTheory described above is a set of consistent MFrags defining a joint distribution over situations involving instances of its RVs.

2.2 Relational Model

In 1969, Edgar F. Codd proposed the Relational Model (RM) as a database model based on first-order predicate logic [Codd, 1969; Codd, 1970]. RM is the most popular database model. A relational database (RDB) is a database that uses RM as its basic representation for data. In RM, data are organized a collection of *relations*. A relation is an abstract definition of a class of entities or a relationship that can hold between classes. An *instance* of a relation is depicted as a table in which each column is an *attribute* of the relation and each row, called a *tuple*, contains the value of each attribute for an individual entity in the domain. An entry in the table, called a *cell*, is the value of the attribute associated with the column for the entity associated with the row. A *key* is one or more attributes that uniquely identify a particular domain entity. A *primary key* for a relation uniquely identifies the individual entities in the relation; a *foreign key* points to the primary key in another relation. The *cardinality* of a relation is the number of rows in the table, i.e., the number of entities of the type represented by the relation. The *degree* of the relation is the number of columns in the table, i.e., the number of attributes of entities of the type represented by the relation.


Attributes : Key, TerrainType, UpperRegion
Primary Key: VehicleKey, TimeKey
Foreign Key: Region
Tuple: First row
Cardinality: 6
Domain: Wheeled, Tracked
Degree: 3
Relation: VehicleObject, Region, Time, Location

VehicleObject	
Key	VehicleType
v1	Wheeled
v2	Tracked
v3	Tracked
v4	Tracked
v5	Wheeled
v6	Tracked

Region		
Key	TerrainType	UpperRegion
r1	OffRoad	null
r1_1	Road	r1
r1_2	OffRoad	r1
r2	OffRoad	null
r2_1	OffRoad	r2
r2_1_1	Road	r2_1

Time	
Key	PreviousTime
t1	null
t2	t1
t3	t2
t4	t3
t5	t4
t6	t5

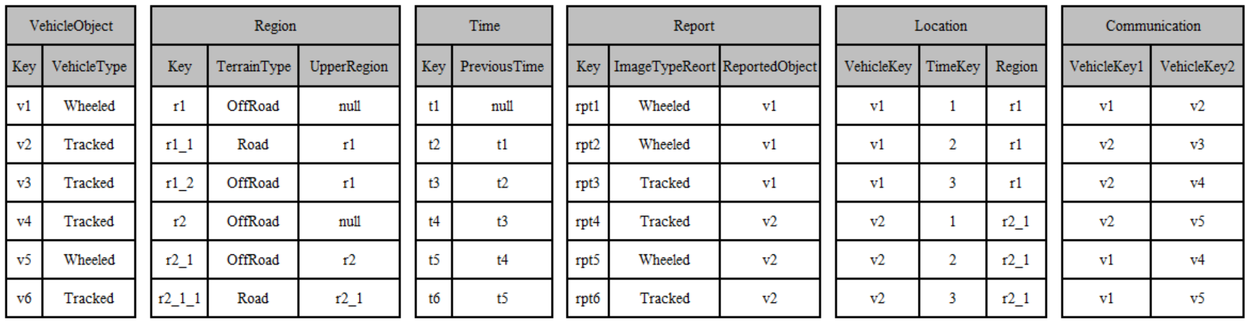
Location		
VehicleKey	TimeKey	Region
v1	1	r1
v1	2	r1
v1	3	r1
v2	1	r2_1
v2	2	r2_1
v2	3	r2_1

Figure 8. Example of Relational Model

Figure 8 shows a relational model for the vehicle identification example. There are four relations in this model: *VehicleObject*, *Region*, *Time* and *Location*. We could imagine different situations, each with different vehicles, regions, etc. Each particular situation, like the one depicted in Figure 8, corresponds to an *instance* of this relational model. The instance is represented as a table for each of the relations, where the columns represent attributes of the relation and the rows represent entities. For example the *VehicleObject* relation has two attributes: *Key*, which uniquely identifies each individual vehicle, and *VehicleType*, which indicates whether the vehicle is tracked or wheeled. The *VehicleKey* attribute in the *Location* relation is a foreign key pointing to the primary key of the *Vehicle* relation. A row of *Location* represents a vehicle being located in a region at a time.

3 MEBN-RM MODEL

As a bridge between MEBN and RM, we suggest the MEBN-RM Model, specifies how to match elements of MEBN to elements of RM. We describe this from the MEBN perspective. We begin by discussing the bridge between context RVs in MEBN and elements of RM. Next, we discuss the bridge between resident RVs in MEBN and elements of RM.



VehicleObject	
Key	VehicleType
v1	Wheeled
v2	Tracked
v3	Tracked
v4	Tracked
v5	Wheeled
v6	Tracked

Region		
Key	TerrainType	UpperRegion
r1	OffRoad	null
r1_1	Road	r1
r1_2	OffRoad	r1
r2	OffRoad	null
r2_1	OffRoad	r2
r2_1_1	Road	r2_1

Time	
Key	PreviousTime
t1	null
t2	t1
t3	t2
t4	t3
t5	t4
t6	t5

Report		
Key	ImageTypeReort	ReportedObject
rpt1	Wheeled	v1
rpt2	Wheeled	v1
rpt3	Tracked	v1
rpt4	Tracked	v2
rpt5	Wheeled	v2
rpt6	Tracked	v2

Location		
VehicleKey	TimeKey	Region
v1	1	r1
v1	2	r1
v1	3	r1
v2	1	r2_1
v2	2	r2_1
v2	3	r2_1

Communication	
VehicleKey1	VehicleKey2
v1	v2
v2	v3
v2	v4
v2	v5
v1	v4
v1	v5

Figure 9. Example Tables

Figure 9 is used as an example for the next sections. It extends the four tables of Figure 8 by adding a fifth *Report* and sixth *Communication* relation. The tables *VehicleObject, Region*, *Time*, and *Report* are called entity tables. Each of these represents a type of entity. Each primary key is a single column, which uniquely identifies the entity. For example, the *Key* column in the *Vehicle* table consists of identifiers for the six vehicles in our situation. The *Location and Communication* table is called a relationship table. The primary key of a relationship table consists of two or more foreign keys (in this case (*VehicleKey*, *TimeKey*) for the *Location* table). The *Location* table represents the region in which an entity is located at a time. The relations and their attributes – that is, a set of empty tables – is called the *schema* for the database. A populated set of tables such as Figure 9 is called an *instance* of the schema. It is clear that many different instances of this schema are possible, each corresponding to a different situation.

3.1 Context Node

In MFrags, context terms (or nodes) are used to specify constraints under which the local distributions apply. Thus, it determines specific entities on an arbitrary situation of a context.

In the MEBN-RM model, we define four types of data structure corresponding to context nodes: Isa, Value-Constraint, Slot-filler, and Entity-Constraint type.

Type	Name	Example
1	Isa	Isa(VehicleObject, obj), Isa(Region, rgn), Isa(Time, t), Isa(Report, rpt)
2	Value-Constraint	VehicleType(obj) = Wheeled
3	Slot-Filler	obj = Reported Object(rpt)
4	Entity-Constraint	Communication(obj1,obj2)

Table 1. Context Node Types on MEBN-RM Model

3.1.1 Isa Type

In MEBN, the Isa random variable (RV) represents the type of an entity. In a RM, an entity table represents a collection of entities of a given type. Thus, an entity table corresponds to an Isa random variable in MEBN. Note that a relationship table whose primary key is composed of foreign keys does not correspond to an Isa RV. A relationship table will correspond to the Entity-Constraint type of Context Node. In the example, the table of *VehicleObject*, *Region*, *Time*, and *Report* are entity tables, so they correspond to Isa RVs such as *Isa*(*VehicleObject*, *obj*), *Isa*(*Region*, *rgn*), *Isa*(*Time*, *t*), and *Isa*(*Report*, *rpt*). The primary key of an entity relation consists of the entities of the given type in our situation. For example, v1, …, v6, the entries in the *Key* attribute of the *Vehicle* relation, denote the six vehicles in the situation depicted by the RM of Figure 9.

3.1.2 Value-Constraint Type

The value of an attribute can be used to select those keys which are related to the value. For example, consider the *VehicleObject* table in which we have the Vehicle entity with the VehicleType attribute. The instances of the Vehicle entity are denoted by the primary key (e.g., v1, v2, v3, v4, v5, and v6). To focus on a case of the entity with "Wheeled" value of the attribute, we will select the set {v1, v5}. In MEBN, this corresponds to the context RV *VehicleType* (*obj*) = *Wheeled*. In this way, we can represent subsets of entities selected on the basis of the values of given attributes.

3.1.3 Slot-Filler Type

Consider the Report table depicting the Report entity which has an attribute *ReportedObject* referring to a foreign key, *VehicleObject.Key*. The *VehicleObject.Key* in the Report entity is an attribute which domain is the key of the Vehicle entity in the *VehicleObject* table. In other words, this attribute points to an entity of type Vehicle. This attribute represents the vehicle associated with the corresponding report. For example, from the first row of the table, we see that *v1* is the *ReportedObject* for the report *rpt1*. That is, *rpt1* is a report about the vehicle *v1*. We call this a *slot filler* attribute, i.e., *v1* fills the *ReportedObject* slot in the *rpt1* report. In MEBN, this slot filler relationship is expressed by *v1* = *ReportedObject*(*rpt1*).

The foreign key, *VehicleObject.Key*, is not a primary key for the *Report* table. This means it is allowed to have a "null" value, which means an empty cell (i.e. no report is available for the vehicle). The intersection set of the Vehicle and Report entity will be {(v1, rpt1), (v1, rpt2), (v1, rpt3), (v2, rpt4), (v2, rpt5), (v2, rpt6)}.

3.1.4 Entity-Constraint Type

A relationship table identifies a connection among entity tables by composing two or more keys of the entity tables. For example, the primary keys of the Communication table are *VehicleKey1* and *VehicleKey2* from the *VehicleObject* table. Composing two keys expresses a relationship between the entities. The connection between entities corresponds to the Entity-Constraint type in MEBN-RM. The Entity-Constraint node *Communication* (*obj1*, *obj2*) in MEBN expresses a relation on vehicle entities. From Figure 9, we see that this relation is{(v1, v2), (v2, v3), (v2, v4), (v2, v5), (v1, v4), (v1, v5)}. This relation corresponds to the set of pairs of communicating vehicles.

3.2 Resident Node

In MFrags, Resident Node can be described as Function, Predicate, or Formula of FOL. MEBN allows the modeler to specify a probability distribution for the truth-value of a predicate or the value of a function. Formulas are not probabilistic, and are defined by built-in MFrags [Laskey, 2008]. As noted above, RM is based on first-order predicate logic. In this section, we describe the correspondence between functions and predicates in FOL and relations in RM.

In FOL, a predicate represents a true/false statement about entities in the domain. It is expressed by a predicate symbol followed by a list of arguments. For example, *Communication*(*x*,*y*) is a predicate that expresses whether the entities indicated by the arguments *x* and *y* are communicating. In MEBN, this predicate corresponds to a Boolean RV with possible values *True* and *False*. In RM, we express a predicate as a table in which the primary key consists of all the attributes. These attributes are the arguments of the predicate, and the rows of the table represent the arguments for which the predicate is true. For example, the six rows of the *Communication* relation of Figure 9 correspond to the six pairs of entities for which the predicate *Communication* holds.

In FOL, a function is a mapping from domain entities called *inputs* to a value called the *output*. For example, the function *VehicleType*(*obj*) is a function that maps its argument to *Wheeled* if it is a wheeled vehicle and *Tracked* if it is a tracked vehicle; *ReportedObject*(*rpt*) is a function that maps its argument to the object being reported upon. In RM, a function is represented by a non-key attribute of a table. It maps its argument(s), the primary key(s) for the relation, to the output, which is the value of the attribute. For example, in Figure 9, the argument of the function *VehicleType* is the primary key of the *VehicleObject* relation, and the output is the value (either *Tracked* or *Wheeled*) of the *VehicleType attribute.*

Table 2 defines the relationship between elements of RM and MEBN.

RM	Resident Node
Attribute	Function/ Predicate
Key	Arguments
Cell of Attribute	Output

Table 2. Function of MEBN-RM Model

4 THE BASIC PARAMETER AND STRUCTURE LEARNING FOR MEBN

This section presents a basic structure and parameter learning method for learning a MEBN theory from a relational database.

4.1 Basic MEBN Parameter Learning

Parameter learning for MEBN is to estimate a parameter of the local distribution for a resident node of an MTheory, given the structure of the MTheory and a dataset expressed in RM. By structure, we mean the nodes, arcs and state spaces in each MFrag, and the parameters of the local distributions for the resident nodes. For this basic algorithm, we use Maximum Likelihood Estimation (MLE) to estimate the parameter. Furthermore, we do not address the problem of the aggregating influences from multiple instances of the same parent. We assume that the test dataset is well modeled by an MTheory with nodes and state spaces as given by the relational database, and that the local distributions are well modeled by the chosen parametric family. In future research, we will address the use of an informative prior distribution to represent *a priori* information about the parameters.

The influence aggregation problem occurs when there are multiple instances of the parents of a resident node that satisfy the context constraints in the MFrag. In this case, a domain expert may provide knowledge about how random variables are aggregated, and an aggregator or combining rule may be used for estimating the parameter [Getoor et al., 2000; Natarajan et al., 2009]. We defer consideration of aggregators and combining rules to future work.

4.2 Basic MEBN Structure Learning

Structure learning for MEBN is to organize RVs into MFrags and identify parent-child relationships between nodes, given a dataset expressed in RM. The MFrags, their nodes (context, input, and resident nodes), and arcs between nodes are learned (See appendix A). The initial ingredients of the algorithm are the dataset (DB) expressed in RM, any Bayesian Network Structure searching algorithm (BNSL_alg), and maximum size of chain (Sc). We utilize a common Bayesian Network Structure searching algorithm to generate a local BN from the joined dataset of the RM.

The first step of the algorithm is to create the default MTheory. All keys in entity tables of the DB are defined as entities of this default MTheory. One default reference MFrag is created, which will include resident nodes used for context nodes. Because context nodes also are random variables, they should be defined an MFrag such as the reference MFrag. Now, using both entity and relationship tables, the MFrags, their nodes, and their connections are learned. There are three For-Loop (#4, #10, and #23in appendix A). The first For-Loop treats all tables, while the second For-Loop treats the joined tables. For all tables of the DB, the dataset for each table is retrieved one by one and, by using any BN structure searching algorithm (BNSL_alg), a graph is generated from the retrieved dataset. If the graph has a cycle and undirected edge, a domain expert sets the arc direction manually. Based on the revised graph, an MFrag is created by using *createMFrag* function in appendix A. In the second For-Loop, for the joined tables, data associated with relationship tables is retrieved until the maximum size of chain (Sc) is reached. This iteration continues until a user-specified maximum size of chain is reached. The MFrags, their nodes, and their arcs are generated in the same way as described in the previous paragraph. One difference is that the aggregating influence situation should be detected by an approach called Framework of Function Searching for LPD (FFS-LPD) which will detect the situation and provide possible LPD function in a heuristic approach. FFS-LPD can be realized by a domain expert or a program. In our initial research, the domain expert detects the aggregating influence situation and provides a reasonable LPD function having aggregating function in FFS-LPD context (An automatic programmed approach is being researched). After checking the LPD function, if any nodes of the new generated graph are not used in any MFrags, create a new resident node having the name of the dataset of the graph on the default reference MFrag and a new MFrag for the dataset. If not, add make edges between resident nodes corresponding to arcs found by the structure learning algorithm. If there is an arc between nodes in different MFrags, add the parent node as an input node to the MFrag of the child node. Lastly, in the third For-Loop, for all resident nodes in the MTheory, LPDs are generated by MLE.

5 CASE STUDY

As noted in Section 1, the purpose of the learned MTheory generated by the presented algorithm is to estimate and predict a situation in PSAW. In this case study, a learned MTheory is evaluated by evaluating its ability to predict queries of interest.

Our case study uses PROGNOS (Probabilistic OntoloGies for Net-centric Operation Systems) [Costa et al., 2009; Carvalho et al., 2010]. PROGNOS includes a simulation which generates simulated ground truth information for the system. The simulation generates 85000 persons, 10000 ships, and 1000 organization entities with various values of attributes and relations between entities. The data for these entities are stored in a relational database which includes three entity tables (*person*, *ship*, and *organization*) and two relationship tables (*ship_crews* and *org_members*). The *ship_crews* table has a paired key comprised of a *ShipKey* and *PersonKey*, representing the persons serving as crew members on ships. The *org_members* table has a paired key comprised of an *OrganizationKey* and *PersonKey*, representing membership of persons in organizations. A ship may have many crew members, each of whom may be affiliated with several organizations. The goal of PROGNOS is to classify ships as to whether they are ships of interest. In our case, this means ship associated with terrorist activities. The classification is made given evidence about the attributes of the entities. For example, if a ship had a crew member who has communicated with a terrorist, the ship was on an unusual route, and it was unresponsive, it is highly likely that the ship is likely to be a ship of interest. The database contains an attribute *IsShipOfInterest* of the *ship* table representing the ground truth for whether it is a ship of interest.

To evaluate the algorithm, training and test datasets were generated by the simulation. The algorithm was used to learn an MTheory from the training dataset as shown in Figure 11 (one of SSBNs from the learned MTheory is shown in Figure 12). In the MTheory, a total of four MFrags were generated. There is the default reference MFrag, the *org_members* MFrag from the *org_members* relationship table, the *person* MFrag from the *person* entity table, and the *ship* MFrag from the *ship* entity table. The *org_members* and *ship_crews* input nodes came from the *org_members* and *ship_crews* relationship tables. After learning the MTheory, the test dataset was used to evaluate the MTheory. First, a test case from the test dataset was retrieved. Because a state of the *IsShipOfInterest* variable is our concern, data from a ship in the test dataset was retrieved. Based on the ship data, other related data were retrieved. All of these were combined to make the test data. For example, if a ship was connected to 3 persons and each of the 3 persons was associated with 3 organizations, then 9 rows of a joined table were retrieved as one test case. Using this test case, a SSBN was generated from the learned MTheory. The context of the SSBN corresponds to the context of the test case. For example, using the previous test case example, 1 ship, 3 person, and 9 organization entities are used for generating a SSBN. After the SSBN is generated, the *IsShipOfInterest* node which was Boolean was queried given several leaf nodes of the SSBN with values of the leaf nodes retrieved from the test data. The queried probability result was stored in an array. This retrieving and querying process continued until all ships were treated.

For each of the SSBNs generated from the test data, and for each instance of the *IsShipOfInterest* RV in the SSBN, the probability of the *IsShipOfInterest* RV was computed given the evidence for the leaf nodes. The accuracy of the queried probability results was measured using the Receiver Operating Characteristic (ROC) Curve. The ROC for our case study is shown in Figure 10. The area under the curve (AUC) is shown in Table 3. The learned MTheory estimated the state of the *IsShipOfInterest* node with the AUC, 0.897206546.

Model	AUC
Learned MTheory	0.897206546

Table 3. AUC of Learned MTheory

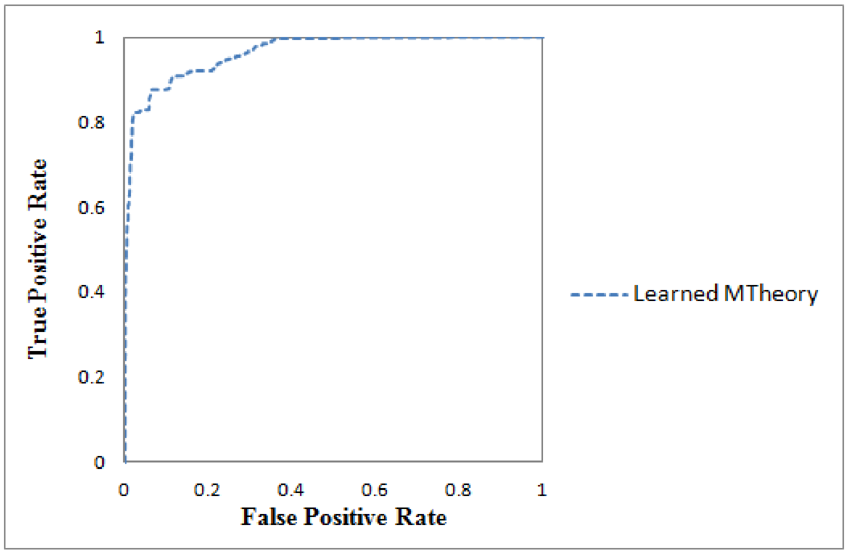


Figure 10. ROC of Learned MTheory

6 DISCUSSION AND FUTURE WORK

This paper discussed reasons why MEBN is a useful modeling tool for PSAW systems, providing a semantically rich representation that also captures uncertainty. MEBN was the core logic for the probabilistic ontologies used in the PROGNOS prototype PSAW system. The original PROGNOS probabilistic ontologies were constructed manually with the help of domain experts. This manual MEBN modeling was labor-intensive and insufficiently agile. To address this problem, we developed a learning algorithm for MEBN-based probabilistic ontologies. To enable learning from relational databases, we presented a bridge between MEBN and the Relational Model, which we call the MEBN-RM model. We also presented a basic parameter and structure learning algorithm for MEBN. Finally, the presented method was evaluated on a case study from PROGNOS.

Although we provided a basic MEBN learning, there are several issues. 1) Aggregating influence problem; how to learn an aggregating function in an aggregating situation where an instance child random variable depends on multiple instance parents which is generated from an identical class random variable? 2) Optimization of learned MTheory; how to learn an optimized structure of an MTheory without losing accuracy of query? 3) Unstructured data learning; how to learn unstructured data which isn't derived from a data model? 4) Continuous random variable learning; how to learn an MTheory which includes continuous random variables? 5) Multiple distributed data learning; how to learn an MTheory from data in multiple distributed databases? 6) Incomplete data learning; how to approximate parameters of an MTheory from missing data? 7) Learning in insufficient evidence; how to learn an MTheory from not enough observations? 8) Incremental MEBN learning; how to learn parameters of an MTheory from updated observations? There remain many open research issues in this domain. Recently, we are studying about the aggregating influence problem and continuous random variable learning in PSAW.

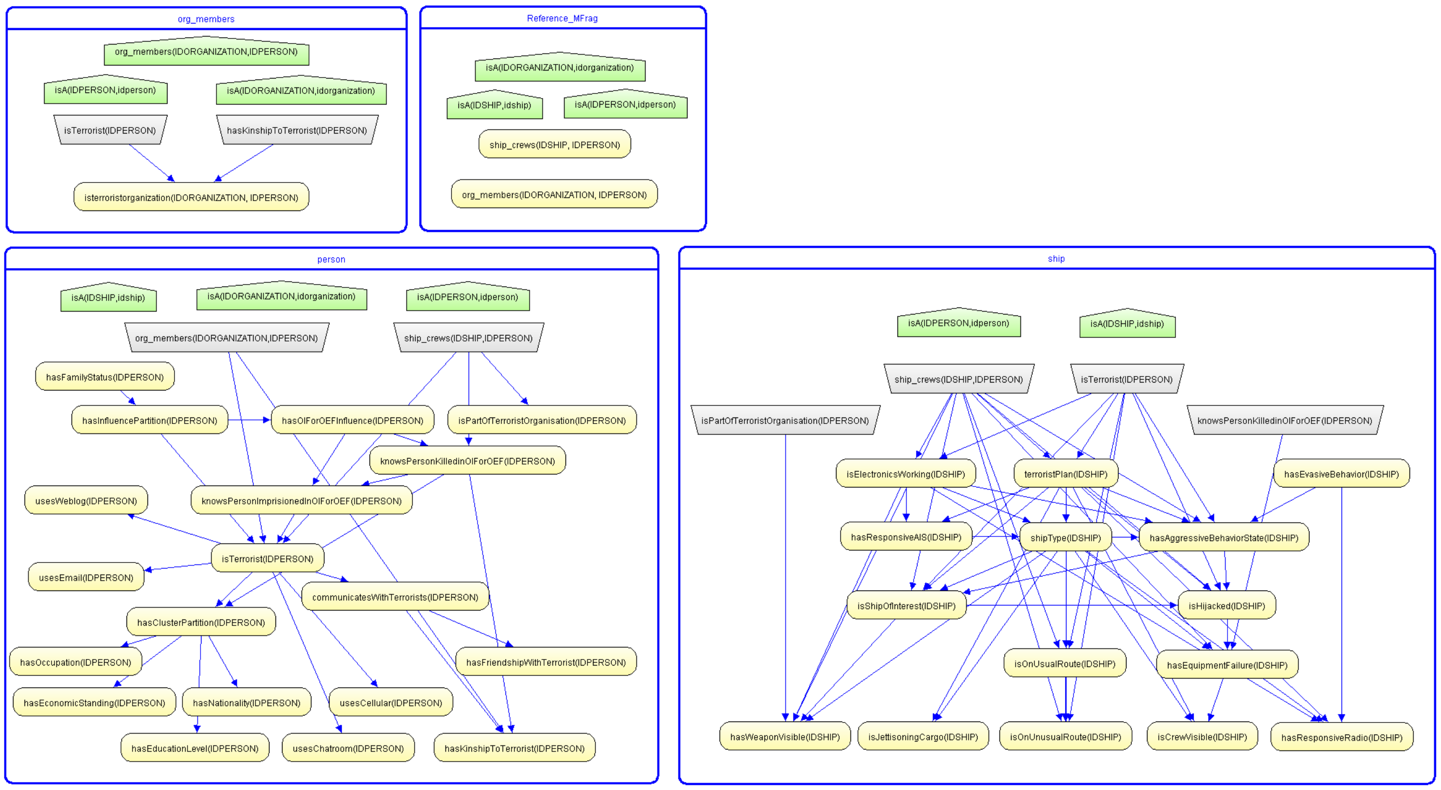

org_members
org_members(IDORGANIZATION,IDPERSON)
isA(IDPERSON,idperson)
isA(IDORGANIZATION,idorganization)
isTerrorist(IDPERSON)
hasKinshipToTerrorist(IDPERSON)
isterroristorganization(IDORGANIZATION, IDPERSON)
Reference_MFrag
isA(IDORGANIZATION,idorganization)
isA(IDSHIP,idship)
isA(IDPERSON,idperson)
ship_crews(IDSHIP, IDPERSON)
org_members(IDORGANIZATION, IDPERSON)
person
isA(IDSHIP,idship)
isA(IDORGANIZATION,idorganization)
isA(IDPERSON,idperson)
org_members(IDORGANIZATION,IDPERSON)
ship_crews(IDSHIP,IDPERSON)
hasFamilyStatus(IDPERSON)
hasInfluencePartition(IDPERSON)
hasOIForOEFInfluence(IDPERSON)
isPartOfTerroristOrganization(IDPERSON)
knowsPersonKilledinOIForOEF(IDPERSON)
usesWeblog(IDPERSON)
knowsPersonImprisionedInOIForOEF(IDPERSON)
isTerrorist(IDPERSON)
usesEmail(IDPERSON)
communicatesWithTerrorists(IDPERSON)
hasClusterPartition(IDPERSON)
hasFriendshipWithTerrorist(IDPERSON)
hasOccupation(IDPERSON)
hasEconomicStanding(IDPERSON)
hasNationality(IDPERSON)
usesCellular(IDPERSON)
hasEducationLevel(IDPERSON)
usesChatroom(IDPERSON)
hasKinshipToTerrorist(IDPERSON)
ship
isA(IDPERSON,idperson)
isA(IDSHIP,idship)
ship_crews(IDSHIP,IDPERSON)
isTerrorist(IDPERSON)
isPartOfTerroristOrganisation(IDPERSON)
knowsPersonKilledinOIForOEF(IDPERSON)
isElectronicsWorking(IDSHIP)
terroristPlan(IDSHIP)
hasEvasiveBehavior(IDSHIP)
hasResponsiveAIS(IDSHIP)
shipType(IDSHIP)
hasAggressiveBehaviorState(IDSHIP)
isShipOfInterest(IDSHIP)
isHijacked(IDSHIP)
isOnUsualRoute(IDSHIP)
hasEquipmentFailure(IDSHIP)
hasWeaponVisible(IDSHIP)
isJettisoningCargo(IDSHIP)
isOnUnusualRoute(IDSHIP)
isCrewVisible(IDSHIP)
hasResponsiveRadio(IDSHIP)

Figure 11. Learned PROGNOS MTheory

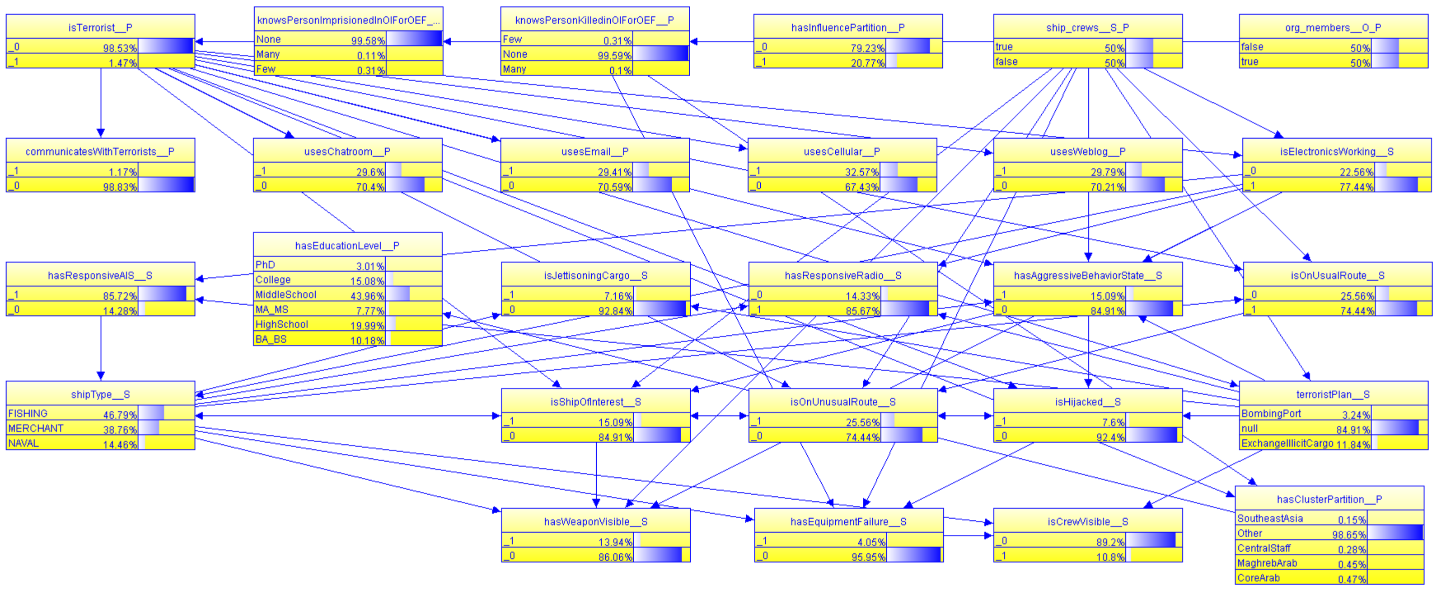


Figure 12. Generated SSBN from Learned *PROGNOS* MTheory. *(_1 and _0 in the state of the node means true and false respectively. The letter S, O, and P in the title of the node means Ship, Organization, and Person respectively.)*

REFERENCES

Breton, R., & Reousseau, R. (2001). *Situation Awareness: A Review of the Concept and its Measurement.* Technical Report No. 2001-220, Defense Research and Development Canada, Valcartier.

Carvalho, R. N., Costa, P. C. G., Laskey, K. B., & Chang, K. C. (2010). *PROGNOS: predictive situational awareness with probabilistic ontologies.* In Proceedings of the 13th International Conference on Information Fusion. Edinburgh, UK.

Carvalho, R. N. (2011). *Probabilistic Ontology: Representation and Modeling Methodology*. PhD Dissertation. George Mason University.

Codd, E. F. (1969). *Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*. IBM Research Report.

Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM.

Costa, P. C. G. (2005). *Bayesian Semantics for the Semantic Web*. PhD Dissertation. George Mason University.

Costa, P. C. G., Laskey, K. B., Takikawa, M., Pool, M., Fung, F., & Wright, E. J. (2005). *MEBN Logic: A Key Enabler for Network Centric Warfare*. In Proceedings of the 178 Tenth International Command and Control Research and Technology Symposium (10th ICCRTS). Mclean, VA, USA: CCRP/DOD publications.

Costa, P. C. G., Laskey, K. B., & Chang, K. C. (2009). *PROGNOS: Applying Probabilistic Ontologies To Distributed Predictive Situation Assessment In Naval Operations.* Proceedings of the 14th Int. Command And Control Research and Technology Symposium. Washington, D.C., USA.

Endsley, M. R. (1988). *Design and evaluation for situation awareness enhancement.* Paper presented at the Human Factors Society 32nd Annual Meeting, Santa Monica, CA.

Endsley, M. R., Bolte, B., & Jones, D. G. (2003). *Designing for situation awareness: An approach to human-centered design*. New York, NY: Talyor & Francis.

Getoor, L., Koller, D., Taskar, B., & Friedman, N. (2000). *Learning Probabilistic Relational Models with Structural Uncertainty*. Paper presented at the ICML-2000 Workshop on Attribute-Value and Relational Learning:Crossing the Boundaries. Stanford, CA, USA.

Laskey, K. B., D'Ambrosio, B., Levitt, T. S., & Mahoney, S. M. (2000). *Limited Rationality in Action: Decision Support for Military Situation Assessment*. Minds and Machines, 10(1), 53-77.

Laskey, K. B. (2008). *MEBN: A Language for First-Order Bayesian Knowledge Bases*. Artificial Intelligence, 172(2-3).

Linggins, M. E., Hall, D. L., & Llinas, J. (2008). *Handbook of Multisensor Data Fusion: Theory and Practice, Second Edition*. Electrical Engineering & Applied Signal Processing Series. CRC Press.

Llinas, J., Bowman, C., Rogova, G., & Steinberg, A. (2004). *Revisiting the JDL data fusion model II.* In: Proc. of the 7th Int. Conf. on Information Fusion, Stockholm, Sweden, pp. 1218–1230.

Natarajan, S., Tadepalli, P., Dietterich, T. G. & Fern, A. (2009). *Learning first-order probabilistic models with combining rules*. Special Issue on Probabilistic Relational Learning, AMAI.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann Publishers.

Steinberg, A. N., Bowman, C. L., & White, Jr., F. E. (1998). *Revisions to the JDL Data Fusion Model*. Proc. 3rd NATO/IRIS Conf. Quebec City, Canada.

White, Jr., F. E. (1988). *A model for data fusion*. Proc. 1st Natl. Symp. Sensor Fusion, vol. 2.

Wright, E., Mahoney, S. M., Laskey, K. B., Takikawa, M. & Levitt, T. (2002). *Multi-Entity Bayesian Networks for Situation Assessment*. Proceedings of the Fifth International Conference on Information Fusion.

APPENDIX A

Algorithm 1: Basic Structure Learning For MEBN

```
Procedure BSL_MEBN ( $DB$,          // Relational database
                     $BNSL\_alg$    // BN Structure Search algorithm
                     $Sc$           // Maximum size of chain
         )
1   $M_{theory}$ ← create a default MTheory
2   $M_{theory}$ ← add entities from the all keys in the tables of $DB$
3   $MF_{ref}$ ← create a default reference MFrag
4   for $i$ = 1, … until size of all tables in $DB$
5     $T_i$ ← get table from $DB$
6     $G_i$ ← search the graphs in $T_i$ using $BNSL\_alg$
7     $G_i$ ← revise the graph to ensure no cycle and undirected edge
8     if $G_i \neq Ø$ then
9       $MF_i$ = createMFrag($G_i$, $T_i$, $M_{theory}$)
10  for $c$ = 1, … until $sc$
11    $JT$ ← joinTables($DB$, $c$)
12    for $i$ = 1, … until size of $JT$
13      $G_i$ ← search the aggregating graphs using $FFS\text{-}LPD$
14      $G_i$ ← search the graphs in $JT_i$ using $BNSL\_alg$
15      $G_i$ ← revise the graph to ensure no cycle and undirected edge
16      if $G_i \neq Ø$ then
17        for $j$ = 1, … until size of $G_i$
18          if any nodes in $G_{ij}$ is not used for any MFrag then
19            $MF_{ref}$ ← create the resident node with the name of $JT_i$ on $MF_{ref}$
20            createMFrag($G_i$, $JT_i$, $M_{theory}$)
21          else
22            addEdges($G_i$, $JT_i$, $Ø$)
23  for $i$ = 1, … until size of all resident nodes in the MTheory
24    $T_{i,}$ ← get dataset related the resident node i
25    calculateLPD($R_i$, $T_{i,}$)
26  return $M_{theory}$
```

```
Procedure createMFrag ( $Gø$          // List of Resident Nodes
                        $Tø$          // dataset of table
                        $M_{theory}$  // Mtheory
         )
1   $MF$ ← create MFrag using the name of $Tø$
2   $N$ ← get the nodes of $Gø$ which is not used for any Mfrags of $M_{theory}$
3   $R$ ← create the resident nodes corresponding to $N$
4   $MF$ ← add R into MF with ordinary variables related with $R$
5   $MF$ ← addEdges($Gø, Tø, MF$)
6   Add MFrag into $M_{theory}$
7   return $MF$
```

```
Procedure addEdges ($Gø$    // List of Resident Nodes
                    $Tø$    // dataset of table
                    $MF$    // the target Mfrag
         )
1   for $i$ = 1, … until the size of the edges of $Gø$
```

```
2   $N_p$ ← get the resident node corresponding to the parent node of $E_i$
3   $N_c$ ← get the resident node corresponding to the child node of $E_i$
4   $MF_p$ ← get the MFrag of $N_p$
5   $MF_c$ ← get the MFrag of $N_c$
6   if $MF = MF_c = MF_p$ then
7     $MF$ ← add edges between $N_p$ and $N_c$ using $E_i$
8   else
9     if $MF_p \neq MF$ then
10      $MF_p$ ← create the input node which was the context node of $MF$ and add it into $MF_p$
11    if $MF_c \neq MF$ then
12      $MF_c$ ← create the input node which was the context node of $MF$ and add it into $MF_c$
13    $MF_c$ ← create the input node from $N_p$ and add it into $MF_c$
14  return $MF$
```

```
**Procedure** calculateLPD ($R$      // List of Resident Nodes
                            $Tø$     // dataset of table
           )
1   for $i$ = 1, … until size of $R$
2     $R_i$. $LPD$ ← calculate default probabilities of $R_i$ using $Tø$
3     if $R_i$ is in Many-to-One connection then
4       $R_i$.$LPD$ ← assigned the LPD which is generated by $FFS$-$LPD$
5     else
6       $R_i$. $LPD$ ← calculate the conditional probabilities of $R_i$
```

```
**Procedure** joinTables ($DB$,     // Relational database
                          $c$       // range of the chain
           )
1   $RT$ ← get the relationship tables of $DB$
2   for $i$ = 1, … until size of $RT$
3     $jt$ ← join all related tables in the range, $c$, from $RT_i$
4     $JT$ ← add $jt$ into $JT$ except the $jt$ already added
5   return $JT$
```