First Order Conditional Influence Language

Sriraam Natarajan and Eric Altendorf
School of EECS,
Oregon State University
USA
Technical Report CS05-30-01

September 23, 2005

1 Introduction

Traditionally, machine learning techniques learn from data in the form of examples. Each example has a fixed set of attributes and the examples are assumed to be independent of each other. But in the real world, data is relational i.e., the data has objects and relations. For instance, in medical domain, to determine the bloodtype of a person, we need to know the parent's bloodtypes. In a university domain, there are students, professors, courses etc and relationships exist between these entities (students take courses, professors teach courses etc). Classical machine learning techniques flatten these relational data i.e., propositionalize the data and learn from it. The model learnt by this method will not be a general one. If generalization has to be achieved, then the model has to be relational. It becomes imperative to investigate methods with first-order abilities to elegantly represent objects and relations.

In the past decade, researchers combined the relational models with probability. This led to the idea of probabilistic logic models where prior knowledge could be expressed in the form of probabilistic influence relationships over relational data. These formalisms include probabilistic relational models (PRMs) [7], directed acyclic probabilistic entity-relationship (DAPER) models [10], Bayesian logic programs (BLPs) [12], probabilistic horn abduction [18], probabilistic logic programs (PLPs) [16], stochastic logic programs (SLPs) [14], Bayesian logic (BLOG) models [13], relational Bayesian networks (RBNs) [11], Markov logic models (ML) [3], and others. While the details of syntax and semantics differ, they all share the attractive property of allowing the modeler to specify the qualitative relationships in the domain that form the basis of a learnable quantitative probabilistic model.

Many of these models such as BLPs and PRMs specify the conditions under which the attribute values of some objects influence the attribute values of related objects. For example, a person's grade in a class he takes depends on the difficulty of the class and his intelligence [7]. Moreover, it might roughly

monotonically decrease with the difficulty and monotonically increase with intelligence. Specifying such known relationships in advance is expected to make the learning task easier by identifying the relevant attributes and the qualitative nature of the relationship between attributes, e.g., monotonicity.

In this paper, we introduce a simple notation that we call First-Order Conditional Influence (FOCI) statements that capture relationships of the above kind. Our modeling language has two primary components: first, an object-oriented or entity-relation modeling language for expressing the relational structure of the dataset, and second, a qualitative modeling language, which allows us to express qualitative domain knowledge about probabilistic relations and influences. A FOCI statement specifies a first-order condition under which some attributes of objects influence other attributes and relationships in a qualitatively prescribed manner[1]. The applicability condition is expressed in terms of known relationships between different objects. When these relationships are not one-to-one, one needs combining rules or aggregators to summarize the individual instance-level influences.

The rest of the paper is organized as follows. We provide an introduction to the modeling of data using the ER model. In the next section, we introduce our language, provide the syntax and semantics of the language with examples. We also explain how the aggregators and combining rules are employed in our language. In the third section, we compare our language with some existing formalisms by transforming the examples in those formalisms to our language. We then conclude the paper by outlining some areas of future work. In the appendix, we introduce the qualitative constraints, provide their semantics and show how they are used in our FOCI statements. We also provide a few examples of applying the qualitative influence statements on a contextual model.

1.1 ER Modeling

The Entity Relationship diagram is a graphical representation that is used for data modeling. The design of an ER model is the first step in creation of a relational database. The ER model can be designed before the collection of data and is meant to anticipate the data and relationships. The emphasis is on the representation of the schemas and not the instances [5].

Figure 1 displays an ER diagram for a student domain. We briefly review the ER diagram notation. Entity types such as $Student$ and $Course$ are shown in rectangular boxes. Entities are objects that exist in the world[5]. Each entity has an attribute that describes the entity. The attributes are shown in ovals. $Course$ has an attribute $Diff$ and $Student$ has an attribute Sat that represents his/her satisfaction level. Note that we have omitted other attributes like Id, $Name$ etc for brevity. $Takes$ is the relationship between the entities $Course$ and $Student$ and is represented as a diamond in the Figure 1. Here $Takes$ is a M-M relationship as a student can take many courses and a course can be taken by many students. The relationship has an attribute $Grade$ which indicates

[1] We use objects to mean the entities and the relationships

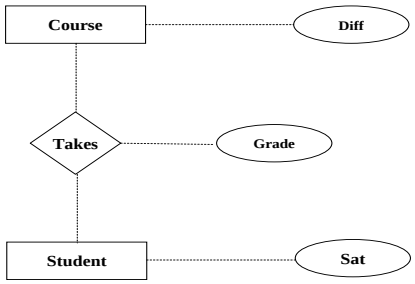


Figure 1: An Example of ER diagram

the grade of a student in a particular course. We use this ER diagram in the subsequent sections to illustrate the features of our language.

2 First Order Conditional Influence Language

The core of our language consists of first-order conditional influence (FOCI) statements, which are used to specify probabilistic influences among the attributes of objects in a given domain. The domain expert is assumed to know the structure of the ER diagram while he writes down the FOCI statements. Each FOCI statement has the form:

$$\textit{If } \langle \textit{condition} \rangle \textit{ then } \langle \textit{ qualitative influence } \rangle$$

where *condition* is a conjunction of literals, each literal being a predicate symbol applied to the appropriate number of variables. Usually, the conditions are used to identify the objects that participate in the qualitative influence. A $\langle$*qualitative influence*$\rangle$ is of the form $X_1, \ldots, X_k$ *Qinf* Y, where the X_i and Y are of the form $V.a$, where V is a variable in *condition* and a is an object attribute. This statement simply expresses a directional dependence of the *resultant* Y on the *influents* X_i. Associated with each FOCI statement is a *conditional probability function* that specifies a probability distribution of the resultant conditioned on the influents, e.g. $P(Y|X_1, \ldots, X_k)$ for the above statement. Consider for example the FOCI statement,

```
If {Person(p)} then p.diettype Qinf p.fitness.
```

which indicates that a person's type of diet influences their fitness level. There is an entity $Person$ and $diettype$ and $fitness$ are two attributes of $Person$. As can be seen, the influent and the resultants are of the form $Object.attribute$. Also, the condition is used to determine the type of the variable p, which is this case is a $Person$. The conditional probability distribution $P(p.\text{fitness} \mid p.\text{diettype})$ associated with this statement (partially) captures the quantitative relationships between these attributes.

Consider the ER diagram in Figure 1. We could say something like,

`If {Student(s), Course(c), Takes(t,s,c)} then c.diff, t.grade Qinf s.satisfaction.`

which means that the satisfaction of a student depends on the difficulty of the course that he took and the grade he obtained in that course. This would correspond to annotating the ER diagram with hyper-arcs as shown in Figure 2. We could have used two separate arcs from *Grade* to *Sat* and from *Diff* to *Sat* on the lines of the DAPER model [10] instead of the hyper-arc. But this will not be sufficient as the *student* and the *course* are tied using the *takes* relationship and so while writing the constraints on the arc between *Diff* and *Sat*, we cannot ignore $Takes$.[2]

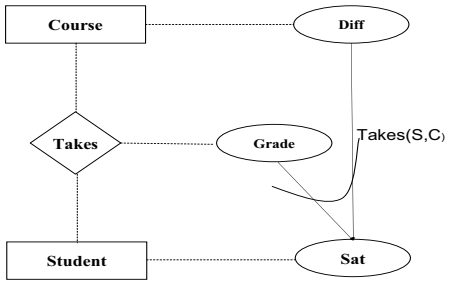


Figure 2: ER Diagram annotated with hyper-arcs for the student satisfaction example

Figure 3.a shows a part of the database (the entire database is not presented). There are three *Courses* and three *Students*. The relationship $Takes$ contains a tuple for each *Student-Course* pair.[3] Figure 3.b shows the ground bayesian network corresponding to the FOCI statement instantiated with values from the database. The instantiation of the $Takes$ relationship is represented as $T_{studentcourse}$. For instance T_{J515} represents the *takes* relationship between *John* and $CS515$. As can be seen, the instantiated values of the influents and the resultants are the random variables in the ground bayesian network.

This example provides an insight into the semantics of the language. We could imagine each attribute of each object in the entire database to be a node of a giant bayesian network and the FOCI statements specify the arcs in the bayesian network i.e., the parents of a particular node is determined by the conditions of the FOCI statement. In this case, the condition is used to select the correct $student-course$ pair from the database. In our language, we restrict the conditions to conjunctions of predicates as our primary goal is not to design the most expressive language but to design the most useful and tractable one. Each

[2] We explain this difference between the DAPER model and ours in the next section.

[3] Note that each student takes only one course in this example. We would later deal with examples in which a student can take more than one course.

Name
John
Mary
Tom

Student Table

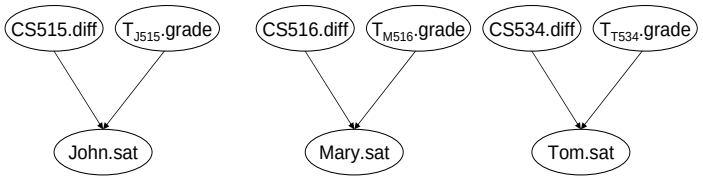
Student	Course
John	CS515
Mary	CS516
Tom	CS534

Takes Table

Course
CS515
CS516
CS534

Course Table

a. A Database for the Student-Course ER Model



b. Ground Bayesian network for the instantiated database

Figure 3: a. A Database corresponding to the student ER model. b. "Unrolled" ground Bayesian Network that is obtained by instantiating the variables of the FOCI statements with the database.

satisfaction node in the Bayesian network will have a *conditional probability distribution*, $P(sat \mid c.\text{diff}, t.\text{grade})$.

2.1 Aggregators

In the previous section, we explained the process of obtaining a ground bayesian network given a FOCI statement and a database. The assumption was that the relationships are one-one i.e., in our example, each student takes one course. But this is not true in real world. Thus if a student takes several courses, we need to describe a generalized conditional probability distribution so that the variable is conditioned on a set of parent variables and the set of parent variables varies from one instance to another. For example, *John* could take four courses while *Mary* could take three courses. In John's case the number of parents is four and Mary's case it is three. In such cases, there are not only a large number of parents but the number of parents vary for different instances. There are two approaches to handling this multiple-parent problem: aggregators and combining rules.

Aggregators have their origin from the database theory [7]. Aggregators are functions that take the *values* of parent variables and combine them to produce a single aggregate value. This value then becomes the parent of the *resultant* variable. In the student satisfaction example, we could say that the average grade that the student obtains in all the courses would influence his satisfac-

tion. This would correspond to a FOCI statement,

`If {Student(s), Course(c), Takes(t,s,c)} then` $Avg([t.score \mid s])$ `Qinf s.satisfaction.` (1)

In the above statement, we are fixing the student and then averaging over the grades of all the courses that the student took. Instead we may need to average over the courses. For example, we may define the *meangrade* of a course as the average of the grades of all the student. The FOCI statement to define this is,

`If {student(s) , course(c) , takes(t,s,c)} then c.meangrade :=` $Avg([t.score \mid c])$

which says that we aggregate over all the grades of a course c. We use the $\mid$ notation as a shorthand for set formers for the ease of specifying the statements. This notation is used to specify that while aggregating the instances of t, we would initially fix the course. Also, this allows us to move all the declarations inside the condition.

Consider the database from figure 3.1. Suppose $John$ takes 4 courses {*CS515, CS516, CS511, CS519*}. The ground bayesian network when the satisfaction FOCI statement with aggregation corresponding to statement 2 is instantiated with $John$ is shown in Figure 4. It can be observed that the grades that $John$ has recieved in these courses are averaged and this is shown as a dotted node in the network. This would correspond to a hidden node in the network. This aggregated node serves as a parent for the *satisfaction* node and each *satisfaction* node will have a conditional probability distribution $P(Sat|AverageGrade)$ associated with it.

2.2 Combining Rules

Another approach to solving the multiple-parent problem is the use of Combining Rules. While aggregators are functions that take values as inputs and output the aggregated values, combining rules take distributions as inputs and output the combined distribution. To understand the use of combining rules in our language, consider the student satisfaction example. Instead of aggregating the grades obtained in the different courses, it may be more natural to determine the satisfaction of a particular course and then combine the satisfactions of all the courses.

More precisely, we could obtain a distribution over satisfaction for a particular course and then combine the distributions of the satisfactions of all the courses. In FOCIL, this could be expressed using a statement like,

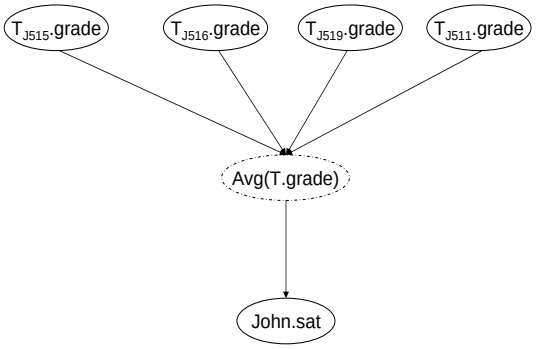


Figure 4: An "unrolled" Bayesian Network with aggregators. The grades in different courses are aggregated using the *Average* aggregator.

```
If {Student(s), Course(c), Takes(t,s,c)} then c.diff, t.grade Qinf
(Mean) s.satisfaction.
```

As with the previous section assume that *John* takes 4 courses and the student is instantiated with *John*. The unrolled network is shown in Figure 5. Each $\langle grade, diff \rangle$ pair would yield a distribution over the satisfaction of *John* and all the distributions are combined using $Mean$ combining rule. The hidden nodes are represented using dotted ovals. To model this using aggregators, we should aggregate the difficulty of the courses separately and aggregate the grades separately and make the aggregated nodes as parents of the satisfaction node. Though this approach would reduce the size of the CPT, the interaction between the difficulty of a course and the grade that was obtained in that course is lost. Combining rules model these interactions more naturally.

In our language, there could be different statements with the same resultant. It could be that the student's satisfaction could also be influenced by his research progress. Now we would end up with two statements with the same resultant i.e., satisfaction. We use combining rules to combine the distributions of these two cases. For example, we could use a *Weighted Mean* combining rule to combine these distributions. In FOCIL, we could write statements such as,

```
WeightedMean{
  If {Student(s), Course(c), Takes(t,s,c)} then c.diff, t.grade Qinf (Mean)
  s.satisfaction.
```

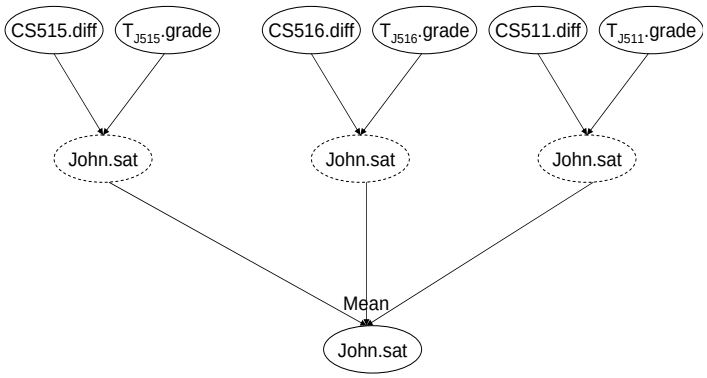


Figure 5: An "unrolled" Bayesian Network with Combining Rules. The different distributions are combined using the mean Combining Rule.

```
  If {Student(s)} then s.progress Qinf s.satisfaction.
}
```

The above example has two FOCI statements with the same resultant *satisfaction* combined using a *Weighted Mean* combining rule. The unrolled network is presented in Figure 6. These two statements are enclosed between { }. The default combining rule is the cross-product combining rule where we do not specify any combining rule but just group the statements together. We force the user to group the statements because it would make the user think about all the qualitative influences for a particular resultant before moving on with the next one. One can imagine writing FOCI statements that has both aggregators and combining rules in the same set of statements. We have implemented the Gradient descent and the EM algorithm for learning the parameters of the combining rules and the parameters of the CPTs [15].

3 Relation to other languages

In this section, we discuss briefly the relationship between a few of the existing probabilistic logic methods and our language. We do not discuss the methodologies in detail but refer the reader to the corresponding works. Probabilistic Relational Models extend Bayesian networks to the relational setting. They are to Bayesian networks as relational logic is to propositional logic [7]. However,

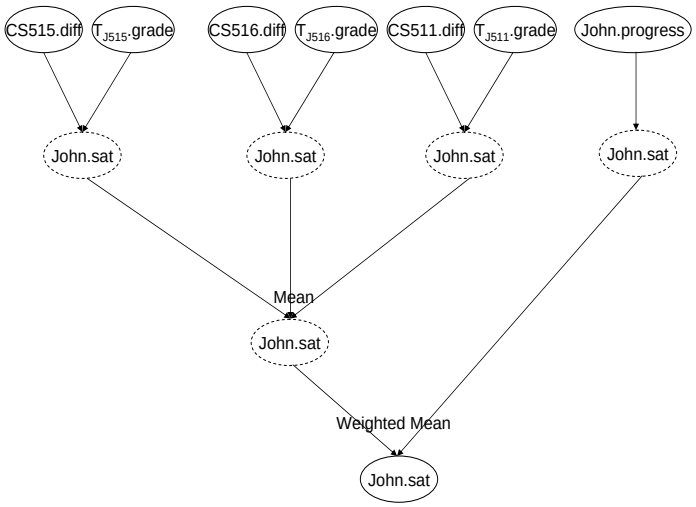


Figure 6: An "unrolled" Bayesian Network with Combining Rules. The different distributions are combined using the mean Combining Rule. The two rules are then combined using the weighted mean combining rule.

PRMs define random variables as slot chains, which can only express joins using binary relations. The conditions in our language enable us to represent relations of arbitrary arity.

Consider the PRM from Figure 7. We could convert this relational schema to an ER diagram. There are two relationships *offers* that exists between a *Professor* and a *Course* and *Registration* that is an M-M relationship that exists between a *Student* and a *Course*. There are six different influence statements. We now provide FOCIL statements corresponding to the different relations. Consider the arc from *Teaching-Ability* of a professor to his *Popularity*. This would correspond to a FOCI statement,

```
If {Professor(p)} then p.teaching-ability Qinf p.popularity.
```

Similarly we could write a statement for the arc between the *Grade* and *Satisfaction* of the registration. Consider the arcs from *difficulty* of a course and *Intelligence* of a student to the *Grade* of the registration. In FOCIL, we would represent this using the statement,

```
If {student(s), course(c), registration(r,s,c)} then s.intelligence,
c.difficulty Qinf r.grade.
```

Consider the relationship between a course *Rating* and the *Satisfaction* of the

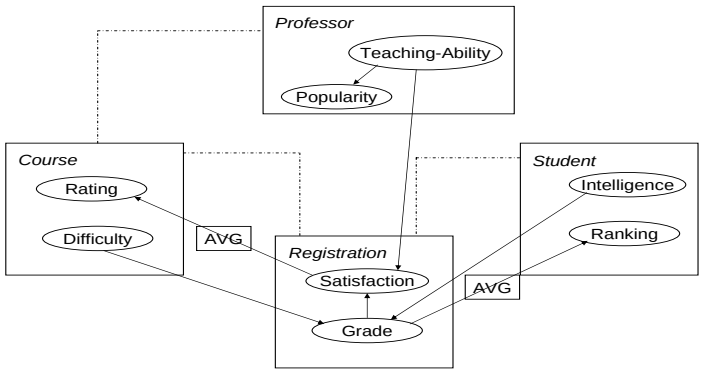


Figure 7: PRM for school domain. This was obtained from [7]

registration. This involves aggregating over all the students who have taken that course. A FOCI statement corresponding to this relationship would be,

`If {student(s) , course(c) , takes(t,s,c)} then` $Avg([t.satisfaction \mid c])$ `Qinf c.rating.`

The other relationship involving aggregators is between the registration's *Grade* and the student's *Ranking*. Here, we have to aggregate over all the courses a particular student took. In FOCIL, this can be represented using the statement,

`If {student(s) , course(c) , takes(t,s,c)} then` $Avg([t.grade \mid s])$ `Qinf s.ranking.`

Finally, consider the arc between a professor's *Teaching-Ability* and the registration's *Satisfaction*. In PRMs, we need to travel through two slot chains. In FOCIL, we need to use the two relationships to represent this influence statement. The statement corresponding to this influence in FOCIL would be,

`If {professor(p) , student(s) , course(c) , takes(t,s,c) , offers(p,c)} then p.teaching-ability Qinf t.satisfaction.`

Heckerman et.al [10] introduced the directed acyclic probabilistic enity-relationship model (DAPER model). An important aspect of the DAPER model is the introduction of the constraints on arcs, which makes them more expressive than slot-chains. However having conditions on individual arcs limits expressivity. As we have mentioned earlier, consider the statement, "the satisfaction of a student depends on the course difficulty and the student's grade in the course". This cannot be easily represented in the current PERM formalism since the two

conditions on the two arcs are not allowed to share free variables. With this restriction, it is not possible to enforce the constraint that it is only the difficulties of the courses the student *takes* that influence his satisfaction. As we have noted earlier, our FOCI statements are equivalent to associating conditions on the hyperarcs in the PERM model as shown in Figure 2.

Kersting and De Raedt introduced Bayesian Logic Programs [12]. This framework is a simplification of that of Ngo and Haddawy [16]. They combine Bayesian Networks with definite clause logic, i.e. pure prolog [12]. Similar to PLPs, BLPs view the atoms as random variables. Bayesian Logic Programs consist of two components: a qualitative component and a quantitative one. The qualitative component is the logical component which captures the structure of the domain. This is similar to that of the Bayesian Network structure. The quantitative component captures the quantitative information about the domain and uses the notion of conditional probability tables(CPTs) and combining rules. The syntax is similar to that of PLPs with the difference being that the quantitative and qualitative part are specified separately. An example of a BLP clause could be,

```
bt(X) | father(F,X) , bt(F), mother(M,X) , bt (M)
```

There would be a CPT corresponding to this clause. In this case, the predicates $mother(M, X)$ and $father(F, X)$ would have boolean values. One could then specify the ground facts like $father(John, Tom)$ etc.

The FOCI statement corresponding to the BLP clause would be,

```
If {Person(p) , Person(m) , Person(f) , mother(m,p) , father(f,p)} then
m.bt, f.bt Qinf p.bt
```

Originally, BLPs do not make a formal distinction between the conditions and the influents, although that they have been extended to do so recently. It can be seen that the Bayesian network constructed from the BLP clause could potentially have more parents compared to the FOCI statements. Hence the size of the conditional probability distribution can be reduced dramatically by separating the conditional predicates from the influence statements. A similar point is made by Fierens et.al [6].

4 Modeling language grammar

Qualitative influences are expressed with statements in the grammar shown in Figure 8. The influents with the same resultant are grouped together using { } and are optionally specified with a combining rule. If the combining rule is not specified, then the default is the cross-product. The statements also have a delimiter for the ease of parsing. Each qualitative influence statement has a label(which will be referenced by synergistic statements), a condition and the qualitative influence. The condition is a conjunction of predicates. The

predicates are used for type setting and for verifying a relation is true. For instance, we could use the predicate *Student(s)* to say that s is a student(which is an entity in our ER model) and *takes(t,s,c)* to say that the relationship *takes* exists between the ground instances s and c. The qualitative influence part has a set of influents and a resultant which are attributes of some objects. If the relationships are **1-M**, either an aggregator or a combining rule must be specified. Note that the FOCI statements could also contain different qualitative influences for different influents. If a statement has multiple influents with different qualitative relationships, they are grouped using {}.

⟨*influences*⟩ ::= ⟨*combrule*⟩ '{'[⟨*statement*⟩ ';']* '}'

⟨*statement*⟩ ::= [⟨*label*⟩ ':' ⟨*relnstmt*⟩] |
'{' ⟨*relnstmt*⟩ [⟨*object*⟩.⟨*att*⟩ ⟨*monoinf*⟩ ⟨*object*⟩.⟨*att*⟩]* '}'

⟨*relnstmt*⟩ ::= 'if' '{' ⟨*condition*⟩ '}' 'then' ⟨*influent*⟩ ⟨*relation*⟩
⟨*combrule*⟩ ⟨*resultant*⟩

⟨*influent*⟩ ::= { [⟨*object*⟩.⟨*att*⟩] | ⟨*aggr*⟩ '([' ⟨*object*⟩.⟨*att*⟩ '|' ⟨*object*⟩
'])'}
{ ',' [⟨*object*⟩.⟨*att*⟩] | ⟨*aggr*⟩ '([' ⟨*object*⟩.⟨*att*⟩ '|' ⟨*object*⟩ '])'}*

⟨*resultant*⟩ ::= ⟨*object*⟩.⟨*att*⟩

⟨*relation*⟩ ::= 'Qinf' | ⟨*moninf*⟩

⟨*moninf*⟩ ::= { 'Q+' | 'Q-' }

⟨*condition*⟩ ::= ⟨*predicate*⟩ [',' ⟨*predicate*⟩]*

⟨*predicate*⟩ ::= A first order predicate

⟨*label*⟩ ::= An optional name which allows the statement to be referenced later on, for example by synergistic statements.

⟨*combrule*⟩ ::= An optional combining rule

⟨*aggr*⟩ ::= An aggregator

Figure 8: Grammar for qualitative influence statements.

5 Future work

We have developed algorithms for learning the parameters of the combining rules along with the parameters of the CPTs [15]. One of our goals is to extend this work to more general classes of combining rules and aggregators including tree-structured CPTs and noisy versions of other symmetric functions. Historically, there has been a lot of work in trying to understand the multiple-parent problem in the propositional setting [17, 20, 9, 2]. In the relational setting, there not been much work in understanding the relationship between aggregators and combining rules. One of our goals is to formalize this relationship and understand them as a first-order extension of the causal independence idea.

We would also like to develop algorithms that can learn in the presence of monotonic relationships. More precisely, we want to extend [1] to FOCIL. Another goal is to learn the parameters in the presence of both combining rules and qualitative constraints.

The other area that we would like to explore is to develop efficient inference algorithms that can exploit the causal independence and the qualitative constraints. One possibility is to derive a support network for the given variable of interest and use a known inference algorithm and answer queries about that variable. The more complex possibility is the development of an efficient first-order inference mechanism in the lines of [19] that would avoid grounding of variables.

References

[1] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of Uncertainty in Artificial Intelligence*, 2005.

[2] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

[3] Pedro Domingos and Mark Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proceedings of of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, 2004.

[4] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: A desktop environment to support multitasking knowledge workers. In *Proceedings of IUI*, 2005.

[5] Ramez A. Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[6] Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical bayesian networks and their relation to other probabilistic logical models. In *Proceedings of ILP*, 2005.

[7] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Invited contribution to the book Relational Data Mining, S. Dzeroski and N. Lavrac, Eds.*, 2001.

[8] David Heckerman. A tutorial on learning with bayesian networks. pages 301–354, 1999.

[9] David Heckerman and John S. Breese. Causal independence for probability assessment and inference using bayesian networks. Technical report, October 1995.

[10] David Heckerman, Christopher Meek, and Daphne Koller. Probabilistic models for relational data. Technical report, March 2004.

[11] Manfred Jaeger. Relational Bayesian networks. In *Proceedings of UAI-97*, 1997.

[12] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 2000.

[13] Brian Milch, Bhaskara Marthi, and Stuart Russell. Blog: Relational modeling with unknown objects. In *Proceedings of of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, 2004.

[14] S.H. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, 2001.

[15] Sriraam Natarajan, Prasad Tadepalli, Eric E. Altendorf, Thomas G. Dietterich, Alan Fern, and Angelo C. Restificar. Learning first-order probabilistic models with combining rules. In *In Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[16] L. Ngo and P. Haddawy. Probabilistic logic programming and Bayesian networks. In *Proceedings ACSC95*, 1995.

[17] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[18] D. Poole. Probabilistic Horn abduction and bayesian networks. *Artificial Intelligence, Volume 64, Numbers 1, pages 81-129*, 1993.

[19] David Poole. First-order probabilistic inference. In *Proceedings of IJCAI*, 2003.

[20] Sampath Srinivas. A generalization of the noisy-or model. In *Proceedings of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI-93)*, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

A Qualitative constraints

In this section, we introduce the qualitative constraints in the language. These constraints operate on the conditional or joint distributions of the variables. The main goal is to reduce the size of the parameter space.

A.1 Monotonicity

Monotonic relation of non-nominal data can be represented by $X\ Q+\ Y$ or $X\ Q-\ Y$. $Q+$ means monotonically increasing and $Q-$ means monotonically decreasing. To understand how a statement $X\ Q+\ Y$ constrain the probability distribution $P(Y|X)$, we employ the notion of *first order stochastic dominance*, which is based on the intuition that increasing values of X shift the probability mass of Y upwards. We now provide the semantics of monotonicity in our language [1].

Definition 1 (First Order Stochastic Dominance) *Given two probability distributions P_1 and P_2, and their respective cumulative distribution functions F_1 and F_2, then*

$$P_1\ FSD\ P_2 \quad \textit{iff} \quad \forall y\ F_1(y) \leq F_2(y) \tag{2}$$

Definition 2 (FSD Monotonicity) *We say that Y is FSD isotonic[4] in X in a context C if,*

$$\forall x_1, x_2,\ x_1 \geq x_2 \Rightarrow P(Y|X = x_1, C)\ FSD\ P(Y|X = x_2, C) \tag{3}$$

Definition 3 (Q+, Q- statements) *Suppose Y has multiple parents $X_1, ...X_q$. The statement $X_i\ Q+\ Y$ means that for all contexts(configurations of other parents) $C \in \times_{j \neq i}$, that Y is isotonic in X_i in context C.*

We now provide an example for monotonic constraint. Our notation is due to [8], in which θ_{ijk} refers to the CPT entry for variable i taking on value k given parent configuration j, that is,

$$\theta_{ijk} = P(x_i = k|pa(X_i) = j)$$

Consider the simplest possible monotonic relation; $X_1\ Q+\ X_0$ for X_0 and X_1 being binary variables. The CPT and single constraint can be found in table 1.

Examples of monotonic relations are:

[4] Antitonic if $x_1 \leq x_2$

	$P(X_0 \mid X_1)$	
x_1	$x_0 = 0$	$x_0 = 1$
0	$\theta_{0;0;0}$	$\theta_{0;0;1}$
1	$\theta_{0;1;0}$	$\theta_{0;1;1}$

$$\theta_{0;0;0} \geq \theta_{0;1;0}$$

Table 1: Example of CPT for a binary variable X_0 given X_1 (also binary) with constraint X_1 Q+ X_0.

- `If {Student(s) , Professor(p) , Course(c) , Teaches(p,c) , Takes(s,c)} then p.interest Q+ s.interest.`

  The student's scholastic interest level is positively influenced by a professor's scholastic interest level if the student takes a course taught by the professor.

- `If {Person(p)} then p.fitness Q- p.weight.`

  A person's fitness level negatively influences the person's weight.

Recall that while writing down different statements with the same resultant, we used {} to group the statements and then apply a combining rule. We use the same idea to represent the cases where there are more than one influent for a resultant and each of them have a different qualitative relationship with the resultant. As an example consider the student satisfaction example. Suppose we want to say that a student's *IQ* and the course *difficulty* would influence the student's *grade* in the course. But we also know that the *IQ* positively influences the *grade* while the *difficulty* negatively influences the *grade*. We could express this in FOCIL as:

```
{
  If {Student(s), Course(c), Takes(t,s,c)} then c.diff, s.iq  Qinf t.grade.
  c.diff Q- t.grade.
  s.iq Q+ t.grade.
}
```

It should also be mentioned that we could use combining rules as well as qualitative constraints in the same statement. Although in this paper we do not learn with these constraints, we have learning algorithms for propositional models with monotonicity constraints [1].

A.2 Synergistic Interactions

When more than one influent monotonically influence a resultant, it is possible to constrain pairs of influents with synergistic statements. We consider three options:

- "Independent": If X $Q+$ Z and Y $Q+$ Z independently then increasing X has the same effect for high values of Y as for low values of Y (and vice versa). For interval-measured variable, this can be referred to as an "additive" synergy, and we have $(X + kY)$ $Q+$ Z for some k.
- "Synergistic": If X $Q+$ Z and Y $Q+$ Z synergistically, then increasing X has a greater effect for high values of Y than low values of Y (and vice versa). For interval-measured variables, we call this "superadditive."
- "Antisynergistic": If X $Q+$ Z and Y $Q+$ Z antisynergistically, then increasing X has a lesser effect for high values of Y than low values of Y (and vice versa). For interval-measured variables, we call this "subadditive."

In cases where no synergy is specified or applicable (such as nominal influents or non-monotonic ordinal influents), we assume the synergistic interaction between the two influents is unknown, and the probability distribution should be unconstrained.

B Contextual Model and Qualitative dependencies

In this section, we provide a contextual model and a few examples of the qualitative dependencies in the model. Also, we present the PERMs corresponding to the dependencies.

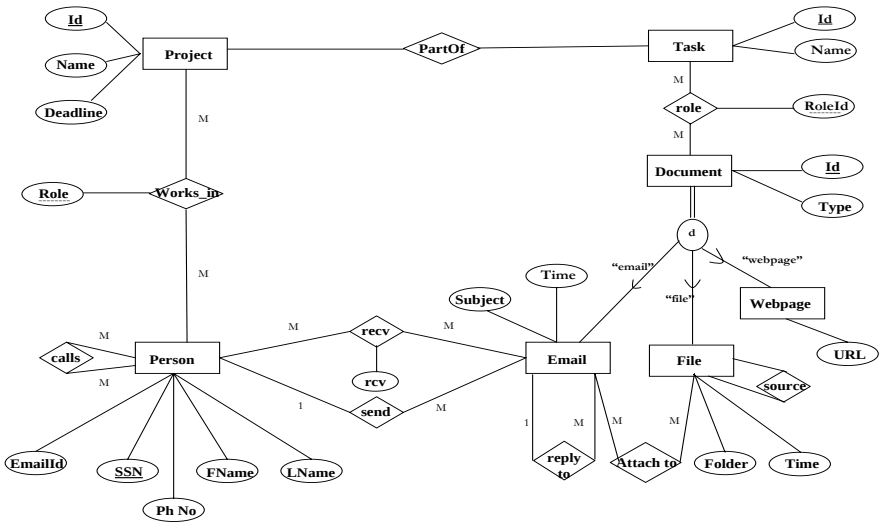
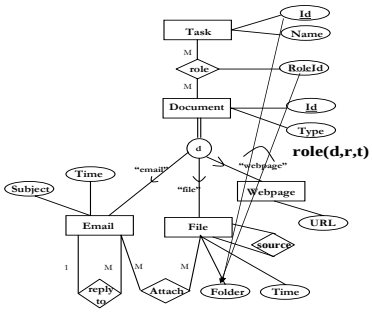
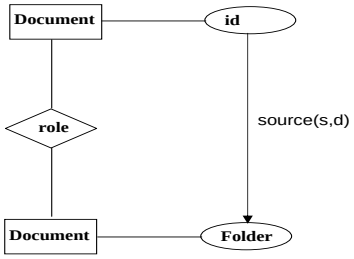


Figure 9: ER Model for Task Tracer [4]

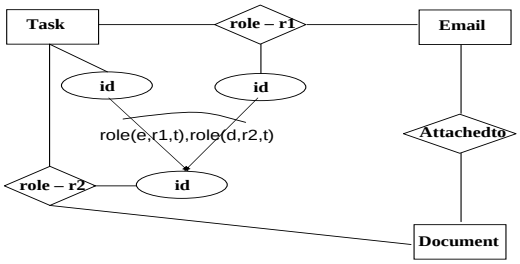
If {task(t),doc(d),role(d,r,t)} then r.id,t.id Qinf d.folder.



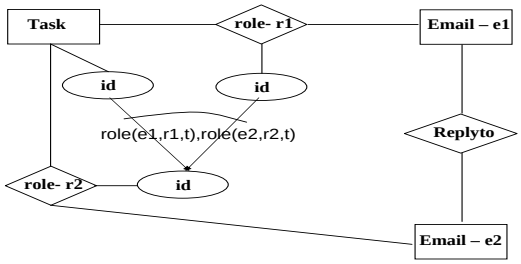
If {doc(d),doc(s)source(s,d)} then s.folder Qinf d.folder.



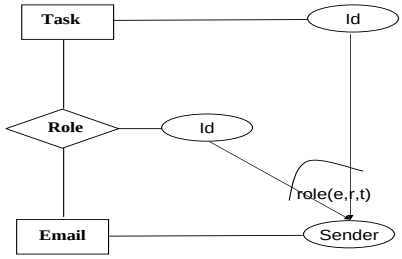
If {email(e),doc(d),attachedto(d,e),task(t),role(e,r1,t), role(d,r2,t) } then t.id,r1.id Qinf r2.id.



If {email(e1),email(e2),replyto(e2,e1),task(t),role(e1,r1,t), role(e2,r2,t) } then t.id,r1.id Qinf r2.id.

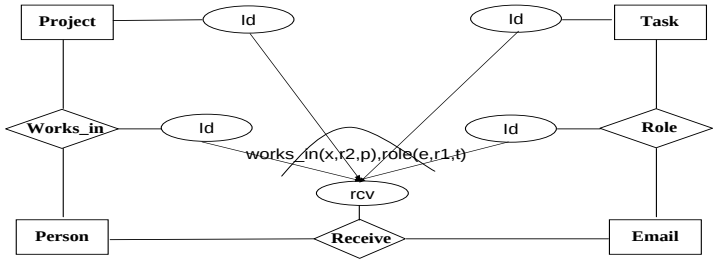


If {email(e),task(t) role(e,r,t)} then t.id,r.id Qinf e.sender.



Note: Here *sender* is a foreign key of the Email table that references the person id

If {email(e),task(t) role(e,r1,t),person(x),project(p), works_in(x,r2,p),receive(r,x,e)} then t.id,r1.id,p.id,r2.id Qinf r.rcv.



Note: We introduce a new variable r here to denote the instantiation of a M-M relationship

If {person(x),person(y),project(p), works_in(x,r1,p), works_in(y,r2,p),calls(c,x,y) } then p.id,r1.id,r2.id Qinf c.call.