

Qualitative Probabilistic Matching with Hierarchical Descriptions

Clinton Smyth

GeoReference Online Ltd.,
Vancouver, B.C., Canada
cpsmyth@georeferenceonline.com
<http://www.georeferenceonline.com/>

David Poole

Department of Computer Science
University of British Columbia
poole@cs.ubc.ca
<http://www.cs.ubc.ca/spider/poole/>

Abstract

This paper is about decision making based on real-world descriptions of a domain. There are many domains where different people have described various parts of the world at different levels of abstraction (using more general or less general terms) and at different levels of detail (where objects may or may not be described in terms of their parts) and where models are also described at different levels of abstraction and detail. However, to make decisions we need to be able to reason about what models match particular instances. This paper describes the issues involved in such matching. This is the basis of a number of fielded systems that do qualitative-probabilistic matching of models and instances for real-world data.

Keywords: hierarchical matching, ontologies, qualitative probability, applications

Introduction

The classic problem of exploration may be “Is there gold in them hills?”. In order to determine whether gold might be in an area, we need to match a description of the area with a model of areas that contain gold. While this may seem straightforward, it is complicated because different parts of the world have been described at different levels of abstraction (some use more general terms than others) and at different levels of detail (some may be divided into parts and some may be described holistically). Similarly, models of where gold can be found are described at various levels of abstraction and detail. While it may seem that we need to use probability and utility to make decisions about where to explore, practitioners are wary of using probabilistic assessments where there is little data to support their values and seem to be happier with qualitative assessments of uncertainty. Moreover, as it is the person, not the computer, who is responsible for making the actual decisions, they are much happier with rankings of matches where the system can explain its valuation. This paper describes the knowledge representation issues and their resolution for a family of applications in geology (MineMatch and PlutonMatch), geographic information systems (LegendBurster) and environmental modelling (HazardMatch).

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

In the domain of geology, some parts of the world are described quite specifically at a huge amount of detail (e.g., where there are mines) and some at a very coarse level (e.g., parts of northern Canada where no one may have actually visited). Similarly, there are some models that people spend careers working on and are described quite specifically in great detail¹ and some that may just be very high-level models put together quickly to cover the cases. Other real-world domains where the models and the instances are built by multiple people will have similar challenges. The running example we will use in this paper is one where we can describe housing units (apartments and/or houses) and associated models.

In order to consider matching, we need an ontology that enables comparison at different levels of abstraction and lets us compare even when different vocabularies are used, a representation for models, and instances and a way to match instances and models.

Describing instances and models

We want to be able to describe models and instances at various levels of abstraction and various levels of detail, and we still want to be able to compare them in a meaningful way.

We assume that we have an ontology that includes taxonomic hierarchies that specify the vocabulary for different levels of abstraction. This forms what is called an “is-a” hierarchy, a “kind-of” hierarchy, or a value hierarchy.

Figure 1 shows two examples of value hierarchies. A *bathroom* is kind of a *room*. A *bedroom* is kind of a *room*. A *kids bedroom* is a kind of *bedroom*. A *master bedroom* is a kind of *bedroom*. We assume that the sibling classes are exclusive (e.g., a *bathroom* is not a *bedroom*). In some domain we may only know that something is a *bedroom*, not whether it is a *master bedroom* or not. We know that it is a *room* and not a *bathroom*.

Instances and models can also be described either holistically or in terms of their parts (i.e., as a *partonomy*), and these parts can in turn be described in terms of their subparts. In general, we assume that objects and parts are described by their properties and their relationships.

For example, a particular apartment may be described as containing a painted bedroom with two beds and a bathroom

¹e.g., see <http://minerals.cr.usgs.gov/team/depmod.html>

Room	
	Bathroom
	Bedroom
	Kids bedroom
	Master bedroom
	Spare bedroom
	Dining room
	Living room
	Family room
	Sitting room
	TV room

WallStyle	
	Painted
	Mottled
	Striped
	Uniformly Painted
	WallPaper
	Wood Panelled
	Oak Panelled
	Pine Panelled

Figure 1: Two partial taxonomies for the housing domain. Indentation shows the subsumption.

with a green bath. We may also say that the apartment does not contain another bedroom. Note that this description is ambiguous as to whether it also has a hall, the sizes of the beds and even if there is a third bed in the room.

We also describe models in terms of their components, but as well as describing the presence and absence of components, we also describe the qualitative probability of the presence or absence of these components, as well as the qualitative probability of their properties. For example, we could say that an apartment that meets Max Jones’s needs² will usually have a king-size bed, often a room that isn’t a bathroom or a bedroom and rarely a green bath.

In this paper, we do not assume there is a unique value for each attribute. For example, if a room has a particular wall style (e.g., that it is painted with stripes), we are not excluding the possibility that it might also have a different wall style (it may also be wood-panelled). That is, we are not assuming that the attributes are functional. While we may need to allow for functional attributes in real applications, including this possibility would only make this paper more complicated because we would then need to consider both cases of functional and non-functional attributes. Similarly, this paper does not consider statements that there are no more values for an attribute (e.g., a room might be mottled and wood panelled, but doesn’t have other values for the wall style attribute).

²We are assuming here that we are describing probabilities, not preferences or utilities. This statement is not representing Max’s preferences, but is modelling whether we would predict that Max likes such an apartment.

Qualitative Probabilities

The systems we are building are meant to support decisions, which depend on probabilities and utilities. They are meant to support a myriad of users each with their own subjective probabilities and utilities. The role of the system is to provide a similarity ranking so that instead of having thousands of models or instances to evaluate, the user is presented with an ordered list of better to worse matches from which they can use their own subjective probabilities and utilities. This is necessarily an approximation. It is more important to have a good explanation facility so that the user can see why some matches were made or not. Note that by probability we mean a subjective belief and not a statistical proportion.

We adopt qualitative probabilities for the following reasons:

- The probabilities are not available, and the users are reluctant to provide numbers.
- The domains we are interested in are ones where there is a community of users, each with their own subjective priors and utilities. The users are suspicious of an exact probability when it doesn’t reflect their prior beliefs.
- In these domains people make decisions and are held accountable for their decisions. Computers can help them make better decisions by reducing the cognitive overload by reducing the number of possibilities the users need to consider. To this end, we want to provide an order of matches; from better to worse so that the user can explore these and apply their own subjective priors and utilities to make better decisions.
- Good explanations are more important than accuracy. The system needs to be able to justify its valuations so that the user can justify and explain their decisions.
- Many extra unmodelled factors come into making a decision, and these factors need to be taken account of by the decision maker. The decision makers need to understand the rationale for the system’s output. The system can’t provide a single inscrutable recommendation.
- We want the output to be consistent with a full probabilistic model. We acknowledge that the right decision may be based on probabilities and utilities; we would hope that this system can integrate with a full probabilistic reasoning system or even migrate to be such a system. In the analysis below, sometimes we discuss what the correct probabilistic answer is, and then show how the qualitative version can be derived from this.
- The full probabilistic computation is often too complex, both for the computer and to explain to a human. There are cases below where the full probability distribution is very complicated, but the qualitative version provides an approximation to the modes of the distribution and is much more tractable.

We adopt a simple form of qualitative probabilities based on the kappa calculus (Spohn, 1988; Pearl, 1989; Darwiche and Goldszmidt, 1994). The kappa calculus measures uncertainty in degrees of surprise. The best way to see the kappa calculus is as an order-of magnitude approximation to

log probabilities. Where probabilities multiply, the kappa-values add, and where probabilities add, the kappa calculus takes maximums (similar to possibility logic and fuzzy logic (Dubois and Prade, 1991; Dubois, Lang and Prade, 1994).

Because we are working with real applications, and not just in the abstract, we need to commit to actual values for the kappa values. After consultation of domain experts we have used a 5-value scale for inputting uncertainty values. We use the following qualitative probabilities to describe the models:

always a proposition is “always” true, means that you are very surprised if it is false. All experience leads you to believe that it is always true, but you are leaving open the possibility that it isn’t true³.

usually a proposition is “usually” true means you are somewhat surprised if it is false.

sometimes a proposition that is “sometimes” true, means you wouldn’t be surprised if it is true or false.

rarely a proposition is “rarely” true means you are somewhat surprised if it is true.

never a proposition is “never” true, means that you are very surprised if it is true.

In terms of the kappa calculus, we choose numbers $\alpha > 0$ and $\beta > 0$ so that:

- always p means $\kappa(\neg p) = \alpha$ and $\kappa(p) = 0$. Thus α is the measure of surprise that p is false.
- usually p means $\kappa(\neg p) = \beta$ and $\kappa(p) = 0$. Thus β is the measure of surprise that p is false. The relative surprises means that $\beta < \alpha$.
- sometimes p means $\kappa(\neg p) = 0$ and $\kappa(p) = 0$. We are not surprised if p is true or not.
- rarely p means $\kappa(\neg p) = 0$ and $\kappa(p) = \beta$. Thus β is the measure of surprise that p is true. Note that “rarely” is the dual of “usually.”
- never p means $\kappa(\neg p) = 0$ and $\kappa(p) = \alpha$. Thus α is the measure of surprise that p is true. Note that “never” is the dual of “always.”

Note that these qualitative uncertainties are only the input values (i.e., as part of the models); on output we give a numerical score (both a raw score as well as a percent of the best match). This finite scale is not adequate to describe the level of matches. In our applications we have used $\beta = 1000$ as the value for being somewhat surprised and $\alpha = 10000$ as the value for being very surprised. (The only significant feature of the values is the 10-fold ratios between them; 10 “somewhat surprised”s is equal to one “very surprised”).

The scale, and even the zero point, are arbitrary if all we really need is a way to compare matches. We take a positive

³It must be emphasised that we are *not* using “always” and “never” in the sense of modal logic, where it would mean true in all possible worlds, and so would exclude a match if the proposition is false. Here we are including the possibilities of mistakes, and the possibility that there is no model that actually matches but we still want best matches. This follows the philosophy of the kappa calculus (Spohn, 1988) where there is no limit to our surprise. This is analogous to not allowing a probability of 0 for any contingently possible proposition.

attitude; surprises will give negative scores, but we also allow rewards that give positive scores. The best match will be the one with the maximum score.

We only allow unconditionally independent hypotheses. In much the same way as Poole (1993), we model dependence by allowing the modeller to invent hypotheses to explain dependencies amongst other variables.

The kappa calculus is the coarsest level of uncertainty we use. There are many cases where the kappa calculus gives the same value for very different probabilities (Darwiche and Goldszmidt, 1994). As well as the kappa calculus we use much lower numerical values for these more subtle distinctions (these numbers are swamped by the numbers provided by the kappa calculus). In general we use a positive number if the probability would be increased by a match and a lower number if the probability would be decreased. For example, if the model makes a prediction that is confirmed in an observation, this results in a positive reward. This model is more likely than a model that is otherwise identical but makes no prediction. Thus we reward the model that makes the prediction over the model that makes no prediction. As the zero is arbitrary we choose the zero to be the reward obtained by a empty model for an empty instance.

Abstraction hierarchies

As part of real-world domains, we assume that the models and the instances can be described at various levels of abstraction. We assume that we have taxonomies that define the hierarchical relationships between the concepts. We are adopting standard taxonomies, for example the British Geological Survey Rock Classification Scheme⁴ describing more than 1000 rocks and the Micronex taxonomy of minerals⁵ with more than 4000 minerals described.

There is a difference between instances and models in what describing something at different levels of abstraction means. For example, consider the taxonomies of Figure 1. If we say that a particular apartment contains a painted room (as opposed to a mottled-painted bathroom), this means that we don’t know what sort of a room it is or which style of painting it is. In a model, if we say that we want an apartment that contains a painted room, it means that we don’t care what sort of room it is, or what style of painting it is. This distinction is important for matching.

Multiple Levels of Detail

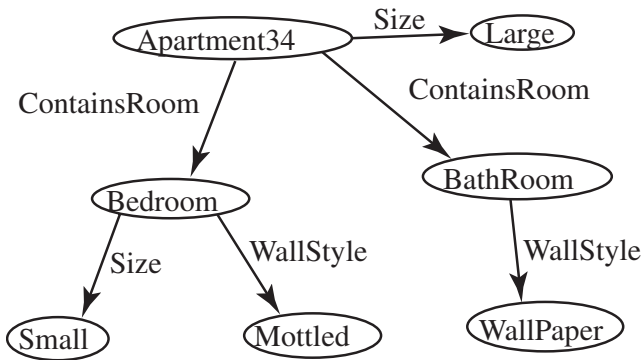
As well as describing instances and models at multiple levels of abstraction, we can also describe them at different levels of detail. We may describe an instance in terms of its parts and subparts⁶ or we may describe it holistically.

One of the challenges is that the same instance may be described at different levels of detail and abstraction. For example, an apartment may be described in one situation as being a “two bedroom apartment” and in another situation it may be described in much more detail, that specifies what

⁴<http://www.bgs.ac.uk/bgsrsc/>

⁵<http://micronex.golinfo.com/>

⁶We use the term *subpart* for a part or a subpart of a part (i.e., any descendent in the partonomy).



Apartment34's Attribute - Value	Present or Absent
ContainsRoom - Bedroom	present
Size - Small	present
WallStyle - Mottled	present
ContainsRoom - BathRoom	present
WallStyle - WallPaper	absent
Size - Large	present

Figure 2: A graphical description of Apartment-34 and the corresponding tabular description

rooms it contains and doesn't contain as well as what the rooms contain. If we have particular requirements, we may be unsure whether the apartment fits the description given the first brief description, but we may be more sure that it fits or doesn't fit given the more detailed description. It is important to note that whether the apartment actually suits our needs is a property of the apartment, not of its description. Thus when matching, we try to define the qualitative probability of the instance, not the qualitative probability of the description.

Putting it together

In describing instances, we need to choose a level of detail and a level of abstraction and describe the whole and its subparts. Note that different subparts can be described at different levels of detail and abstraction.

For example, we could describe a particular large apartment as containing a small mottled bedroom and bathroom that isn't wallpapered. We can write this hierarchically as in Figure 2. As part of the ontology, we have top-level objects (in this case apartments) that are named (e.g., "Apartment34"). We have other entities, which are not named, but are labelled with their types (e.g., "Bedroom" or "Bathroom") and other attributes where we just give the values (e.g., "Large", "Wallpaper", etc.).

We can describe a model using qualitative probabilities to describe how likely the model will contain certain values for attributes. For example, an agent may come up with a description of what they think Joe would like, forming a "apartment model 22" (see Figure 3). In this model, Joe always wants a master bedroom (that is usually painted, rarely wallpapered and never wood panelled), usually wants a spare bedroom that has painted walls, always wants a bathroom (that usually doesn't have wallpaper), rarely wants a hall and rarely wants to be on level 1.

ApartmentModel22's Attribute - Value	Expected Frequency
ContainsRoom - MasterBedroom	always
WallStyle - WoodPanelled	never
WallStyle - WallPaper	rarely
WallStyle - Painted	usually
ContainsRoom - SpareBedroom	usually
WallStyle - Painted	usually
ContainsRoom - BathRoom	always
WallStyle - WallPaper	rarely
ContainsRoom - Hall	rarely
OnLevel - 1	rarely

Figure 3: A hierarchical description of apartment model 22

Matching

The aim of the matcher is, given a database of instances, and given a model, to order the instances by their qualitative probability given the model. Thus, we want the qualitative analogue of $P(instance|model)$; if we wanted the probability of the model given the instance, we would multiply this by the probability of the model (which for a single model and multiple instances, just multiplies everything by a constant) and normalise. As the instance is a conjunction of features, we multiply the probabilities of the individual features. In the kappa calculus, we are working in the log-probability space, so we add the qualitative probabilities. Note that this assumes independence of the features given the model. This is not a real restriction as we can invent causal hypotheses (new attributes) to explain any dependence (Poole, 1993).

Matching at different levels of abstraction

If not for the hierarchical descriptions, we add the values of the "surprises" to get the qualitative probability of the instance given the models (Spohn, 1988; Pearl, 1989).

There are a number of problems with the interaction of the hierarchical descriptions and the qualitative probabilities:

- A more specific description is always less likely than a more general description. It is always less likely to have a bathroom than a room; or, in terms of surprises, we are always more surprised to have a bathroom than a room.
- A more detailed description is also less likely (or more surprising) than a less detailed description.

In general, we want to consider not the probability of a description, but the expected probability of a fully specified instance of that description.

For example, suppose we have as instance descriptions:

- i_1 : something that is painted
- i_2 : a painted room,
- i_3 : a painted bathroom,
- i_4 : a mottled bathroom

i_1 , as a description, is more likely than i_2 , which is more likely than i_3 , which is more likely than i_4 , independently of the model being matched, because everything that fits the description i_2 also fits the description i_1 , everything that fits the description i_3 also fits the description i_2 , and everything that fits the description i_4 also fits the description i_3 (mottled is a type of painted; see Figure 1).

ARoomInstance’s Attribute — Value	Present or Absent
CupboardStyle — PinePanelled	present
DividerStyle — Painted	absent
EastWallStyle — Painted	present
NorthWallStyle — Painted	present
SouthWallStyle — PinePanelled	present
WestWallStyle — OakPanelled	present

Figure 4: A description of a room

ARoomModel’s Attribute — Value	Expected Frequency
CeilingStyle — Painted	always
DividerStyle — Painted	usually
EastWallStyle — Painted	usually
NorthWallStyle — Mottled	usually
SouthWallStyle — WoodPanelled	always
WestWallStyle — Mottled	usually

Figure 5: A description of a room model

However, consider the model that says we always want a *painted bathroom*. It is less likely that the actual room that i_1 describes fits our model than the actual room that i_2 describes. Thus i_2 should be a better match than i_1 . Similarly i_3 is better than i_2 . i_4 is the same goodness as the match i_3 ; they are both perfect matches as they are both painted bathrooms.

For the probabilistic case, if the instance was described just as a painted room, the probability that it is a painted bathroom is the proportion of painted rooms that are bathrooms. Thus, when the instance is described at a higher level of abstraction than the model, the probability needs to be multiplied by the probabilities down the tree. When we are using qualitative probabilities, we need to add surprise values indicating how surprised we are that a room is a bathroom.

As our numbers are purely for comparing instances, the scale and the zero are arbitrary. It is more natural to provide a reward for a match than a penalty for a non-match. Similarly, if we have a more general description of the instance than the model (as in i_2), we provide a lower penalty, or if we are adding rewards, a lower reward.

To see how the matching works with multiple levels of abstraction, consider the room description in Figure 4 and the room model in Figure 5. These use the WallStyle hierarchy of Figure 1. Note that this example is designed to show the different cases of missing and present attributes and the relative positions in the abstraction hierarchy. Figure 6 shows the details of the match.

In the first line, the ceiling style of the model has no corresponding instance value. We do not know the ceiling style of the room. This “unmatched” is not as good a match as if we knew the ceiling style of the instance was *painted*, but is better than if it was not painted.

In the second line, the cupboard style of the instance has no corresponding model value. The model does not care what the style of any divider is.

In the third line, the model specifies a painted divider is

usually present, but this is explicitly absent in the instance. There is a penalty for this surprise.

In the fourth line, the East wall styles match exactly. There is a reward for this match, as it makes the room more likely that it fits the model. We need such a reward as we want models that specify values lower in the abstraction hierarchy (i.e., that are more specific) to be rewarded when they find a match. Otherwise there is no way to distinguish the more general models from the more specific one.

In the fifth line there is a match but not an exact match. The model wants a mottled wall, but the instance is just painted; we do not know whether the instance is mottled or not. We have more of a reward than if we knew nothing about the instance, as our belief that it is mottled has gone up as we have ruled out some other alternatives.

In the sixth line, there is also an exact match. The model requires a wood panelled South wall, and the instance has wood panelled South wall (as pine panelling is a sort of wood panelling). This gives the same match as if the instance was just wood panelling; we don’t penalise an instance for being more specific than it needs to be.

The last two lines illustrate a more subtle point. If the WestWallStyle attribute was functional (it could only have one value), there would be a mismatch and we would have a penalty (the β of the qualitative probabilities). We however assume that it is possible that the West wall can have multiple styles, and that we are not assuming they are all given. In this case, we don’t know if the West wall is also mottled.

Consider the analogy of being asked to get a big red ball, but you can only find a big green ball (or even a small green ball); you can either say you can’t find a big red ball, or that you found the ball but it is the wrong colour (both of which are true). You should really specify the best of these: If it is worse to have the wrong colour than to not find a ball, you should say you couldn’t find a big red ball. If it is worse to not find the ball than to have the wrong colour, then you should say you found the ball, but it is the wrong colour.

If we have a choice as to having a match or not, we choose whichever of these has the best score. If the match has the best score, we have a match, otherwise we don’t have a match. Recall that the zero-point in our scale was arbitrary; it was chosen to be the value of the null match with the null match. Given this, we choose rewards and penalties so that the above choice makes sense.

In summary, in determining a match, we sum over the matches of the individual components. When there is a choice between a match or no match, we choose whichever one has the highest score.

Matching at different levels of detail

Instances and models can also be described at various levels of detail. In this section we discuss how some properties can inherit upwards and some inherit downwards, and how when properties are described at different levels of details can be matched. See (Atrale, Franconi, Guarino and Pazzi, 1996) for a more detailed discussion of partonomies and the part-of relationship.

Universal properties that are properties of all the subparts are “downward inherited”. For example, consider the prop-

ARoomInstance: Attribute	ARoomInstance's Value	ARoomInstance: Present or Absent	ARoomModel: Attribute	ARoomModel's Value	ARoomModel: Expected Frequency	Match Type
			CeilingStyle	Painted	always	unmatched
CupboardStyle	PinePanelled	present				
DividerStyle	Painted	absent	DividerStyle	Painted	usually	freq
EastWallStyle	Painted	present	EastWallStyle	Painted	usually	exact
NorthWallStyle	Painted	present	NorthWallStyle	Mottled	usually	maybeAKO
SouthWallStyle	PinePanelled	present	SouthWallStyle	WoodPanelled	always	exactAKO
			WestWallStyle	Mottled	usually	unmatched
WestWallStyle	OakPanelled	present				

Figure 6: A depiction of the match of the room model of Figure 5 and the room instance of Figure 4

erty “all painted”. If a house is all painted, and the house contains a suite, then the suite is all painted, and if the suite contains a room then the room is painted. The considerations of reasoning at different levels of abstraction apply to this case.

Existential properties, that refer to the existence of some component are “upward inherited”. For example, consider the property “contains a bath”. If a suite is part of a house, and a room is part of the suite, and the room contains a bath, then the suite contains a bath and the house contains a bath. In this case the inheritance is in the opposite direction to the abstraction hierarchy.

These two interact. For example, if at some level, we know that we have a (non-empty) part that is all painted, then for all subparts, all painted is true, and for all superparts “contains some painted parts” is true. They are also the dual of each other; the negation of a universal property is an existential property, and the negation of an existential property is a universal property.

When comparing models that may contain multiple parts with instances that may contain multiple parts, we have to identify which parts of the models correspond to which parts of the instances, which parts of the models do not have correspondences in the instances, and which instance parts do not have corresponding model parts.

We make the assumption that in a single description, different descriptions of parts pertain to different parts. For example, a description of an instance that contains a mottled room and a painted bathroom actually contains (at least) two rooms, as opposed to containing one room that is a mottled bathroom.

The simplest case with multiple levels of detail is where one of the the model or the instance is described with parts and the other isn't. In this case, we have some similar issues to the multiple levels of abstraction and some new issues related to the inheritance of properties.

If the model does not specify parts, then it does not care which part some existential property is true in. For example, suppose the model specifies that the apartment contains a bath, but does not describe the apartment in terms of its subparts (i.e., its rooms). This should also match an instance

that is described in terms of rooms as long as one one of the rooms contains a bath. In this case, we inherit upwards those properties of the instance parts that can be inherited upwards (such as contains a bath), and do standard matching.

If the model specifies a universal property, then the property must be true in all subparts for a match. If the instance has a subpart for which the property does not hold, we have a conflict as the property isn't true of the instance, and so must get the appropriate penalty or reward. Otherwise, there could be a partial match, as the property could be true. Allowing statements that the parts listed are all of the parts (e.g., if we are told that silence implies absence) for the instance could let us conclude the universal property is true of the instance.

If the instance does not specify parts, but the model does, we could have a partial match for existential properties if the property is true for an instance part. For example, suppose the instance specifies the apartment contains a bath, but the model specifies that the apartment contains a bathroom that contains a bath. In this case we have a partial match, as there is some evidence that the instance has a bath in the bathroom (we know it has a bath somewhere).

If the instance has a universal property, then there should be a perfect match with a model that has this property for a subpart. For example, if the instance apartment is all-painted, the it should match with a model that specifies the bathroom is painted (or all-painted).

This sort of reasoning carries over even if the model (or instance) does specify parts. If it has a property described outside of any of its parts, the same considerations need to be considered to determine if it matches with any subpart of the instance (or model) that mentions that property.

The next simplest case is when the model that contains a single part of a certain type and the instance contains multiple corresponding parts. To do a proper probabilistic match, we would have a probability distribution over the instance parts together with no-part. This would specify the probability that each instance part corresponds to the model part and the probability that there was no instance part that corresponded to the model part.

For example, suppose that the model specifies that the person always (or never) likes a house that contains a mottled

bathroom, and the instance has as parts a mottled room and a painted bathroom. To find how good a match this is, we need the probability of each of the four hypotheses:

- the mottled room is a bathroom,
- the painted bathroom is mottled,
- there is another room that is a mottled bathroom
- there is no mottled bathroom

From these probabilities we can compute the probability that the house contains a mottled bathroom.

If we had multiple parts in the model and multiple parts in the instance being matched, we would need a probability distribution over the possible assignments of model parts and instance parts. This is too difficult to compute, let alone explain to a user.

Because we are working with qualitative probabilities, we don't need this complexity. The correspondence to finding the probability of the distribution is finding the best match (as sum in probability corresponds to max in the kappa calculus). We need to find the best match between the model parts and the instance parts. However, even this is computationally intractable. Instead, we do a myopic search finding the best match between model parts and instance parts, remove this pair and continue till we have run out of model parts or instance parts.

Thus, in order to determine the match for the mottled bathroom, we find the best match amongst the 4 hypotheses above, and use the corresponding qualitative probability. The above analysis holds whether the model specifies it always wants a mottled bathroom or never wants a mottled bathroom. Thus once we have found the best match, it incurs a reward or a penalty.

As a detailed example, consider the match depicted in Figure 7 between apartment 34 (Figure 2) and model 22 (Figure 3). This is the match output from our program that is presented to the user (but with colour).

In this example, the instance has a bedroom, but the model contains two bedrooms (a master bedroom and a spare bedroom). The matcher considers the hypotheses:

- The instance bedroom corresponds to the master bedroom of the model.
- The instance bedroom corresponds to the spare bedroom of the model.
- The instance bedroom does not correspond to either bedroom of the model.

For each of these it computes the score, using the methods above. For the first two hypotheses, it has a partial match on the bedroom (as it does not know which type of bedroom the instance is), but for each of them it finds the painted wall style. Because the master bedroom is always present (the spare bedroom is only usually present), it is the best match, and is matched and reported.

Ontology Language

The representation we use for ontologies is as close to OWL (Patel-Schneider, Hayes and Horrocks, 2003) as we can get.

Unfortunately, OWL is not adequate for our needs. For the instances we want to be able to say whether it is true or false that some attribute has some value. For the models, we want to be able to give a qualitative probability that some object has some value for some attribute. We can see either the truth values or the qualitative probabilities as frequency values.

While we could reify the object-attribute-value triples, this makes the knowledge base much more obscure than it needs to be. We extend the ontology representation to have object-attribute-value-frequency-citation tuples, where we specify the frequency (truth value for attributes and qualitative probability for models) and the citation specifies where the information came from.

Conclusions

There has been a limited amount of work on combining probabilistic reasoning and taxonomic hierarchies. Pearl (1988, Section 7.1) gives a way to allow probabilistic reasoning with taxonomic hierarchies. The idea is to allow for a strict taxonomy, but with a probability distribution at the leaves; he showed how to incorporate such knowledge into Bayesian network inference. Koller, Levy and Pfeffer (1997) allow for similar reasoning in a richer logic. Koller and Pfeffer (1998), allow for conditional probabilities for attribute values at various levels of abstraction. None of these allow for the hypotheses to be hierarchically described. None of them give the probability of the instance; they all give the probability of a description; so more general descriptions are also more likely.

In summary, this paper presents a framework for matching in areas where a number of different people have developed models (perhaps over many years), and there are many instances described, and all of these are at different levels of abstraction and detail. This seems to be the sort of knowledge that many scientific disciplines have accrued. This research has arisen from practical issues in geology.

This is not an idle academic exercise. There are currently many geological surveys of provinces and countries that are inputting descriptions of all of their mineral occurrences into the system. We currently have thousands of instances represented; many at a high level of abstraction and with little detail, but a number of quite detailed descriptions of instances of mineral deposits and mines, for example the European mineral deposits published by GEODE⁷. We have many models described including those of the US Geological Survey⁸ and the British Columbia Geological Survey⁹.

This paper should be seen as a general overview of work in progress. Trying to find what experts consider to be good matches is a non-trivial task, but the work reported here has been found to provide significant advantages over methods currently in use.

⁷<http://www.gl.rhbc.ac.uk/geode/index.html>

⁸<http://minerals.cr.usgs.gov/team/depmod.html>

⁹<http://www.em.gov.bc.ca/Mining/Geolsurv/>

ApartmentModel22: Attribute	ApartmentModel22's Value	ApartmentModel22: Expected Frequency	Apartment34: Attribute	Apartment34's Value	Apartment34: Present or Absent	Match Type
ContainsRoom	SpareBedroom	usually				
WallStyle	Painted	usually				
ContainsRoom	BathRoom	always	ContainsRoom	BathRoom	present	exact
WallStyle	WallPaper	rarely				
ContainsRoom	Hall	rarely				
ContainsRoom	MasterBedroom	always	ContainsRoom	Bedroom	present	maybeAKO
			Size	Small	present	
WallStyle	WallPaper	rarely				
WallStyle	Painted	usually	WallStyle	Mottled	present	exactAKO
WallStyle	WoodPanelled	never				
OnLevel	1	rarely				
			Size	Large	present	

Figure 7: A depiction of the match of apartment 34 with model 22

References

- Atrale, A., Franconi, E., Guarino, N. and Pazzi, L. (1996). Part-whole relationships in object-centered systems: an overview, *Data and Knowledge Engineering* **20**: 347–383.
- Darwiche, A. and Goldszmidt, M. (1994). On the relation between kappa calculus and probabilistic reasoning, *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 145–153.
- Dubois, D., Lang, J. and Prade, H. (1994). Possibilistic logic, in D. Gabbay, C. J. Hogger and J. A. Robinson (eds), *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, Oxford University Press, Oxford, pp. 439–513.
URL: citeseer.nj.nec.com/dubois92possibilistic.html
- Dubois, D. and Prade, H. (1991). Epistemic entrenchment and possibilistic logic, *Artificial Intelligence* **50**(2): 223–239.
- Koller, D., Levy, A. and Pfeffer, A. (1997). P-classic: A tractable probabilistic description logic, *Proc. 14th National Conference on Artificial Intelligence*, Providence, RI, pp. 390–397.
- Koller, D. and Pfeffer, A. (1998). Probabilistic frame-based systems, *Proc. 15th National Conference on Artificial Intelligence*, AAAI Press, Madison, Wisconsin.
- Patel-Schneider, P. F., Hayes, P. and Horrocks, I. (2003). Owl web ontology language semantics and abstract syntax, *Candidate recommendation, W3C*.
URL: <http://www.w3.org/TR/owl-semantics/>
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA.
- Pearl, J. (1989). Probabilistic semantics for nonmonotonic reasoning: A survey, in R. J. Brachman, H. J. Levesque and R. Reiter (eds), *Proc. First International Conf. on Principles of Knowledge Representation and Reasoning*, Toronto, pp. 505–516.
- Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks, *Artificial Intelligence* **64**(1): 81–129.
- Spohn, W. (1988). A general non-probabilistic theory of inductive reasoning, *Proc. Fourth Workshop on Uncertainty in Artificial Intelligence*, pp. 315–322.