Milestone M4.34

Prototype for taxonomic generalization

Methodology specification for generalization of descriptive data (e.g. from specimens to taxon) and provide a prototype implementation in Xper^2

Leading partner: UPMC

Compiled by: RégineVignes-Lebbe, Thomas Burguière, Nils Paulhe

Date: July 31, 2012

The generalization of descriptive data

Descriptions of specimens and taxa have to be consistently related to the taxonomic classification. If all taxa belong to the same taxonomic rank, they express a partition (a set of non overlapping groups) of some living organisms. The checkbase function of the Xper2 software is perfectly suitable to check the consistency and overlapping or nonoverlapping of descriptions for a partition of taxa (by comparing their extensional coverage).
If a knowledge representation includes taxa which belong to different taxonomic rank, these taxa express a hierarchical classification instead of a partition. The descriptions of the class must be consistent with the inclusive relations of the taxa. Descriptive data managed at a taxonomic rank can be generalized to infer descriptions at an upper rank, the same way as specimen observations are generalized to infer the description of a species group.

Improvement of Xper2 data model was necessary before working on the generalization of descriptive data
- First, taxonomic hierarchies are now supported in Xper^2, but it currently does not provide users with functions to infer generalizations or to check the consistency of the taxonomic descriptions at different ranks.
- Secondly, the implementation of multi-descriptions (scope, or multi-instances) in the Xper^2 descriptive model was recently introduced. This modification of the descriptive model allows us to propose generalization methods that produce disjunctive generalizations. In this case, a taxonomic group is described by a set of descriptions, each one covering a subset of the group. Managing scopes and taxa hierarchies makes it possible to include such polythetic descriptions of taxonomic groups.
- But the descriptive model does not handle modifiers for explicit frequencies and distribution values. Only a comment field stores such information as plain text.

We are working to enhance the Xper^2 editor and to make it able to handle the generalization of descriptive data. Implementing a solution for this problem will facilitate incrementing descriptive data sets, and the generation of taxonomic descriptions using specimen descriptions.

Generalization context

The general problem is how to create and to update descriptions at a rank level k from descriptions at a lower rank k-1, with respect to the differences between the described classes.

Let I be a set of instances or concepts, and $o_i \in I$

Let {di} be a set of descriptors

$D(o_i)$ is the description of o_i : $D(o_i) = \{(d1(o_i)$ AND $d2(o_i)$ ... $dj(o_i)$ ….AND $dk(o_i))$, $(d'1(o_i)$ AND $d'2(o_i)$ ... $d'j(o_i)$ ….AND $d'k(o_i)$, (…)}

For each descriptor dj(o_i) includes all the possible values for o_i

A description is **conjunctive** if it includes a single set of attributes.

A description is **disjunctive** if it includes a set of conjunctive descriptions. It is the case if a concept is described by a set of instances or sub-concepts. In Xper2, a disjunctive description forces to create a scope for each conjunctive partial description.

Cover(o_1 , o_2) = true if $D(o_1) \cap D(o_2) \neq \varnothing$, which means that the two descriptions overlap, or have common elementary instances.

A description is **complete** for a set of instances or concepts if it includes all applicable descriptors and all possible states for this set of instances or concepts.

A description is **consistent** for a set of instances or concepts comparatively to a set of negative instances (counterexamples) if it does not cover any negative instance.

Simple conjunctive generalization sg()

Let us consider o_1 and o_2

$sg(o_1 , o_2) = \forall di , di(o_1) \cup di(o_2) \in sg(o_1 , o_2)$

sg is necessarily complete (due to the construction process) because sg covers all positive instances and considers all the descriptors. But sg is not necessarily consistent since sg can cover negative instances.

Common generalization cg()

Let us consider o_1 and o_2

$cg(o_1 , o_2) = \forall di\ di(o_1) \cap di(o_2) \in cg(o_1 , o_2)$

The building process does not cover negative examples, so cg is consistent (except if the positive and negative instances are not distinct). On the other hand, cg is not always complete since it considers only the common values of o_1 and o_2 and not all the possibles values of o_1 and o_2.

Most specific generalization msg()

Let us consider o_1 and o_2

$msg(o_1 , o_2)$ is a more specific generalization covering o_1 and o_2 if there is no other description $g(o_{1,}o_2)$ covering o_1 and o_2 such as $g(o_{1,}o_2) \subset msg(o_1 , o_2)$.

A consistent disjunctive generalization

(1) A version space algorithm (bottom-up algorithm, or aggregating algorithm)

To infer the description of a taxon from a set of specimens, we choose to apply the version space algorithm (Mitchell, 1982). This algorithm builds two sets of generalizations S (sets of all most specific generalizations consistent with the positive instances) and G (sets of all the most general generalizations matching every positive instances). S and G tend to converge when adding new positive and negative instances.

Initialize the sets S and G, respectively, to the sets of maximally specific and maximally general generalizations that are consistent with the first observed positive training instance.

for each subsequent instance, i
begin
if i is a negative instance,
then begin
--Retain in S only the generalizations which do not match i.
--Make generalizations in G that match i more specifically, only to the extent required so that they no longer match i, and only in such ways that each remains more general than some generalization in S.
--Remove from G any element that is more specific than some other element in G.
end
else if i is a positive instance,
then begin
--Retain in G only those generalizations that match i.
--Generalize members of S that do not match i, only to the extent required to allow them to match i, and only in such ways that each remains more specific that some generalization in G.
--Remove from S any element that is more general than some other element in S.
end
end

(algorithm extracted from Mitchell 1982).

This algorithm will always find a solution if the positive and negative instances are distinct. It is independent of the order to consider the instances.
This algorithm was developed to generalize instances (e.g. specimens to taxa). We extend it to generalize concepts (e.g. species to genera).
We test it on qualitative data but it is possible to adapt the algorithm for numerical data (in which case the generalization of two values is not a set but an interval).

Example:

Let us consider the following set of theoretical species (including polymorphism) and their genus attribution :

	d1	d2	d3	d4	d5
species 1 (genus A)	a	a	b	a	c
species 2 (genus A)	a	a	a	d	b
species 3 (genus B)	b	a	b	a/c	a
species 4 (genus B)	b	b	b	a	b
species 5 (genus A)	b	a	a	c	c

(1) Generalization of genus A with the version space algorithm:

step1: positif instance species 1

- ***S1*** = { (a, a, b, a, c) } S1 we initialize the sets S with the most specific generalization covering species 1, so S1 = D(species 1)
- ***G1*** = { (?, ?, ?, ?, ?) } we initialize G1 with the most general generalization covering species 1. the "?" means any value.

step 2: **positif instance species 2**

- ***S2*** = { (a, a, ab, ad, bc) } S1 don't cover species 2 so we generalize S1 to S2
- ***G2*** = { (?, ?, ?, ?, ?) } G covers species 2 so we don't modify it

step 3: negative instance species 3

- ***S3*** = { (a, a, ab, ad, bc) } S don't cover species 3 so we don't modify the sets *S*
- ***G3*** = { (a, ?, ?, ?, ?) (?, ?, ?, ?, bc) } G covers species 3 so we make G more specific and only in such ways that each remains more general than some generalization in S (two possibilities, so G3 includes the two maximally general generalization, or two possible external boundaries)

step 4: negative instance species 4

- ***S4*** = { (a, a, ab, ad, bc) } S don't cover species 4 so we don't modify the sets *S*
- ***G4*** = { (a, ?, ?, ?, ?) (a, ?, ?, ?, bc) (?, ?, ?, ad, bc) } G covers species 4 (second set) so we make this G set more specific and only in such ways that each remains more general than some generalization in S
- ***G4*** = { (a, ?, ?, ?, ?) (?, ?, ?, ad, bc) } Then we remove from G any element that is more specific than some other element in G : (a, ?, ?, ?, bc) is more specific than (a, ?, ?, ?, ?) so we remove it

step 5: positive instance species 5

- S don't cover species 5 so we have to extent the sets *S but if we create S5* = { (ab, a, ab, acd, bc) } *S don't remain more specific than the generalization in G (indeed* (ab, a, ab, acd, bc) and (a, ?, ?, ?, ?) overlap and (ab, a, ab, acd, bc) and (?, ?, ?, ad, bc) *too. So*
  S5 = { (a, a, ab, ad, bc), (b, a, a, c, c)}
- each **G** don't cover species 5 so we can't retain in G any generalization that match species 5 without to have G overlaping with S

Result of the generalization of genus A

Genus A is generalized in two morphotypes ***S5*** = { (a, a, ab, ad, bc), (b, a, a, c, c)} covering all the species of A and no external species:

morphotype 1 = (a, a, ab, ad, bc)

morphotype 2 = (b, a, a, c, c)}

These two morphotypes of genus A will be stored as two scopes in the knowledge base.

A simple generalization of genus B will be distinct from the two morphotypes of genus A and will be stored as a unique conjunctive description (b, ab, b, ac, ab).

Remark: a simple generalization for both genus A (ab, a, ab, acd, bc) and genus B (b, ab, b, ac, ab) will produce overlapping descriptions and sg(genus A) will cover many false combinations of character states, (sg(genus A) covers 24 different possible combinations ; the 2 morphotypes covering genus A includes only 9 possible combinations and a more accurate generalization).

This algorithm may be used to update descriptions when incrementing descriptive data (new specimens, news taxa, revision with modification of the classification).

(2) A top-down algorithm using identification key

A second way to characterize a set A of concepts (specimens or taxa) against other concepts not included in A (negative instances of A) is to build a decision tree, *i.e.* an identification key. The terminal units of this key are the set A and the negative concepts. The identification key includes one or more paths to conclude on the terminal unit A. Each path corresponds to a description composed of the descriptors and values of the internal nodes and branches, and to the union of the possible values in A for the other descriptors (not used in the key). A complete and consistent generalization of A is then composed of the disjunction of the descriptions of all the paths to identify A.

To implement this generalization method we adapt the algorithm already implemented in Ikey+: at each step the capability of each descriptor to separate the remaining concepts is not computed according to all the pairs of taxa, but limited to the pairs of taxa including one taxon of A and one taxon not in A.
This algorithm will be integrated in Xper^2 for generalization purpose.

To generalize a set of species into several genera, the algorithm builds for each genus:
- the identification key for this genus against the other species
- it creates the description for each path of the tree identifying A
- it adds as many scopes as necessary (as many scopes as paths in the key for A) to express the description of A in Xper^2.

This algorithm is independent of the order to generalize the descriptions. It was adapted from Vignes (1991). It verifies at the same time if the groups are distinguishable.

Different parameters have an impact on the topology of the key. Thus, it is possible to run Ikey+ with different parameters and then to choose one key to create the generalization.
A criteria to select a solution is the coverage of the generalizations in order to limit the extra combinations.

For a description with qualitative states, the coverage of a description is the product of the number of states for each descriptor:

(a, ab, c, a) covers two combinations: {(a, a, c, a), (a, b, c, a)}

Example:

Depending of the parameters to build the key we obtain different topologies and so different generalizations. We propose here three solutions on our theoretical example :

key 1 1-1 d1 = a → Genus A 1-2 d1 = b 2-1 d3 = a → Genus A 2-2 d3 = b → Genus B	**key 2** 1-1 d3 = a → Genus A 1-2 d3 = b 2-1 d1 = a → Genus A 2-2 d1 = b → Genus B	**key 3** 1-1 d5 = a → Genus B 1-2 d5 = b 2-1 d1 = a → Genus A 2-2 d1 = b → Genus B 1-3 d5 = c → Genus A
Genus A (a, a, ab, acd, bc) (b, a, a, acd, bc) coverage = 18 combinations	Genus A (ab, a, a, acd, bc) (a, a, b, acd, bc) coverage = 18 combinations	Genus A (ab, a, ab, acd, c) (a, a, ab, acd, b) coverage = 18 combinations
Genus B (b, ab, b, ac, ab) coverage = 4 combinations	Genus B (b, ab, b, ac, ab) coverage = 4 combinations	Genus B (b, ab, b, ac, b) (b, ab,b, ac, a) coverage = 8 combinations

How to choose a generalization method?

A simple generalization is much more simple to execute than any other algorithm. It is already implemented in Xper^2. If the simple generalization of a set of descriptive data (“g”) produces a description which is too large, this generalization is not disjoint from the descriptions external to the group, “g” is a polythetic group (otherwise it is a monothetic group).
To create generalization of descriptive data, we propose the following process:

(1)
- for each group g, create sg(g) with the union of all possible values for each descriptor element
- for each sg(g), check if sg(g) covers external instances or not. If yes, sg(g) is not consistent and a more sophisticated generalization has to be done (step 2) ; g is a polythetic group, it cannot be defined by a single description but by several descriptions, each one partially covering g. Otherwise, we keep sg(g) as the description of g (g is a monothetic group: it can be defined by a single description).

(2)
- if the generalization process is used to generalize from specimens to taxa we recommend the version space algorithm
- if the generalization process is used to generalize from taxa or from groups (including large polymorphism) to higher taxa or larger groups, we recommend the top-down algorithm with the

construction of identification keys

(3)
- the conjunctive (simple generalization) or disjunctive descriptions are then stored in scopes.

Limitations

The extension of the data model to give a weight to descriptors reflecting their importance (not available in Xper^2, in development in Xper^3) and to add modifiers (“rarely”, “frequent”, or probability) will allow data-driven generalization and thus help solve the following problems:

- How to use data properties (importance of a character, taxonomic relationship …) to guide the generalization process ?
- How to add frequency in the generalization ?

Implementation

We are developing the prototype of the algorithms described in this document. After their validation, improved versions will be integrated into Xper^2 and Xper^3. This will complement the checkbase function (to test description consistencies in case of taxonomic hierarchy and in case of multi-instances) and provide the descriptive data editor with a new merging function (only simple generalization is currently available).

References:

MITCHELL, T.M., "Generalization as search" in Artificial Intelligence, Vol 18, N°2 pp. 203-226, March 1982. (http://www.cs.cmu.edu/~tom/mlbook.html)

Vignes R., 1991. Caractérisation automatique de groupes biologiques: Thèse de l’UPMC, INRIA, 1991, ISNB 2726107427, 9782726107423. 260 pages

http://fr.wikipedia.org/wiki/Espace_de_versions