Multi-dimensional Representations of Conceptual Hierarchies

Peter Becker

Distributed System Technology Centre (DSTC)
Knowledge, Visualization and Ordering Laboratory (KVO)
Griffith University
PMB 50, Gold Coast 9726, Australia
pbecker@dstc.edu.au

Abstract. Dealing with conceptual hierarchies is a fundamental task when working with conceptual structures. In this paper we present a projection of the implications underlying such a hierarchy into the n-dimensional space. This vector representation of the structure allows more complex mathematical treatment of the conceptual hierarchy.
A first implementation of this idea is an enhanced representation of concept lattices from Formal Concept Analysis (FCA). From this representation we generate two-dimensional diagrams that respect certain aspects of the underlying semantics. In addition, manipulation and general interaction with the diagram become much more intuitive to the user. The experience with this implementation shows that a well-founded representation of the conceptual hierarchy improves the usability of a system based on this hierarchy.

1 Introduction

One of the most important applications of the theory of conceptual structures is building systems people are able to interact with in an intuitive fashion. When a system for conceptual structures is built, some theory about the world to be modeled is stated and some knowledge about dependencies and implications between concepts or attributes is externalized. For example when defining a system of conceptual graphs the concept type hierarchy is of fundamental importance, describing a theory of implications. Likewise the scales of an Conceptual Information System as implemented e. g. with Toscana ([VW95]) represent a set of local theories, describing the world of objects to be treated by the system. For the scope of this paper we consider concept lattices as conceptual structures in Conceptual Information Systems, but the ideas described apply likewise to representations of concept type hierarchies. We assume the reader knows the basic concepts of FCA as presented in [GW99].

While developing Cernato, a data analysis tool from Navicon GmbH, we searched for a representation of concept lattices that allows the user to interact

with the diagrams in a way he intuitively perceives the conceptual structure inherent in the data. The data analysis performed with CERNATO starts with an empty scaling, i. e. the concept lattice has no attributes and all objects belong to the one concept displayed. By adding scale attributes (from scales previously entered) he can then extend the diagram and change the layout by dragging points around.

The algorithm we required had two main aspects: first, the diagram should be easily readable after the addition of an attribute. Second, and even more important, the change of the layout of the diagram when interacting with it should help the user to understand the conceptual structure of the data. This latter point is considered even more important than the aesthetics of the diagram itself. For this reason, the first approach, using the algorithms from the FCA Library developed by Frank Vogt (cf. [Vo96]) was unsuccessful. These algorithms are based on geometric and aesthetic considerations and are not intended to retain semantic information.

	Type	*Color*	*Size*
Yoyo	cat	black	small
Samson	dog	brown	large
HAL	computer	red	huge

Table 1. Example context with three multi-valued attributes

	animal	*cat*	*dog*
cat	×	×	
dog	×		×
computer			

Table 2. A conceptual scale for the multi-valued attribute "Type"

For CERNATO we tried to split the ontological implications, i. e. those which are always true in the model the user defined from those found in the current data set but that can not be derived from the model. See the context in Table 1 and the scale for the multi-valued attribute "Type" in Table 2 which contain examples for both types of implications: there are ontological implications like "every cat is an animal" which can be found in the structure of the scale and some other implications like "each brown object is an animal" which is supported by the data set but not generally true. The first implication can be found by looking at the scale only while the latter needs the context to be found.

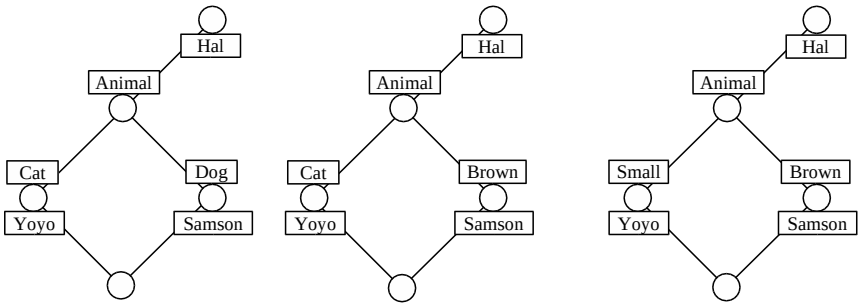

Fig. 1. Examples of misleading diagrams: identical layout with three alternate semantics

Fig. 1 presents some examples of line diagrams based on this context that demonstrate how the diagrams produced by the FCA Library can be easily misinterpreted – we use a nominal scale for the attributes "Color" and "Size" in the diagrams. While all three diagrams use the same layout, which is a typical layout for the FCA Library, they have different meanings. Intuitively, the user interpretes the fact that the vector pointing to the left attribute is prolongated as indicating a stronger implication between the left attribute and the one above in each diagram compared to the implication between the right attribute and the one above.

This is true only for the diagram in the middle, where "cat" is an ontological refinement of "animal" while a "brown object" could be something else – although the data used for the diagram does support the implication "each brown object is an animal" as stated above. The left diagram shows two implications which are true in general and not only for the given data set, but one implication is accentuated by the layout. The right diagram shows the opposite situation where all implications are supported by the data analysed but not by the model entered by the user. The idea behind Cernato is to create diagrams that emphasize those implications that are true in general by laying them out in chains if possible. The information on the model is extracted from the structure of the conceptual scales defined for the multi-valued attributes.

No knowledge about implications valid in general is used by the FCA Library, thus it is producing the same diagram layout for all three situations. The layout is also dependent on the order in which the attributes are treated by the algorithms – the diagram in the middle could even be created with the left and right point exchanged, thereby emphasizing the weaker implication.

One of the main goals in the development of Cernato (and thereby in the algorithms described here) was to use the additional information in the conceptual scales to present line diagrams in a more meaningful manner. This relates

also to the problem of allowing changes to the diagram layout that repect this underlying structure – a problem not addressed in tools like ANACONDA, where the user may change the diagram layout, even if the result leads to diagrams that are easily misinterpreted. The flexible layout of ANACONDA was dropped to get a more stable editing user interface that allows even inexperienced users to edit the diagrams.

However, it is important to note that the intuitive understanding of the conceptual structure isn't always inherent in a static diagram produced by CERNATO, but mainly in the animated behaviour it shows when the user interacts with the diagram. While the first layout CERNATO creates is often not as aesthetically pleasing as the layout from the FCA Library, it ensures that the semantics that can be extracted from the scales are displayed and the option of moving the points allows the user to get a more aesthetically pleasing diagram unless there is a direct conflict between the aesthetic criteria and the semantics.

2 *N*-Dimensional Projection of Concept Lattices

To be able to display the semantics in a meaningful manner we project the lattice into a n-dimensional space first. This gives us the degree of freedom needed to model complex relations in a diagram, while a two dimensional diagram is not suited for this. This idea is not dissimilar to the idea of finding the minimal set of chains on irreducible concepts as done in the FCA Library ([Sk89],[Vo96]), but there are two main differences in what we do: first we use the order found on the scales of the multi-valued attributes instead of the order of the context. Additionally we keep the n-dimensional structure and reuse it when the user moves points.

The attribute order which we try to respect by our projection is a partial order on the attributes. If the attribute m implies the attribute n, then m is more specific and we write $m \leq n$. As seen in the examples in the previous section, there are different kinds of dependencies between attributes. Some implications are ontological (like "cat" implying "animal"), others specific to the data analysed (like "brown" implying "animal"). The latter implication is not any "less true", but it is dependent on the given data set. To make the interaction with the diagrams in CERNATO more intuitive, we consider only the ontological implications given by the user in the scales. Therefore we have $m \leq n$ if there is a conceptual scale $(G_{\mathbb{S}}, M_{\mathbb{S}}, I_{\mathbb{S}})$ in the Information System with $m, n \in M_{\mathbb{S}}$ and $m^{I_{\mathbb{S}}} \subseteq n^{I_{\mathbb{S}}}$.

The goal for the projection into the n-dimensional space is to respect this order partially when translating the attributes into $\mathbb{Z}^n$. The vectors of $\mathbb{Z}^n$ are naturally ordered by the component-wise comparision: If $v := (v_1, v_2, \ldots, v_n)$ and $w := (w_1, w_2, \ldots, w_n)$ are vectors of $\mathbb{Z}^n$, then we write $v \leq w$ if and only if $v_i \leq w_i$ for every $i = 1, \ldots, n$.

Some of the attributes of a scale might be just combinations of other attributes such that saying that an object has the attribute m is equivalent to

saying that it has the attributes in $B \subseteq M$ ($m \notin B$). These attributes are called reducible:

Definition 1 (reducible). *Let $(G_{\mathbb{S}}, M_{\mathbb{S}}, I_{\mathbb{S}})$ be a scale in a Conceptual Information System, $m \in M_{\mathbb{S}}$ an attribute of this scale and $B \subseteq M_{\mathbb{S}}$ with $m \notin B$. If $\bigcap_{b \in B} b^{I_{\mathbb{S}}} = m^{I_{\mathbb{S}}}$ for any such B the attribute m is called* reducible *with respect to the scale.*

These are the attributes in scales that are removed when purifying the context of the scale (cf. [GW99]). Note, that attributes which are reducible with respect to the partial order on the attributes defined above aren't necessarily reducible with respect to the scale. In this paper, "reducible" means reducible with respect to the corresponding scale. Since reducible attributes don't provide additional information we ignore them in our projection by assigning them a zero vector.

Our algorithm to project the conceptual structure into the n-dimensional space $\mathbb{R}^n$ has two parts. The first assigns a vector in $\mathbb{Z}^n$ to every attribute, the second calculates the projection for the concepts into $\mathbb{R}^n$ based on the results of the first algorithm and thereby projects the whole lattice.

Algorithm 1. `calculateAttributeVectors()`

```
for all Attributes m do
  m.vector() ← 0
  if m is not reducible then
    if m has no predecessor or at least one predecessor of m has other successor
    then
      create new dimension d
      m.vector(d) ← 1
    else
      for all predecessors m_p of m do
        for all existing dimensions d do
          if m_p.vector(d) = 1 then
            m.vector(d) ← 1
          end if
        end for
      end for
    end if
  end if
end for
```

The basic idea for the projection is to reuse vectors for the chains in the attribute order as long as two points are not projected onto the same coordinates. For example we want to project the implication "every cat is an animal" into a chain, since it is a chain in the attribute order. Since the same is true for the

implication "every dog is an animal" we would create the same projection for two points if we reuse the vector for "animal" for both, "cat" and "dog" in the diagram in the middle of Fig. 1.

Whenever this happens we add new dimensions into the projection vectors for all attributes that collide. Since an attribute belongs to a chain in the lattice if and only if it is a lower neighbour of another, the conflicting situation is the situation where a predecessor of an attribute has more than one successor – in this situation Algorithm 1 creates new dimensions for the conflicting attributes.

The resulting vectors for the attributes have only ones and zeroes as coordinates, which can be read as the attribute having a specific semantic content or not. Of course this means that they have different lengths in the Euclidean space as well. Since we don't want to use different lengths in the line diagram for the lattice, we normalize vectors in the usual manner before using them to project the concepts.

Algorithm 2. `projectConcept`(C)

C.`vector`$() \leftarrow 0$
for all m in the intent of C **do**
  C.`vector`$() \leftarrow C$.`vector`$() + \frac{m.\texttt{vector}()}{||m.\texttt{vector}()||}$
end for

The projection algorithm for the concepts is shown in Algorithm 2. It simply sums all normalized vectors for the attributes to calculate a position in $\mathbb{R}^n$, thus creating an additive drawing of the lattice structure. The vectors are normalized since the length of the attribute vectors depend on the amount of semantic content they have, but we don't want to have different distances in the diagram created. The results for the diagrams of Fig. 1 are shown in Fig. 2 and the attribute order with the vectors for the attributes shown in Fig. 3. The attribute vectors in the examples are not normalized to simplify the values.

As can be seen in the example in the middle of Fig. 3, the chain of attributes ("Cat" and "Animal" with "Cat"$\leq$"Animal") is layed out as a chain in the n-dimensional space while the outer diagrams are using three dimensions – one for each attribute. Although the user can choose to create chains for the outer diagrams by dragging points in Cernato, he is not able to break the chain in the diagram in the middle. This will be discussed in detail in Section 4.

A slightly more complex example shows how two vectors are combined in the algorithm. In Fig. 4 a context is presented which has numeric intervals as attributes and numbers as objects. The attribute order is shown with the vectors for the attributes, using the interval inclusion order. Note that the vector for the small interval is the combination of both – the effect of this is that the lowest edge in the diagram (ending at the bottom) is a direct prolongation of the diagonal of

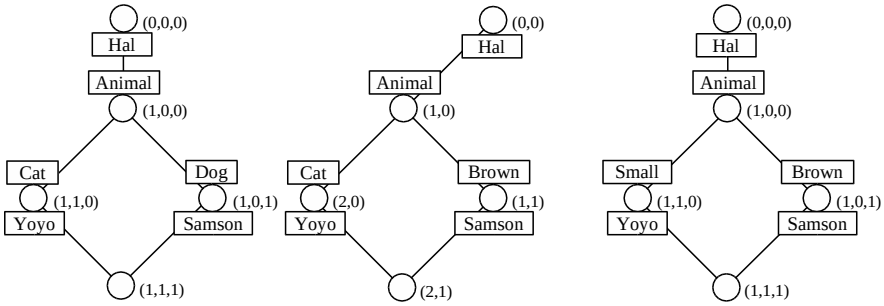

Fig. 2. Examples of vector assignments using the new algorithm (without normalization).

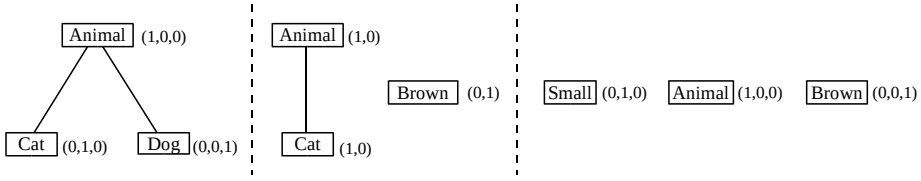

Fig. 3. The attribute order for the examples and the vectors assigned.

the diamond above, depicted as a dashed line. In this way the diagram presents the lower edge as a continuation of the combination of the upper two vectors.

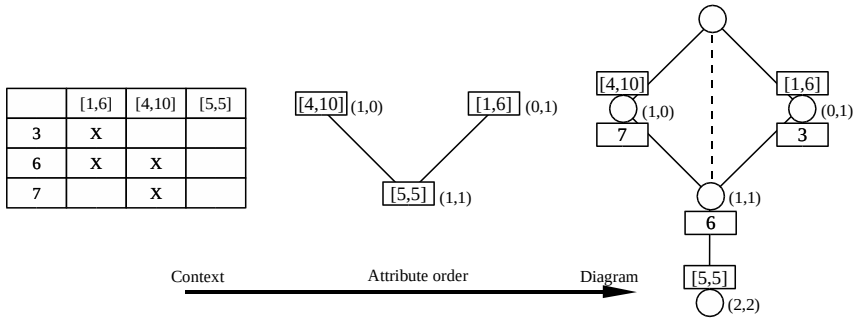

Fig. 4. A more complex example, introducing the combination of two vectors.

3 Projection Onto the Plane

The process of projecting the n-dimensional structure onto the Cartesian plane is an ordinary parallel projection. The parallel projection is used since it keeps the additive drawing of the line diagram that we created in $\mathbb{R}^n$: you can find a set of vectors in $\mathbb{R}^2$ for the infimum irreducible concepts and then find the vector for each concept in the lattice by adding all vectors for the infimum-irreducible concepts above. In this way we get the common layout pattern of an additive diagram which is drawn from top to bottom. Many projects have shown that this kind of diagram layout is suitable to create diagrams that are easy to read (see e. g. [Sk89]).

When using parallel projection we have to find a set of vectors in $\mathbb{R}^2$ for the projection base. There needs to be exactly one vector for each infimum-irreducible concept in the lattice – these are the concepts which introduce new dimensions, each concept that can be written as the infimum of two concepts other than itself is assigned the combination of the vectors of all infimum-irreducible concepts above it. This is in fact a design issue for the algorithms – if you attach a dimension to each attribute and thereby to each attribute concept, including reducible concepts, you get a different layout. This was not wanted for Cernato, as it is less structured. Therefore, we decided to attach dimensions and thereby base vectors only to infimum-irreducible concepts.

Another aspect that has to be addressed when projecting the diagrams relates to the reading rules for line diagrams: the diagram is in fact a directed graph but the edges don't have any arrows since it is assumed that their direction is depending on the position of the nodes. This leads to a restriction for the projection base: all vectors have to point downwards.

Although the basic projection is simple there remains the issue of finding suitable vectors for the projection base. Since Cernato creates the diagram incrementally, by adding single attributes into the structure, and the old projection should be kept to let the user see the incremental change, the problem of finding the base is reduced to finding a single new base vector.

The existing base vectors might have been changed by the user so the algorithm for finding the new vector can't assume any knowledge about the existing base. The first approach was to take all existing vectors and put a new vector into the biggest free angle – adding two edges to the left and right for ensuring the downward direction. The first vector created will be the vertical one (assuming the left and right borders are symmetrically set).

The results of this were not satisfactory. For example it is often possible to use the same vector for more than one dimension, the most obvious example is a conceptual lattice that is a chain with respect to the concept order and with the n-dimensional vectors: $(0, 0)$, $(1, 0)$ and $(1, 1)$. This can be displayed as a chain if and only if the two dimensions will be projected into one direction, i. e. if they use the same base vector. Even if it is possible to break this chain later, because it is not derived from implications that are true in general, it is often convenient

to get it in the first layout since it is aesthetically more pleasing. The result of the simple approach for this diagram would be the vertical downward vector for the first dimension and a vector close to the diagonal for the second. This is usually not the diagram one expects and there are further examples where this approach fails to achieve asthetically pleasing or even easily readable diagrams.

The approach taken for CERNATO is only slightly different: instead of using the whole base for finding the free space we use a smaller set of vectors. This set of vectors is found by looking at the way the ideal of the attribute concept connects to the rest of the diagram. Obviously the lines connecting the ideal to the rest of the diagrams should not use the same directions as other children of the upper neighbours of the ideal. Since all new points in the ideal are connected to old points in the rest of the diagram along the new dimension, we do not want to use directions for the new projection vector that are already used for edges between upper neighbours of the ideal and their children in the rest of the diagram.

Algorithm 3. `findNew2DVector(Concept` C`)`

$I \leftarrow$ ideal of C
$U \leftarrow$ all upper neighbours of concepts in I that are not in I
$V = \emptyset$
for all C_u in U **do**
  for all C_c which are child of C_u but not in I **do**
    $V \leftarrow V \cup (\text{coord}(C_v) - \text{coord}(C_c))$
  end for
end for
if $V = \emptyset$ **then**
  Use vertical downward vector as new projection vector
else
  Find downward vector such that the angles to vectors in V are maximized
end if

The algorithm used is presented as Algorithm 3, which takes the new attribute concept as a parameter and assuming that it is a new concept not existing in the diagram without the new attribute. This algorithm ignores the fact that there are more directions involved whenever an old concept was only moved and not duplicated – in this case the direction of its connection to its upper neighbour outside the ideal will be a combination of the old difference and the new projection vector. But the implementation in CERNATO demonstrates that the result of this simple algorithm is easily readable in most of the cases and was sufficient for the purpose of CERNATO. There where only slight changes for avoiding symmetry (which leads to collisions) and using a different weighting, preferring vertical vectors but stretching the flat ones.

4 Moving Points in the Diagram

As stated before we wanted the user to be able to change the layout of the diagram without breaking the projection of the underlying semantics. Of course the first restriction coming from the reading rules of the diagram is easily implemented by ensuring that the vectors in the projection base always point downwards. But how can the semantics of the diagram hold while the user is able to move every point in the diagram[1]? We re-use the example shown in the middle of Fig. 1 and show how a movement of each of the points would change the diagram (see Fig. 5).

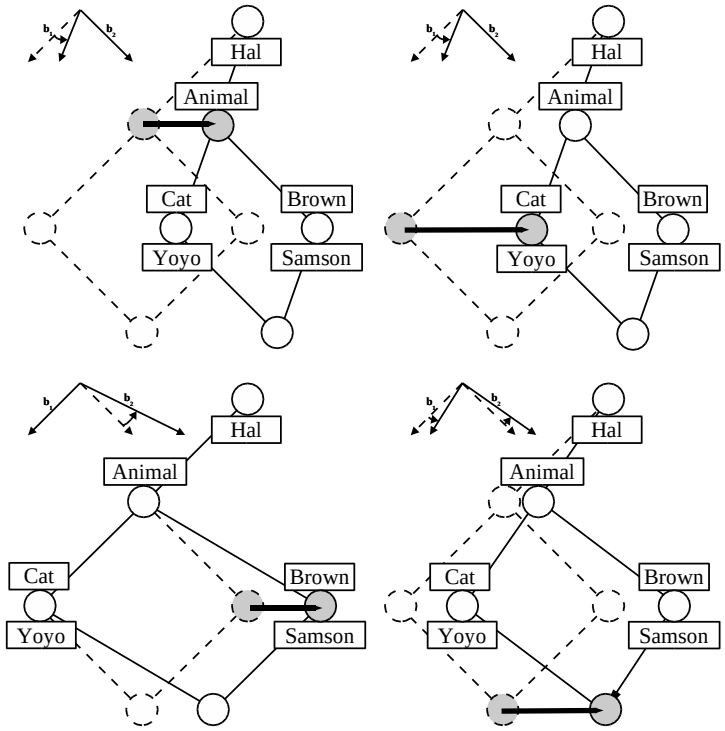

Fig. 5. Examples of algorithm behaviour when individual points are moved

The upper two diagrams show how the movement of a chain is handled: whenever one of the points in the chain is moved, the base vector for this dimension is changed so the point is projected to the new position. In the examples shown only the horizontal coordinates of the points are changed, the chain is projected into dimension one in the n-dimensional structure and therefore the

[1] Except the top which is used as origin for the whole diagram

x-coordinate of the projection vector from dimension one onto the plane has to be changed to move the point to its new position.

This is true for the upper two examples, although the change is related to the position in the chain: if the lower point is moved, the change in the base vector is smaller compared to a movement of the upper point – this behaviour is similar to a lever and is a direct result of the parallel projection used. Another effect of the parallel projection is the change of all points below – since the other projection vectors are kept, all points below are also moved.

If the right point is moved (lower left diagram in Fig. 5), the behaviour is quite similar. This time the point has one connection to an upper neighbour which is in the direction of dimension two in the n-dimensional diagram. So this time the projection vector for the second dimension has changed, thus moving the point itself and the bottom element, which is below the moved point and has to be moved to keep the parallel projection.

If the bottom element of the example is moved, the situation gets a little more complex. This time there is no single vector which can be used to find the projection vectors to change. The algorithm used in CERNATO in this situation is to distribute the movement along the projection vectors of each dimension in which the point differs from one of its successors. This results in a kind of shearing of the whole diagram, an effect that is quite welcome in this situation.

A more detailed version of this algorithm can be found in Algorithm 4. The parameters are the point moved and a vector representing the change in the plane that the user indicated by dragging the corresponding point.

Algorithm 4. `movePoint(Concept` C`, 2DVector` $v2d$`)`

let v be a vector in $\mathbb{R}^n$
$v \leftarrow 0$
for all concepts P that are predecessors of C **do**
  $v \leftarrow v + C.\texttt{vector}() - P.\texttt{vector}()$
end for
$d \leftarrow$ number of dimension in v which are $\neq 0$
for all dim which are dimensions of the n-dimensional structure **do**
  if $v(dim) \neq 0$ **then**
    change the projection vector for dim by $\frac{v2d}{d * C.\texttt{vector}(dim)}$
  end if
end for

The n-dimensional structure stays unchanged while the projection onto the plane is changed, which can be related to a specific movement of the viewpoint in the n-dimensional space. This might be a reason why the process of moving the diagram is experienced as intuitive by users.

5 Extending the Diagram

Another important aspect when using FCA interactively is the way diagrams are expanded when the lattice is refined by adding additional attributes. Although the space in this paper isn't sufficient to describe the process used in Cernato in detail, we will try to give the basic idea. Whenever the user adds a new attribute to the existing diagram, Cernato extends the diagram in a way that the relation between the old and the new can be seen. This is done by keeping as much structure as possible from the old diagram and animating the process of the change.

The concept used in Cernato is based upon an algorithm presented by Godin and Missaoui ([GM94], for a more complete description see [Be99]). The algorithm extends an existing lattice by adding an additional attribute concept and building the closure with respect to the infimum operation, i. e. adding all infima between existing concepts and the new attribute concept.

All the new concepts created are either the attribute concept itself or infima of it and some concept that already existed in the old diagram. For this reason the new projection base needs at most one new vector. All points in the ideal of the attribute concept are either new points, created as infimum from the attribute concept and an existing point, thus being different from their predecessor in the existing diagram structure only by the vector of the attribute concept, or they are a concept that existed in the old diagram but now its intent has been extended with the new attribute. In this case its position has been changed by the vector for the new attribute.

This means that the attribute concept and all points below are related to a concept in the old diagram – either to its old position or to one of their predecessors in the new diagram, the one that is not in the ideal of the attribute concept. This fact is used by Cernato to create an animation from the old to the new diagram: the whole ideal of the new attribute concept is moved out of the old diagram structure, partly by moving existing points, partly by duplicating points and moving the new points.

The opposite process of removing attributes or the dual processes of adding and removing objects are not implemented yet. The algorithms needed for doing this can be derived by reverting the Godin & Missaoui algorithm and/or using its dual form.

6 Further Research

The author is currently evaluating a new algorithm, for a similar effect, that behaves in a slightly different and more consistent way. The basic idea for this algorithm is to find the maximal chains in the attribute order and assigning each of them a dimension and thereby a vector with a single coordinate set. An attribute is assigned the sum of all the vectors of the chains it is part of.

The major difference of this approach in comparison with the implemented algorithm can be seen by looking at the left diagram in Fig. 2: the chains for the implications "every cat is an animal" and "every dog is animal" are simply ignored to avoid conflicts in the projection. The new algorithm should create a diagram using only two dimensions, prolongating the diagonal of the diamond in the same manner as done in Fig. 4. The new projection is shown in Fig. 6.

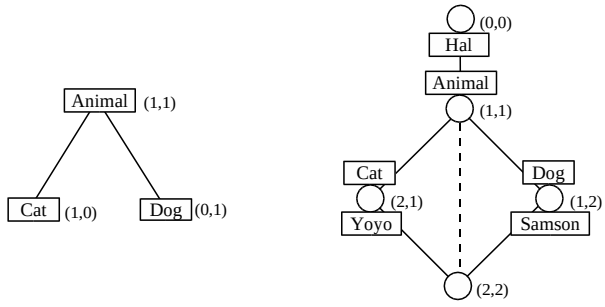

Fig. 6. Layout using the new algorithm

To allow sharing algorithms and code between different applications, the author started an Open Source project which can be found on the World Wide Web ([Tockit]). The project plans to implement a framework for Conceptual Knowledge Processing in Java, including the algorithms described in this paper. Within this framework applications for special purposes will be built by combining suitable components, like a component for editing conceptual scales in an ANACONDA-like style and another for presenting line diagrams in a Webbrowser. The project is also intended to synchronize the implementation of scientific work on this topic. There will be information available on the Website and the communication structures like mailing lists and news forums allow the sharing of ideas between researchers and developers.

Another interesting aspect for further work is to create a more abstract version of these algorithms to show their potential to other fields of Conceptual Knowledge Processing. Since we operate only on the partial order of the attributes, the algorithms may well be applicable to a large number of Conceptual Knowledge Processing tools. The work of Barwise & Seligman ([BS97]) seems to give valuable terms and patterns for this goal.

7 Conclusion

The success of CERNATO in target groups, formerly not addressed by tools like TOSCANA and ANACONDA, shows that by easing the interface to comfort untrained users, the methods of Formal Concept Analysis can be used in different,

more-demanding environments. The interaction based on the conceptual model described here and the use of animation and of terms comprehensible to the user leads to an intuitive understanding of the underlying concepts and allows a more experimental process for creating conceptual systems.

From the experience gained with CERNATO we can learn for future applications. Although CERNATO has been successful in the applications it was designed for, there is still a need for other tools like TOSCANA, which offers the ability to handle large data sets and to present a simpler user-interface due to a reduced set of functionality offering less interactivity. Experience using both tools should be used for constructing new applications.

8 Acknowledgments

The author wants to thank Jo Hereth for helping him understanding the work presented in this paper and refining the algorithm – including the ideas presented in Section 6. Also NaviCon GmbH contributed much to this paper by giving him the chance to work on CERNATO. Further thanks go to Peter Eklund and Thomas Tilley for their comments on the final drafts.

References

[Be99] P. Becker: *Einsatz der Formalen Begriffsanalyse zur Dokumentennavigation*. diploma thesis (in german) Computer Science Department of the Philipps Universität Marburg. 1999.

[BS97] J. Barwise, J. Seligman: *Information Flow: the Logic of Distributed Systems*. Cambridge Tracts in Theoretical Computer Science, 44, Cambridge University Press, 1997.

[GW99] B. Ganter, R. Wille: *Formal Concept Analysis : Mathematical Foundations*. Springer Verlag, Berlin – Heidelberg – New York, 1999.

[GM94] R. Godin, R. Missaoui: *an incremental concept formation approach for learning from databases*. In: *Theoretical Computer Science*, **133**, p. 397–419, 1994.

[NaviCon] Homepage of NaviCon: http://www.navicon.de

[Sk89] M. Skorsky: How to draw a concept lattice with parallelograms. In: R. Wille (ed.): *Klassifikation und Ordnung*, p. 191–196, Indeks-Verlag, Frankfurt 1989.

[Tockit] Homepage of the TOCKIT project: http://tockit.sourceforge.net.

[Vo96] F. Vogt: *Formale Begriffsanalyse mit C++– Datenstrukturen und Algorithmen*. Springer, Berlin – Heidelberg – New York 1996.

[VW95] F. Vogt, R. Wille: TOSCANA — a graphical tool for analyzing and exploring data. In: R. Tamassia, I. G. Tollis (eds.): *Graph Drawing '94*. LNCS **894**, p. 226–233, Springer, Berlin–Heidelberg 1995.