B/K-3-
André

Good

A FORMALIZATION OF METAPHORS AND IMAGE-SCHEMAS IN USER INTERFACES

WERNER KUHN, ANDREW U. FRANK
National Center for Geographic Information and Analysis
and Department of Surveying Engineering
University of Maine
Orono, ME 04469 (USA)
Bitnet: Kuhn@mecan1, Frank@mecan1

ABSTRACT. Sound engineering approaches to user interface design require the formalization of key interaction concepts, one of them being metaphor. Work on interface metaphors has, however, been largely non-formal so far. The few existing formal theories of metaphor have been developed in the context of natural language understanding, learning, or reasoning. We propose to formalize interface metaphors by algebraic specifications. This approach provides a comprehensive formalization for the essential aspects of metaphorical user interfaces. Specifically, metaphor domains are being formalized by algebras, metaphorical mappings by morphisms, and image-schemas by categories. The paper explains these concepts and the approach, using examples of spatial and spatializing metaphors.

1. Introduction

Metaphor pervades communication. Metaphorical thought, action, and language are not only essential to interpersonal communication [Lakoff and Johnson 1980], but to human-computer communication as well. Since communication with computers is an outcome of design, building effective user interfaces requires a formal theory of metaphor [Carroll, Mack, and Kellogg 1988].

This paper presents a methodological foundation for such a theory. It proposes a formalization of interface metaphors based on abstract algebra [MacLane and Birkhoff 1967]. The algebraic approach allows designers to specify not only metaphor domains, but metaphorical mappings and image-schemas as well. The paper describes the basic elements of the approach.

After explaining the underlying understanding of interface metaphors in Section 2, important related work is discussed in Section 3. The formalization method is presented with a running example in Section 4 and applied to spatial zooming operations in Section 5. The last section contains conclusions and suggestions for future work.

2. Interface Metaphors and Image-Schemas

Systems with user interfaces based on the DESKTOP metaphor and its variants [Apple Computer 1987, Smith et al. 1982] have established metaphor as a powerful means to control complexity in human-computer interaction [Carroll, Mack, and Kellogg 1988]. The structure and role of interface metaphors are, however, still understood in many different ways [Blumenthal 1990, Carroll, Mack, and Kellogg 1988, Erickson 1990a, Smith et al. 1982, Wozny 1989].

The understanding of interface metaphors underlying our formalization derives from recent work in cognitive linguistics [Johnson 1987b, Lakoff 1987, Lakoff and Johnson 1980]. Johnson [1987b, p. XIV] has characterized metaphor as

"*...a pervasive mode of understanding by which we project patterns from one domain of experience in order to structure another domain of a different kind.*"

To appear in: Mark, D. M. and A. U. Frank (Eds.). *Cognitive and Linguistic Aspects of Geographic Space.* NATO ASI Series. Kluwer Academic Press.

The two domains are commonly called the *source* and *target* domains of a metaphor and the metaphorical projection is a *mapping* from source to target.

Interface metaphors, in this projective view, go beyond explaining unfamiliar domains to novices. They determine how labor is distributed between a user and a system, what concepts a user has to deal with, and in what terms user and system communicate. In short, they structure application domains and organize tasks.

It is not clear how much metaphor-independent structure there can be in a target domain. A recent case study of metaphors in interface design [Erickson 1990a] indicates that the structure available to the designer before a metaphor is consciously chosen is either quite abstract or already metaphorical. In Erickson's example of a DESKTOP metaphor extension, that structure consists of links between documents. Interestingly, the link structure is a so-called *image-schema*. Image-schemas are image-like reasoning patterns, consisting of a small number of parts and relations, made meaningful by sensori-motor experience [Johnson 1987b, Lakoff and Johnson 1980]. Examples are the CONTAINER, SURFACE, LINK, PATH, NEAR-FAR, PART-WHOLE, and CENTER-PERIPHERY schemas.

Image-schemas may in fact be the kind of structure which is preserved by interface metaphors. This assumption agrees with Lakoff's invariance hypothesis [Lakoff 1990] which claims that image-schemas remain invariant under metaphorical mappings. It also generalizes to user interfaces of any kind the suggestion that image-schemas play a fundamental role in interfaces of Geographic Information Systems [Mark 1989]. The design of interfaces for international use [Nielsen 1990] is one area where image-schemas, due to their presumed culture-independence, are likely to be of particular importance.

3. Existing Metaphor Formalizations

While there is a vast literature on metaphor and analogy (e.g., [Johnson 1981, Lakoff and Johnson 1980, Mac Cormac 1985, Ortony 1979, Vosniadou and Ortony 1989]), formal approaches are rare (see [Hall 1989] for a survey and [Indurkhya 1989, Martin 1990] for recent treatments). Most existing formalizations employ traditional knowledge representation formalisms, such as semantic networks, to specify metaphorical domains, but not mappings or image-schemas. Furthermore, the formalizations have been developed for natural language understanding, machine learning or automated reasoning, while discussions of metaphor in human-computer interaction have been largely non-formal so far.

The best known formal theory of metaphor and analogy is Gentner's structure mapping theory [Gentner 1983, 1989]. It describes analogies as mappings between source and target domains, each represented by semantic networks. It does not formalize the mappings themselves, however, and rests on a syntactical distinction of different kinds of relations. While Gentner's theory deals with structural aspects, it neglects the role of tasks in metaphor use [Carroll, Mack, and Kellogg 1988].

Our formalization addresses these problems by formalizing mappings as morphisms and expressing tasks and actions through algebraic operators and their effects. Describing domains algebraically rather than relationally may only be a syntactic difference; it does, however, allow for relating metaphors to tasks. For example, the informal task description of parts of the DESKTOP metaphor given in [Carroll, Mack, and Kellogg 1988] can directly be expressed by an algebraic formalism (see next section).

In the terminology of Carroll et al., our approach is primarily *structural*, addressing the representation of metaphor domains and mappings, but incorporates the *pragmatic* aspects of actual user tasks. It does not deal with operational analyses of the effects of interface metaphors on learning or using a system.

Most theories of metaphor stress the importance of mapping source structure into the target domain. Surprisingly, the mathematical devices for describing structure mappings, i.e., morphisms, are rarely applied. An exception is a forthcoming treatment [Indurkhya 1989] which describes cognitive models by algebras and metaphorical projections by morphisms.
Our independently developed algebraic approach differs from Indurkhya's theory mainly by:

- describing metaphorical domains as elements of algebra *classes*, thereby dealing with the relation of metaphors to image-schemas;
- specifically addressing *interface* metaphors;
- using the established software design technique of algebraic specifications.

Existing metaphor formalizations provide little support for interface designers because they are either purely theoretical or rely on specialized software in prototype form. The advantage of using the technique of algebraic specifications is that it allows designers to apply the method immediately, even with the possibility of using computer supported design tools such as Larch [Guttag, Horning, and Wing 1985] or CLEAR [Burstall and Goguen 1981].

4. An Algebraic Approach

4.1. ALGEBRAIC SPECIFICATIONS

Algebraic specifications describe objects in terms of operations. The associated theory has been developed and used for specifying abstract data types [Goguen, Thatcher, and Wagner 1978, Guttag and Horning 1978]; practical introductions can be found in [Ehrig and Mahr 1985, Liskov and Guttag 1986, Woodcock and Loomes 1989].
An algebraic specification consists of three parts:

- a list of *sorts*, naming the objects involved;
- a list of *operators* (Ops), giving the sorts of arguments and results;
- a list of *axioms* (Eqs), defining equivalences for different sequences of operations.

Sorts and operators together are called the *signature* of an algebraic specification. *Algebras* are interpretations of signatures, assigning a set to each sort and an operation on these sets to each operator. An algebraic specification, i.e., sorts, operators, and axioms together, describes a *category*, i.e. the class of algebras satisfying the specification [Ehrich, Gogolla, and Lipeck 1989].

The technique of algebraic specifications is best demonstrated by applying it immediately to the formalization of source and target domains from Apple's composite DESKTOP metaphor [Apple Computer 1987].

4.2. SOURCE AND TARGET DOMAINS

The following is an algebraic booleans specification of physical office desktops. It lists three sorts: Desktops, Folders, and ~~Bools~~ (truth values). Desktops are the sort actually being specified, Folders will be specified below, and Bools are assumed to have been specified previously. There are operators to create a new desktop, put a folder on a desktop, get a folder from a desktop, and check whether a folder is on a desktop. The meaning of these operators is defined by the equations and does not rely on an interpretation of their names. The equations make free use of universally quantified sort variables (dt for desktops, f for folders, etc.).

```
Desktop
Sorts     Desktop, Folder, Bool
Ops  new: → Desktop
     put: Desktop x Folder → Desktop
     get: Desktop x Folder → Desktop
     on:  Desktop x Folder → Bool
Eqs  on(new,f) = false
     on(put(dt,f1),f2) = if f1=f2 then true else on(dt,f2)
     get(put(dt,f1),f2) = if f1=f2 then dt else put(get(dt,f2),f1)
```

Now, the technique is applied to specify the corresponding target domain in Apple's DESKTOP metaphor. The sort and variable names for the electronic analogues of office objects are prefixed by "El" and "el", respectively.

```
ElDesktop
Sorts     ElDesktop, ElFolder, Bool
Ops  new: → ElDesktop
     put: ElDesktop x ElFolder → ElDesktop
     get: ElDesktop x ElFolder → ElDesktop
     on:  ElDesktop x ElFolder → Bool
Eqs  on(new,elf) = false
     on(put(eldt,elf1),elf2) = if elf1=elf2 then true else on(eldt,elf2)
     get(put(eldt,elf1),elf2) = if elf1=elf2 then eldt else
          put(get(eldt,elf2),elf1)
```

Obviously, these two specifications are identical up to sort and variable names. In practice, this means that Macintosh desktops behave like real desktops with respect to the operations defined. For the sake of simplicity, only the most salient features have been specified. By including additional operators, dissimilarities could be shown, such as the fact that things can fall from physical, but not from electronic desktops. In an actual interface design process, the designer has to decide which features of a source domain are to be considered salient and which are not.

these are to be
[illegible]

Next, clipboards are specified. First the physical variety, with the assumption that clips are added to and removed from the top. Note that this specification resembles that of desktops. We will come back to this parallel when discussing image-schemas in 4.4.

```
Clipboard
Sorts      Board, Clip, Bool
Ops  new:  → Board
     put:  Board x Clip → Board
     get:  Board → Board
     on:   Board x Clip → Bool
Eqs  on(new,c) = false
     on(put(b,c1),c2) = if c1=c2 then true else on(b,c2)
     get(put(b,c)) = b
```

Secondly, the electronic variety of clipboards, as provided by the Macintosh to move data between documents, is specified:

```
ElClipboard
Sorts      ElBoard, ElClip, Bool
Ops  new:  → ElBoard
     put:  ElBoard x ElClip → ElBoard
     get:  ElBoard → ElBoard
     on:   ElBoard x ElClip → Bool
Eqs  on(new,elc) = false
     on(put(elb,elc1),elc2) = if elc1=elc2 then true else false
     get(put(elb,elc)) = put(elb,elc)
```

The axioms show two differences between physical and electronic clipboards:

- a clip remains on the Macintosh clipboard only until the next clip is put on - then it is lost;
- getting back a clip does not change the electronic clipboard and the same clip can be retrieved multiple times.

While the first mismatch may be harmful, the second is rather an advantage. The point is, however, that the specification *reveals* these mismatches. In an actual interface design process, it is the designer's task to decide about their desirability [Halasz and Moran 1981, Johnson 1987a].

Folders are another part of the DESKTOP metaphor. Here is a specification of the salient features of physical office folders:

```
Folder
Sorts          Folder, Document, Name, Bool
Ops  new:      → Folder
     label:    Folder x Name → Folder
     insert:   Folder x Document → Folder
     remove:   Folder x Document → Folder
     empty:    Folder → Folder
     in:       Folder x Document → Bool
Eqs  remove(label(f,n),d)  =  label(remove(f,d),n)
     remove(insert(f,d1),d2)  =  If d1=d2 then f else
               insert(remove(f,d2),d1)
     empty(new) = new
     empty(label(f,n))  =  label(empty(f),n)
     empty(insert(f,d))  =  empty(f)
     in(new,d) = false
     in(label(f,n),d)  =  in(f,d)
     in(insert(f,d1),d2)  =  If d1=d2 then true else  in(f,d2)
```

The following specification of electronic folders is very similar. The only difference is that the empty operator is missing because the Macintosh does not offer such an operation. (It does, of course, allow users to select all documents in a folder and remove them or to discard an entire folder, but these are different operations.)

```
ElFolder
Sorts          ElFolder, ElDocument, Name, Bool
Ops  new:      → ElFolder
     label:    ElFolder x Name → ElFolder
     insert:   ElFolder x ElDocument → ElFolder
     remove:   ElFolder x ElDocument → ElFolder
     in:       ElFolder x ElDocument → Bool
Eqs  remove(label(elf,n),eld)  =  label(remove(elf,eld),n)
     remove(insert(elf,eld1),eld2)  =  If eld1=eld2 then elf else
               insert(remove(elf,eld2),eld1)
     in(new,eld)  =  false
     in(label(elf,n),eld)  =  in(elf,eld)
     in(insert(elf,eld1),eld2)  =  If eld1=eld2 then true else  in(elf,eld2)
```

The final pair of DESKTOP metaphor domains to be specified here is that of trash cans. Some operators (e.g., the possibility to discard folders) have been omitted for brevity:

```
Trash Can
Sorts          Trash, Document, Bool
Ops  new:      → Trash
     discard:  Trash x Document → Trash
     remove:   Trash x Document → Trash
     empty:    Trash → Trash
     in:       Trash x Document → Bool
Eqs  remove(discard(t,d1),d2) = if d1=d2 then t else
     discard(remove(t,d2),d1)
     empty(new) = new
     empty(discard(t,d)) = new
     in(new,d) = false
     in(discard(t,d1),d2) = if d1=d2 then true else in(t,d2)
```

The specification of the target domain is similar to that of the source domain; however, it also reveals a mismatch:

```
ElTrash
Sorts          ElTrash, ElDoc, Bool
Ops  new:      → ElTrash
     discard:  ElTrash x ElDoc → ElTrash
     remove:   ElTrash x ElDoc → ElTrash
     empty:    ElTrash → ElTrash
     in:       ElTrash x ElDoc → Bool
Eqs  remove(discard(et,ed1),ed2) = if ed1=ed2 then et else
               discard(remove(et,ed2),ed1)
     empty(new) = new
     empty(discard(et,ed)) = new
     in(new,ed) = false
     in(discard(et,ed1),ed2) = true or false
```

The difference in the last equation captures the fact that a Macintosh user does not have complete control over the Trash can contents. A document can be retrieved immediately after discarding it, but may be lost after another operation, such as opening a document or inserting a disk.

Each specification of a source or target domain given in this section describes a class of algebras. The *designer's* conceptual model of a domain is seen as an algebra in this class, i.e., an interpretation of the corresponding specification which assigns symbols to the sorts and operators such that the axioms hold. An important challenge in implementing an interface metaphor is to make sure that the *user's* conceptual model belongs to the same class.

4.3. METAPHOR MAPPINGS

Metaphors are mappings from source to target domains. When the domains are specified algebraically, it is natural to use morphisms, which are mappings between algebras, to define metaphorical mappings.

There are different kinds of morphisms, reflecting different degrees of similarities between algebras. The weakest kind of similarity is a *signature-morphism* which is a correspondence between the signatures defining two algebras [Ehrich, Gogolla, and Lipeck 1989]. A stronger kind is a *homomorphism* between two algebras A and B. It is a family of mappings from the sets of A to those of B, such that it does not matter whether an operator is first applied in A and the result mapped to B, or the arguments are first mapped to B and the corresponding operator in B is applied. Thus, homomorphisms are structure-preserving mappings. A one-to-one homomorphism is called an *isomorphism.*

If the source and target domain are both specified, as in the above examples, the morphism is determined. It indicates the degree of similarity between source and target for the specified (and, in this case, implemented) interface metaphors:

- The mapping between office and electronic *desktops* is an isomorphism, since the effects of the specified operators on office desktops correspond to the effects of the analogous operators on electronic desktops, and vice versa.
- The mapping from physical to electronic *clipboards* is a signature-morphism. The signatures are the same, but the effects of some of the operators are different. For example, the signatures of the physical put and get operators map to those of the electronic ones, but the effects of applying get after put are different, as can be seen from the third axioms in the two specifications.
- The *folder* specifications exemplify another isomorphism, but only with respect to the new, label, insert, remove, and in operators; the empty operator has no equivalent in the target domain.
- The *trash can* specifications are linked by another signature-morphism. The signatures correspond, but the result of the in operator after discarding a document can be different for the two domains.

In the normal course of an interface design, the target domains are not predetermined. Source domains are specified first and appropriate morphisms are chosen to achieve the intended matches. These morphisms then determine parts of the target domain by projecting the desired elements of the source structure.

4.4. IMAGE-SCHEMAS

The algebraic specifications of source and target domains from the DESKTOP metaphor exhibit two image-schemas: Desktops and clipboards are instantiations of the SURFACE schema; folders and trash cans represent the CONTAINER schema. The SURFACE schema embodies the logic of a surface on which items can be put. The CONTAINER schema consists of an interior, an exterior, and a boundary between them, so that items can be in a container or outside of it [Lakoff 1987].

Thus, not only are image-schematic structures believed to be invariant in metaphorical mappings, they can also be common to different parts of a composite metaphor. Composite metaphors whose domains are related to common image-schemas are more likely to be perceived as *coherent* than those where this is not the case. It can be argued [Erickson 1990b] that the success of the Macintosh DESKTOP metaphor is in part due to a largely consistent use of the

CONTAINER image-schema.

If an image-schema is invariant in a metaphorical mapping, it must be a common part of the source and target domains. It is therefore possible to obtain the algebraic specifications of these domains by extending a common core specification which formalizes an image-schema. Such a process of adding operators to algebraic specifications is called an *enrichment* [Ehrich, Gogolla, and Lipeck 1989]. Thus, a formal version of Lakoff's invariance hypothesis is:

For any metaphor, there is an algebraic specification which describes an image-schema or a combination of image-schemas and which can be enriched toward specifications of the source and target domains.

Since an algebraic specification describes a class of algebras or category, image-schemas are formalized as categories. These categories contain the algebras of the source and target domains as well as the morphisms between them.

The following is an algebraic specification of the SURFACE image-schema. It can easily be enriched toward the above specifications of the physical and electronic desktops by renaming the sorts (Surface → Desktop, Item → Folder, etc.) and adding the get operator with its axiom. The same can be said for the physical, but not for the electronic clipboard: The fact that a previously clipped item is lost when a new item is put on is not only a deviation from the behavior of physical clipboards, but also from the logic of the SURFACE schema.

```
Surface
Sorts      Surface, Item, Bool
Ops  new:  → Surface
     put:  Surface x Item → Surface
     on:   Surface x Item → Bool
Eqs  on(new,i) = false
     on(put(s,i1),i2) = if i1=i2 then true else on(s,i2)
```

The other image-schema dominating the DESKTOP metaphor is the CONTAINER. Its specification happens to be isomorphic to that of the SURFACE schema. This correspondence may, in fact, contribute to the coherence of the DESKTOP metaphor. From a cognitive perspective, however, two different kinds of experiences are described. The CONTAINER schema can be specified as follows (see also [Guttag, Horning, and Wing 1985]):

```
Container
Sorts          Container, Item, Bool
Ops  new:      → Container
     insert:   Container x Item → Container
     in:       Container x Item → Bool
Eqs  in(new,i) = false
     in(insert(c,i1),i2) = if i1=i2 then true else in(c,i2)
```

This specification can again be enriched toward the previously given specifications of physical and electronic folders as well as physical trash cans. Thus, all these objects behave the way one intuitively expects from containers. No enrichment is possible, however, for electronic trash cans:

The second axiom of the CONTAINER specification is violated by those cases where a discarded document is not in the trash without having been removed. Given how deeply rooted image-schemas are in human cognition, such mismatches with image-schemas are more likely to be harmful than source-target differences which are above the image-schematic level.

5. A Spatial Application

5.1. SPATIALIZED AND SPATIAL DOMAINS

The DESKTOP metaphor affords a spatialization for an abstract, non-spatial application domain: The source concepts of the physical office impose a spatial structure on the abstract target domain of information processing. A wide range of other spatializing interface metaphors provides evidence for the crucial role of spatialization in human-computer interaction. Examples include the NAVIGATION [Conklin 1987], TOURIST [Fairchild, Meredith, and Wexelblatt 1989], ROOM [Henderson and Card 1986], and MUSEUM [Travers 1989] metaphors. Indeed, the two- or three-dimensional nature of visual interaction itself implies a spatialization for visual interfaces [Tauber 1987] in general.

[illegible]

At the same time, cognitive science has established that "most of our fundamental concepts are organized in terms of one or more spatialization metaphors" [Lakoff and Johnson 1980] and that many image-schemas are spatial in nature. These findings suggest, too, that spatialization is probably the rule rather than the exception for interface metaphors.

When dealing with geographic space, however, interface metaphors for inherently spatial applications are of primary interest. An example of such an interface metaphor is furnished by zoom and pan operations in Geographic Information Systems (GIS) and other graphics applications.

The terms "zoom" and "pan" are used in computer graphics in analogy to manipulations of cameras or other optical instruments. An investigation of zoom and pan interface designs for GIS concluded, however, that intuitive and effective interface tools require a deeper understanding of the metaphors involved [Jackson 1990]. The remainder of this section applies the formal tools developed in the previous section to an analysis of the metaphorical and image-schematic structure of zooming.

5.2. ZOOMING AND THE "DISPLAYS ARE VIEWS" METAPHOR

The Oxford English Dictionary (second edition, 1989) defines the original meaning of "zoom" as follows:

"To make a continuous low-pitched humming or buzzing sound;
to travel or move (as if) with a 'zooming' sound;
to move at speed, to hurry."

The use of the term in photography and cinematography is, thus, already doubly metaphorical: It explains the variation of the focal length by a (fictive) motion of rapidly closing up on a subject which, in turn, is metaphorically related to the corresponding sound effect. (It is an interesting twist that one of the key metaphors in visual interfaces is originally echoic, i.e. rooted in auditory perception.)

Combined with our deeply rooted visual experience that the viewing distance determines *what* we see, zooming acquires the interpretation of changing the level of detail at which we perceive and conceptualize the world or a computer model.

This understanding of zooming suggests the more general metaphor DISPLAYS ARE VIEWS, where views are understood as visual fields, containing what humans see in a given situation [Kuhn 1990]. This metaphor accommodates various transformations of the visual field in a common framework: Zooming is the transformation caused by changing the viewing distance; panning is moving the view to another part of a "panorama" without changing the level of perceived detail. Since transformations of the visual field correspond to "basic cognitive processes such as focusing, scanning, superimposition, figure-ground shifting, vantage-point shifting" [Lakoff 1988], they are ideal candidates for metaphor sources.

5.3. IMAGE-SCHEMAS IN "DISPLAYS ARE VIEWS"

The basic image-schema involved in the visual field is the CONTAINER schema. It structures the visual field as a bounded space, consisting of a boundary, an interior, and an exterior: Things are either in or out of sight and they come into or go out of it [Lakoff 1987]. The visual field has also a center of attention and a surrounding region. Thus, a CENTER-PERIPHERY schema is combined with the CONTAINER schema, providing for the distinction of foveal and peripheral vision.

Enriching the previously specified CONTAINER schema by a center-periphery distinction produces the following specification (at this point, no axioms are given to constrain the inCenter operator):

```
Container with Center
Sorts               CentCont, Item, Bool
Ops   new:          → CentCont
      insert:       CentCont x Item → CentCont
      in:           CentCont x Item → Bool
      inCenter:     CentCont x Item → Bool
Eqs   in(new,i) = false
      in(insert(cc,i1),i2) = if i1=i2 then true else in(cc,i2)
```

The visual field is further characterized by an interaction of the PART-WHOLE with the NEAR-FAR schema. We experience objects in the world as configurations of parts, forming wholes. Our perception has evolved so that it can distinguish parts and wholes. Unaided visual perception, in particular, relies on motion of the body or of the objects to extend this distinction beyond the limited range of configurations present in one view: Getting parts into view involves moving nearer and vice versa. This connection is the essence of zooming.

To formalize the superimposition of the PART-WHOLE and NEAR-FAR schemas on top of the centered CONTAINER, the above specification is further enriched by a closeUp operator and a part-whole relation (which actually belongs to a separate Item specification). Let us call this multiply enriched container a View Container. With these additional operators, it becomes possible to capture the meaning of zooming in one axiom for the closeUp operator:

```
View Container
Sorts        ViewCont, Item, Bool
Ops  new:      → ViewCont
     insert:   ViewCont x Item → ViewCont
     in:       ViewCont x Item → Bool
     inCenter: ViewCont x Item → Bool
     partOf:   Item x Item → Bool
     closeUp:  ViewCont → ViewCont
Eqs  in(new,i) = false
     in(insert(vc,i1),i2) = if i1=i2 then true else in(vc,i2)
     in(closeUp(vc),i) = inCenter(vc,i) or (inCenter(vc,i1) and partOf(i,i1))
```

Note that the closeUp operator is specified quite generally: It is only constrained to keep center items in view and get their parts into view; no details about the degree of zooming or the extent of the center region are given. More specific axioms would require metric concepts, while an image-schematic analysis is only concerned with topological properties.

Further enrichments with metric concepts would lead to complete source (view) and target (display) specifications for a particular user interface. They could involve, for example, a viewing distance in the source domain and a zoom factor or view extent in the target domain.

It should be emphasized that DISPLAYS ARE VIEWS is a metaphor and not a literal equivalence. The matching between the two domains of a metaphor is by definition partial. Aspects which are not image-schematic, such as the shape of the container boundary (elliptic, rectangular) or the type of optical projection involved (central, parallel), need not be equal or even comparable.

6. Conclusions

An algebraic approach to the formalization of interface metaphors has been presented. Algebraic specifications define metaphor domains as algebras, metaphorical mappings as morphisms, and invariant image-schemas as categories. The gist of the approach should be captured in the examples from a spatialization metaphor (DESKTOP). The sketched application to zoom operations should demonstrate the relevance to the design of user interfaces for systems dealing with spatial information.

The proposed formalization of interface metaphors is not a formal method for the design of metaphorical interfaces, but we believe it is a useful and immediately applicable step toward it. The principal strengths of the proposal are that:

- the underlying understanding of metaphors is in agreement with recent findings in cognitive science;
- the algebraic approach provides a comprehensive formalization of all essential aspects of metaphors: conceptual domains, structure-mappings, and image-schemas;
- the technique of algebraic specifications is well established in software design and supported by design tools.

The proposed formalization has yet to be validated in actual user interface designs. Our initial experiences from applications to Geographic Information Systems [Jackson 1990] and to an interface based on book metaphors [Volta, Cicogna, and Kuhn 1990] have been positive. The rigor imposed and the questions raised by the approach have clearly been beneficial in the design process.

Formal approaches to design are mostly motivated by the need to evaluate designs. Distinguishing different kinds of morphisms and separating the image-schematic structure from surface-level structure are solid bases for such evaluations, but much more theoretical and experimental work needs to be done in this direction. Evaluating interface metaphors also involves dealing with the coherence of composite metaphors. Since image-schemas are believed to play an important role for coherence, their formalization is a key feature of this approach.

Some aspects of our work may have implications beyond human-computer interaction. The description of metaphorical domains by algebraic specifications, for instance, is an attempt at formalizing the notion of Idealized Cognitive Models (ICM) [Lakoff 1987]. It accommodates propositional and image-schematic structure as well as metaphoric mappings. Another general result is the formal version of the invariance hypothesis.

ACKNOWLEDGEMENT. Support from NSF under NCGIA grant No. SES 88-10917, from Digital Equipment Corporation under grant No. TP-765536, and from Intergraph Corporation is gratefully acknowledged. We also thank Max Egenhofer and Jeffrey Jackson for discussions on the topic.

References

Apple Computer, Inc. 1987. *Human Interface Guidelines: The Apple Desktop Interface.* Reading, MA: Addison-Wesley.

Blumenthal, B. 1990. Incorporating Metaphor in Automated Interface Design. *Proceedings Interact '90* : 659-664.

Burstall, R.M., and J.A. Goguen. 1981. *An Informal Introduction to Formal Specifications Using CLEAR.* In *The Correctness Problem in Computer Science.* Edited by R. S. Boyer and J. Moore. 185-213. New York: Academic Press.

Carroll, J.M., R.L. Mack, and W.A. Kellogg. 1988. *Interface Metaphors and User Interface Design.* In *Handbook of Human-Computer Interaction.* Edited by M. Helander. 67-85. North-Holland: Elsevier Science Publishers.

Conklin, J. 1987. Hypertext: An Introduction and Survey. *IEEE Computer* 20 (9) : 17-41.

Ehrich, H.-D., M. Gogolla, and U.W. Lipeck. 1989. *Algebraische Spezifikation abstrakter Datentypen.* Leitfäden und Monographien der Informatik. Stuttgart: B.G. Teubner.

Ehrig, H., and B. Mahr. 1985. *Fundamentals of Algebraic Specification.* Berlin: Springer-Verlag.

Erickson, T.D. 1990a. *Working with Interface Metaphors.* In *The Art of Human-Computer Interface Design.* Edited by B. Laurel. 65-73. Addison-Wesley.

Erickson, T.D. 1990b. Personal communication.

Fairchild, K., G. Meredith, and A. Wexelblatt. 1989. The Tourist Artificial Reality. *Proceedings ACM CHI'89* : 299-304.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7 : 155-170.

Gentner, D. 1989. *The mechanisms of analogical learning.* In *Similarity and Analogical Reasoning.* Edited by S. Vosniadou and A. Ortony. 199-241. Cambridge, UK: Cambridge University Press.

Goguen, J. A., J. W. Thatcher, and E. G. Wagner. 1978. *An Initial Algebraic Approach to the Specification, Correctness, and Implementation of Abstract Data Types*. In *Current Trends in Programming Methodology*. Edited by R. Yeh. 80-149. Englewood Cliffs, N.J.: Prentice-Hall.

Guttag, J.V., and J.J. Horning. 1978. The Algebraic Specification of Abstract Data Types. *Acta Informatica* 10 (1) : 27-52.

Guttag, J.V., J.J. Horning, and J.M. Wing. 1985. *Larch in Five Easy Pieces*. Digital Equipment Corporation, Systems Research Center.

Halasz, F., and T.P. Moran. 1981. Analogy Considered Harmful. *International Journal of Man-Machine Studies* 14 : 383-386.

Hall, R.P. 1989. Computational Approaches to Analogical Reasoning: A Comparative Analysis. *Artificial Intelligence* 39 : 39-120.

Henderson, D.A., Jr., and S.K. Card. 1986. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *ACM Transactions on Graphics* 5 (3) : 211-243.

Indurkhya, B. 1989. *Metaphor and Cognition*. Manuscript.

Jackson, J.P. 1990. *Visualization of Metaphors for Interaction With Geographic Information Systems*. Masters of Science Thesis, University of Maine, Department of Surveying Engineering.

Johnson, J. 1987a. How faithfully should the electronic office simulate the real one? *ACM SIGCHI Bulletin* 19 (2) : 21-25.

Johnson, M. 1981. *Metaphor in the Philosophical Tradition*. In *Philosophical Perspectives on Metaphor*. Edited by M. Johnson. 3-47. Minneapolis: University of Minnesota Press.

Johnson, M. 1987b. *The Body in the Mind*. Chicago: The University of Chicago Press.

Kuhn, W. 1990. *Are Displays Maps or Views?* Technical Report, Department of Surveying Engineering, University of Maine.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things*. Chicago: The University of Chicago Press.

Lakoff, G. 1988. *Cognitive Semantics*. In *Meaning and Mental Representations*. Edited by U. Eco, M. Santambrogio and P. Violi. 119-154. Bloomington: Indiana University Press.

Lakoff, G. 1990. The Invariance Hypothesis: is abstract reason based on image-schemas? *Cognitive Linguistics* 1 (1) : 39-74.

Lakoff, G., and M. Johnson. 1980. *Metaphors We Live By*. Chicago: The University of Chicago Press.

Liskov, B., and J. Guttag. 1986. *Abstraction and Specification in Program Development*. The MIT Electrical Engineering and Computer Science Series. Cambridge, MA: The MIT Press (MacGraw-Hill).

Mac Cormac, E.R. 1985. *A Cognitive Theory of Metaphor*. Cambridge, MA: The MIT Press.

MacLane, S., and G. Birkhoff. 1967. *Algebra*. New York: Macmillan.

Mark, D. 1989. Cognitive Image-Schemata for Geographic Information: Relations to User Views and GIS Interfaces. *Proceedings GIS/LIS'89* 2 : 551-560.

Martin, J.H. 1990. *A Computational Model of Metaphor Interpretation*. New York: Academic Press.

Nielsen, J. 1990. *Designing User Interfaces for International Use*. Amsterdam: Elsevier Science Publishers.

Ortony, A. 1979. *Metaphor and Thought*. Cambridge, England: Cambridge University Press.

Smith, D.C.S., C. Irby, R. Kimball, B. Verplank, and E. Harslam. 1982. Designing the Star

User Interface. *BYTE.* 7 (4) April 1982 : 242-282.

Tauber, M.J. 1987. *On Visual Interfaces and their Conceptual Analysis.* In *Visualization in Programming.* Edited by P. Gorny and M.J.Tauber. 106-123. Springer-Verlag.

Travers, M. 1989. A Visual Representation for Knowledge Structures. *Proceedings Hypertext '89* : 147-158.

Volta, G.S., R. Cicogna, and W. Kuhn. 1990. *The MaineBook.* Technical Report, Department of Surveying Engineering, University of Maine.

Vosniadou, S., and A. Ortony. 1989. *Similarity and Analogical Reasoning.* Cambridge: Cambridge University Press.

Woodcock, J., and M. Loomes. 1989. *Software Engineering Mathematics.* SEI Series in Software Engineering. Reading, MA: Addison-Wesley.

Wozny, L.A. 1989. The application of metaphor, analogy, and conceptual models in computer science. *Interacting with Computers* 1 (3) : 273-283.