Thinking Outside (and Inside) the Box

Michael Grüninger and Salim Bouafoud

Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
gruninger@mie.utoronto.ca

Abstract. In many problems in commonsense reasoning and intelligent manufacturing, we need to reason about cutting, joining, and folding sheets of materials such as cardboard and metal. In this paper we introduce a first-order ontology of shape (called BoxWorld) that can support these applications. We reuse an existing ontology of shape for object recognition with 2D shapes (called CardWorld) and extend the axioms to three-dimensional shapes in the BoxWorld Ontology. A distinguishing characteristic of these ontologies is that they use only the notions of incidence and betweenness rather than Euclidean geometry as axiomatized by Hilbert and Tarski.

Prelude

When Alice went to the kitchen for her breakfast cereal, she opened the box by removing the tab from the slot at the top of the box. Discovering it empty, she reached for a new box, which she opened by detaching one side of the lid from the other. She closed the box by opening the slot on one slide of the lid and then inserted the tab into the slot. After breakfast, she disassembled the old box and dropped it into the recycling bin.

1 Introduction

As can be seen from the tale in the Prelude, we need an ontology to describe shapes that are composed of surfaces, such as a box and its components. The ontology should also support the specification of the processes that are performed when using the box, which means that we need to represent not only the box, but the initial cardboard surface and all intermediate states as well.

Although shape representations arise in a number of areas, from computer vision to computational geometry ([1]), there exist few explicit axiomatizations of these approaches within first-order logic. There are semiformal descriptions of polyhedra ([2], [4]) as classes of abstract polytopes, but even if these were axiomatized, they would not be sufficient. One problem is that although the box is a polyhedron, the initial and intermediate states are not polyhedra. Another problem is that we need to be able to represent objects with holes (e.g. slots within the lid of a box), but the existing axiomatizations of abstract polytopes and polyhedra do not in general allow holes.

An ontology for shapes composed of surfaces also has applications in sheet metal fabrication, in which products are manufactured by folding, cutting, and joining pieces of sheet metal. A shape ontology for sheet metal processes would be able to answer questions such as:

– *What objects can we make from a single sheet of metal, either by cutting or folding?*
– *What objects can we make from joining multiple surfaces?*

In the remainder of the paper, we present two closely related first-order ontologies for surfaces and boxes. For both ontologies, we ask the question of how far we can go with only a notion of incidence and ordering, that is, without using full geometry as axiomatized by Tarski or Hilbert.

2 CardWorld

The first theory we discuss is based on the CardWorld Ontology, which is a first-order ontology for 2D-object recognition in scenes with occlusion and images with noise. This ontology was also used in [3] as part of a cutting process ontology for sheet metal manufacturing.

In the axiomatization of the CardWorld Ontology (Section 4.1) we focus on the mereotopological relations (i.e. parthood and connection) rather than geometric relations (such as relative alignment and length of segments, or the notions of curvature or surface area). This also means that there is no notion of a "straight" edge; all we know is that a surface is composed of edges, and that each edge is composed of points.

We can decompose the axioms of the CardWorld Ontology into five modules – *cardworld_part*, *cardworld_edge*, *cardworld_outer*, *cardworld_surface*, and *cardworld*. The relationships between these theories is shown in Figure 2.

2.1 Basic Ontological Commitments

The root theory within the CardWorld Ontology is *cardword_part*:

```
(cl-text cardworld-part

(forall (x)
        (if     (point x) (and (not (edge x)) (not (surface x)))))

(forall (x)
        (if     (edge x) (not (surface x))))

(forall (x y)
        (if     (part x y) (part y x)))
```

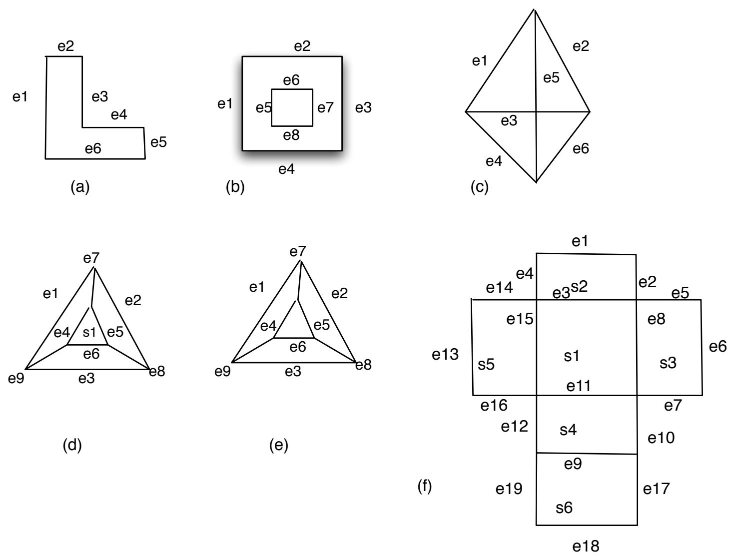

Fig. 1. Shapes in CardWorld and BoxWorld.

```
(forall (x y)
        (part x x))

(forall (x y)
        (if     (and    (part x y) (point x) (point y))
                (= x y)))

(forall (x y)
        (if     (and    (part x y) (edge x) (edge y))
                (= x y)))

(forall (x y)
        (if     (and    (part x y) (surface x) (surface y))
                (= x y)))
)
```

Within models of this theory, there are three (disjoint) sorts of objects in the domain – surfaces, edges, and points. The axioms of this module guarantee that *part* relation forms a weak tripartite incidence structure over these objects.

Figure 1(a) and 1(b) show two examples of surfaces; intuitively, the second surface has a hole. We emphasize that there are no geometric notions being used in CardWorld; there is no concept of a space into which the points, edges, and surfaces are embedded. We do not speak about edges being straight or surfaces being flat. Within a particular application domain, it is necessary to identify the surfaces, and the edges intuitively constitute the boundaries of surfaces.

2.2 Edges and Surfaces

From the root theory, we know that every point is part of some edge and every edge is part of a surface; the axiom in $cardworld\_surface$ strengthens this condition so that every point and every edge is part of a unique surface:

```
(cl-text cardworld-surface

(cl-imports cardworld-edge)

(cl-comment "Each point and edge are part of a unique surface.")
(forall (x s1 s2)
        (if     (and    (surface s1) (surface s2) (part x s1) (part x s2)
                        (or (point x) (edge x)))
                (= s1 s2)))
)
```

The intuition here is that with CardWorld we are restricting our attention to two-dimensional objects; if we consider edges to be one-dimensional objects, then the objects that contain edges are the two-dimensioanl objects, which are the surfaces. An object that contains multiple connected surfaces (which have edges in common) would be three-dimensional, and such objects are the focus of the BoxWorld Ontology.

A vertex is a point that is part of two edges; every edge contains exactly two vertices. Every edge in a surface meets another distinct edge in that surface; consequently, every surface contains at least two edges. Since each edge meets exactly two other edges (one at each vertex), the set of edges in a surface form cycles within models of *cardworld_edge*:

```
(cl-text cardworld-edge

(cl-imports cardworld-part)

(cl-comment "Each point is part of some edge")
(forall (p)
        (if     (point p)
                (exists (e)
                        (and    (edge e) (part p e)))))

(cl-comment "Every edge contains at least two vertices.")
(forall (e1)
        (if     (edge e1)
                (exists (e2 e3 v1 v2)
                        (and    (meet e1 e2 v1) (meet e1 e3 v2)
                                (not (= e1 e2)) (not (= v1 v2))))))
```

```
(cl-comment "An edge contains at most two vertices.")
(forall (e v1 v2 v3)
        (if     (and    (edge e) (vertex v1) (vertex v2) (vertex v3)
                        (part v1 e) (part v2 e) (part v3 e))
                (or     (= v1 v2) (= v1 v3) (= v2 v3))))

(cl-comment "Each edge is part of some surface.")
(forall (e)
        (if     (edge e)
                (exists (s)
                        (and    (surface s) (part e s)))))

(cl-comment "Each surface contains an edge.")
(forall (s)
        (if     (surface s)
                (exists (e)
                        (and    (edge e) (part e s)))))

(forall (e1 e2 v)
        (iff    (meet e1 e2 v)
                (and    (edge e1) (edge e2) (point v) (not (= e1 e2))
                        (part v e1) (part v e2))))

(forall (v)
        (iff    (vertex v)
                (exists (e1 e2)
                        (meet e1 e2 v))))
)
```

2.3 Holes

The edges in a surface are partitioned into disjoint cycles; intuitively, one of these cycles contains the outer edges of the surface, while the remaining cycles of edges constitute holes within the surface. The axioms of *cardworld_outer* guarantee that there exists a unique set of outer edges that are all elements of the same cycle.

```
(cl-text cardworld-outer

(cl-imports cardworld-edge)

(cl-comment "Outer edges are edges.")
(forall (e)
        (if     (outer e) (edge e)))
```

```
(cl-comment "Every surface contains an outer edge.")
(forall (s)
        (if     (surface s)
                (exists (e)
                        (and    (outer e) (part e s)))))

(cl-comment "All outer edges in the same surface are comparable by
the sbetween relation.")
(forall (e1 e2 s)
        (if     (and    (outer e1) (surface s) (edge e2)
                        (part e1 s) (part e2 s))
                (iff    (outer e2) (sbetween e1 e2 s))))

(cl-comment "The sbetween relation is a cyclic ordering on the set
of edges in a surface.")

(forall (e1 e2 e3)
        (if     (sbetween e1 e2 e3) (sbetween e2 e3 e1)))

(forall (e1 e2 e3)
        (if     (sbetween e1 e2 e3)
                (not (sbetween e3 e2 e1))))

(forall (e1 e2 e3 e4)
        (if     (and    (sbetween e1 e2 e3) (sbetween e1 e3 e4))
                (sbetween e1 e2 e4)))

(forall (e1 e2 e3)
        (if     (sbetween e1 e2 e3)
                (or     (exists (e4) (sbetween e1 e4 e2) (not (= e4 e2))))
                        (exists (v) (meet e1 e2 v)))))

(forall (e1 e2 e3)
        (if     (sbetween e1 e2 e3)
                (exists (s)
                        (and    (surface s) (edge e1) (edge e2) (edge e3)
                                (part e1 s) (part e2 s) (part e3 s)))))
)
```

The surface in Figure 1(b) has a hole; the edges e_1, e_2, e_3 are the outer edges of the surface, while the edges e_4, e_5, e_6 are not outer edges, but rather belong to the hole in the surface.

2.4 Closure in BoxWorld

The sole new axiom in *cardworld* extends the other theories to say that surfaces, edges, and points are the only objects in the domain:

```
(cl-text cardworld

(cl-imports cardworld-outer)
(cl-imports cardworld-surface)

(cl-comment "All objects are either points, edges, or surfaces.")
(forall (x)
        (or     (point x) (edge x) (surface x)))
)
```

3 BoxWorld

The extension of CardWorld to BoxWorld was originally motivated by the problem of manufacturing products by folding sheet metal. Initially one starts with a single sheet (which can be represented as a surface using CardWorld). However, once one folds this sheet, one constructs a shape that consists of two surfaces that have a common edge, which violates an axiom of CardWorld. We therefore needed an extension of CardWorld that was able to represent shapes that are composed of multiple surfaces. The axioms of BoxWorld are decomposed into six modules – *boxworld_part*, *boxworld_edge*, *boxworld_peak*, *boxworld_border*, *boxworld_surface*, and *boxworld*. The relationships between these theories and the CardWorld Ontology is shown in Figure 2.

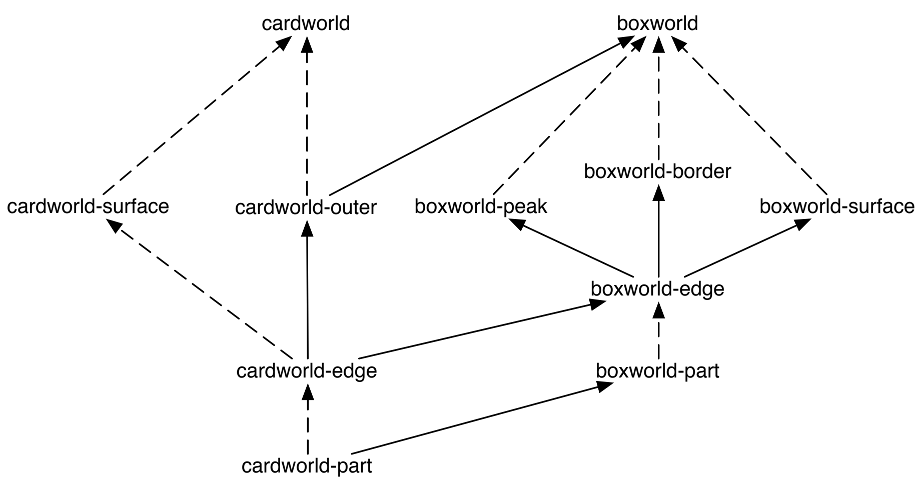

Fig. 2. Relationships between the modules in the CardWorld and BoxWorld Ontologies. Solid lines denote conservative extension and dashed lines denote nonconserative extension.

3.1 Basic Ontological Commitments

The module *boxworld_part* extends *cardworld_part* with the new class of boxes, which are composed of surfaces:

```
(cl-text boxworld-part

(cl-imports cardworld-part)

(forall (x)
        (if     (or (point x) (edge x) (surface x))
                (not (box x))))

(forall (x y)
        (if     (and    (part x y) (box x) (box y))
                (= x y)))
)
```

Every point is part of an edge, every edge is part of a surface, and every surface is part of a box.

The shapes in Figure 1(c),(d), and (e) are possible shapes in models of the BoxWorld Ontology. In particular, Figure 1(f) is a net, which is an arrangement of polygons joined at their edges and which can be folded along these edges to become the faces of a polyhedron; in this example, we have the net for a cube.

3.2 Edges and Surfaces

The module $boxworld\_surface$ guarantees that each point, edge, or surface is contained in a unique box:

```
(cl-text boxworld-surface

(cl-imports boxworld-edge)

(cl-comment "Each point, edge, and surface is part of a unique box.")
(forall (x1 x2 y)
        (if     (and    (part y x1) (part y x2) (box x1) (box x2)
                        (or (point y) (edge y) (surface y)))
                (= x1 x2)))
)
```

For any nontrivial box (that is, a box with multiple distinct surfaces), each surface will contain edges (known as ridges) that are part of exactly two surfaces. In a polyhedron, every edge is a ridge. For example, Figure 1(c) is a tetrahedron, that is, it contains four surfaces. The surface s_1 contains the edges e_1, e_2, e_3; the surface s_2 contains the edges e_2, e_5, e_6; the surface s_3 contains the edges e_1, e_4, e_5; the surface s_4 contains the edges e_3, e_4, e_6.

Although polyhedra are models of the BoxWorld axioms, there are also models that are not polyhedra; in such models, there exist edges that are parts of unique surfaces. An edge that is part of a unique surface is a border.

```
(cl-text boxworld-edge

(cl-imports boxworld-part)
(cl-imports cardworld-edge)

(cl-comment "An edge is part of at most two surfaces.")
(forall (e s1 s2 s3)
        (if     (and    (edge e) (surface s1) (surface s2) (surface s3)
                        (part e s1) (part e s2) (part e s3))
                (or     (= s1 s2) (= s2 s3) (= s1 s3))))

(cl-comment "A surface that is part of a box containing other surfaces also
contains a ridge.")
(forall (x s1 s2)        (if     (and    (box x) (surface s1) (surface s2) (not (= s1 s2))
                        (part s1 x) (part s2 x))
                (exists (e)
                        (and    (ridge e) (part e s1)))))

(cl-comment "Every edge in a surface meets another distinct edge in that surface
."
(forall (e1 s v)
        (if     (and    (edge e1) (part e1 s) (surface s) (vertex v) (part v e1))
                (exists (e2)
                        (and    (edge e2) (part e2 s) (meet e1 e2 v)))))
)

(cl-comment "If three distinct edges meet at the same vertex, then they
cannot all be part of the same surface.")

(forall (e1 e2 e3 v s)
       (if     (and     (edge e1) (edge e2) (edge e3)
                        (not (= e1 e2)) (not (= e1 e3)) (not (= e2 e3))
                        (vertex v) (part e1 s) (part e2 s) (part v e2)
                        (meet e1 e2 v) (meet e1 e3 v))
               (not (part e3 s))))

(cl-comment "Every border edge meets two distinct border edges.")
(forall (e1)
        (if     (border e1)
                (exists (e2 e3 v1 v2)
                        (and    (border e2) (border e3)
                                (not (= e2 e3)) (not (= e1 e3))
                                (meet e1 e2 v1) (meet e1 e3 v2)))))

(cl-comment "Every border meets another unique border at a vertex.")
(forall (e1 e2 e3 v)
        (if     (and     (border e1) (border e2) (not (= e1 e2))
                         (meet e1 e2 v) (meet e1 e3 v)
                         (not (= e1 e3)) (not (= e2 e3)))
                (ridge e3)))
```

```
(cl-comment "A ridge is an edge that is part of two surfaces.")
(forall (e)
        (iff    (ridge e)
                (exists (s1 s2)
                        (and    (edge e) (surface s1) (surface s2) (not (= s1 s2))
                                (part e s1) (part e s2)))))

(cl-comment "A border is an edge that is part of a unique surface.")
(forall (e)
        (iff    (border e)
                (and    (edge e) (not (ridge e)))))
)
```

For example, consider the box in Figure 1(d), which intuitively is a triangular bowl in which the surface s_1 is the bottom; the border edges in this case are e_1, e_2, e_3. In Figure 1(e) on the other hand, we intuitively have an open strip, and the edges e_1, e_2, e_3, e_4, e_5, e_6 are border edges.

The axioms of the module *boxworld_border* requires each border edge to meet two distinct border edges. In other words, if border edges exist, then they form cycles:

```
(cl-text boxworld-border

(cl-imports boxworld-edge)

(forall (e1 e2 e3)
        (if     (bbetween e1 e2 e3) (bbetween e2 e3 e1)))

(forall (e1 e2 e3)
        (if     (bbetween e1 e2 e3)))
                (not (bbetween e3 e2 e1))))

(forall (e1 e2 e3 e4)
        (if     (and    (bbetween e1 e2 e3) (bbetween e1 e3 e4))
                (bbetween e1 e2 e4)))

(forall (e1 e2 e3)
        (if     (bbetween e1 e2 e3)
                (exists (x)
                        (and    (box x) (border e1) (border e2) (border e3)
                                (part e1 x) (part e2 x) (part e3 x)))))

(forall (x e1 e2 e3)
        (if     (and    (box x) (border e1) (border e2) (border e3)
                        (not (= e1 e2)) (not (= e1 e3)) (not (= e2 e3))
                        (part e1 x) (part e2 x) (part v3 x))
                (or     (bbetween e1 e2 e3) (bbetween e2 e3 e1) (bbetween e3 e1 e2))))
)
```

3.3 Relationships between Edges

In order to ensure that polyhedra are models of the axioms, we need to explicitly axiomatize the cell embedding of a graph in a topological surface [5]. This achieved by imposing a cyclic ordering over the set of edges that are coincident with each vertex within the box; the module *cardworld_peak* axiomatizes these conditions:

```
(cl-text boxworld-peak

(cl-imports boxworld-edge)

(forall (e1 e2 e3)
        (if     (rbetween e1 e2 e3) (rbetween e2 e3 e1)))

(forall (e1 e2 e3)
        (if     (rbetween e1 e2 e3)))
                (not (rbetween e3 e2 e1))))

(forall (e1 e2 e3 e4)
        (if     (and    (rbetween e1 e2 e3) (rbetween e1 e3 e4))
                (rbetween e1 e2 e4)))

(forall (e1 e2 e3)
        (if     (rbetween e1 e2 e3)
                (exists (v)
                        (and    (vertex v) (edge e1) (edge e2) (edge e3)
                                (part v e1) (part v e2) (part v e3)))))

(forall (v e1 e2 e3)
        (if     (and    (vertex v) (edge e1) (edge e2) (edge e3)
                        (not (= e1 e2)) (not (= e1 e3)) (not (= e2 e3))
                        (part v e1) (part v e2) (part e v3))
                (or     (rbetween e1 e2 e3) (rbetween e2 e3 e1) (rbetween e3 e1 e2))))

(forall (e1 e2 e3 e4)
        (if     (and    (border e1) (border e2) (ridge e3) (ridge e4)
                        (rbetween e1 e3 e2))
                (not (rbetween e2 e4 e1))))
)
```

3.4 Closure in BoxWorld

The sole axiom in the module *boxworld* is a closure axiom, so that all objects are either points, edges, surfaces, or boxes.

```
(cl-text boxworld

(c-imports cardworld-outer)
(cl-imports boxworld-border)
(cl-imports boxworld-peak)
(cl-imports boxworld-vertex)
(cl-imports boxworld-box)

(forall (x)
        (or     (point x) (edge x) (surface x) (box x)))
)
```

Interlude: Breakfast with Alice

As Alice opens the cereal box by removing the tab from the slot at the top of the box, the first thing she notices is that the slot is a hole in one of the surfaces of the box, so that the edges that form the slot are not outer edges. The six edges that are part of the slot form a cycle, and they do not meet any other edges in the box. All of the other edges in the box are outer edges.

Although she is disappointed that the box is empty, Alice is intrigued when it dawns on her that the opened box is not a polyhedron, since there are edges in the box that belong to unique surfaces. In fact, there are four surfaces that contain border edges – the two surfaces that fold over for the lid of the box, and the two surfaces that are fold down at the side. Even with the growing hunger pangs, she is somehow delighted to realize that the border edges in these surfaces are connected and form a cycle within the set of all edges of the box.

The unopened box is a polyhedron, since all edges are shared by exactly two surfaces. Detaching one side of the lid creates a surface containing a border edge. Opening a slot on the other side of the lid creates a hole – whereas the box previously had only outer edges, it now has a set of non-outer edges.

Her breakfast now complete, Alice is content.

4 Consistency of the CardWorld and BoxWorld Ontologies

We have used Mace4 to construct nontrivial[1] models of both the CardWorld and BoxWorld Ontologies, thereby establishing their consistency. One of the simplest models that has been constructed is known as a hosodedron, which is a box with two surfaces and two shared edges.

The question remains, though, as to exactly which shapes are models of these ontologies. In particular, are all intuitive shapes also models of the axioms?

[1] A nontrivial model is one in which all relations have a nonempty extension.

Are there models of the axioms which do not correspond to intuitive shapes? Ulimately, these questions can only be answered by representation theorems for the ontologies; in the meantime, however, we can give some preliminary results by looking at models generated and checked by Mace4.

By specifying partial conditions (e.g. sentences entailing the existence of a set of surfaces, edges, and points), Mace4 was able to construct models of the CardWorld and BoxWorld axioms that correspond to the shapes in Figure 1(a)-(e). Unfortunately, Mace4 has been inadequate for exploring the set of all models of the CardWorld and BoxWorld Ontologies, in the search for potentially unintended models. Even if we restrict our attention to small domains (e.g. a box with five surfaces, twelve edges, and eight vertices), Mace4 is unable to find models. We will be pursuing the use of other model constructors such as Paradox to see whether they are able to construct models on different domains.

5 Summary

Reasoning about shapes composed of surfaces plays a critical role in problems ranging from commonsense reasoning to knowledge-based manufacturing. We have introduced two first-order theories – CardWorld for two-dimensional shapes consisting of unique surfaces, and BoxWorld for three-dimensional shapes consisting of multiple surfaces. In future work, we will prove the Representation Theorems for these theories.

References

1. Camossi, Elena; Giannini, Franca; Monti, Marina Deriving Functionality from 3D Shapes: Ontology Driven Annotation and Retrieval *Computer-Aided Design and Applications* 4:773-782.
2. Grünbaum, B. (2003) Are your polyhedra the same as my polyhedra?, em Discrete and Computational Geometry: the Goodman-Pollack Festschrift, ed. Aronov et al. pp. 461-488.
3. Grüninger, M. and Delaval, A. (2009) A First-Order Cutting Process Ontology for Sheet Metal Parts, *Fifth Conference on Formal Ontology Meets Industry*, Vicenza, Italy.
4. McMullen, Peter; Schulte, Egon (2002) *Abstract Regular Polytopes.* Cambridge University Press.
5. Mohar, B. (1999) A linear time algorithm for embedding graphs in an arbitrary surface, *SIAM Journal of Discrete Mathematics* 12:6-26.