Geometry in Highly Structured Design Spaces

Teng-Wen Chang[1] and Robert F. Woodbury[2]
[1]National Chiao Tung University, Taiwan
[2]Simon Fraser University, Canada
[1]http://www.arch.ntcu.edu.tw/tengwen/ - tengwen@arch.nctu.edu.tw
[2]http://www.surrey.sfu.ca/ - rw@sfu.ca

The Australian branch of the SEED project created a new formalism for design spaces in which the fundamental structuring operator is information specificity, formally characterised as subsumption. Here design space navigation is composed as combinations of the primitive operators of resolution, unification, anti-unification, search, query and hysterical undo. The structures needed to support such a view are highly constrained in a mathematical sense and it is in these constraints that the problems for representation of geometry arise. The research challenge is to add the formal design space exploration constraints into an existing geometric representation scheme or alternatively to discover a new scheme in which the constraints are realized.
Based on Typed Feature Structures (TFS), Geometric Typed Feature Structures (GTFS) are a representation scheme and method for performing the basic design space exploration operations on geometric objects. The crucial insight behind extending TFS to geometry is to discover useful algebraic structures of geometric objects affording the mathematics required of TFS. In this paper we describe Geometric Typed Feature Structures through one example of form: IOPSet. Our method of exposition is both mathematical and graphical: for each structure we will demonstrate both how it meets the necessary formal conditions as well as the sorts of form-sculpting operations it enables. An architectural example: insulated enclosure is used as a demonstration of subsumption operations over IOPSet. One alternative description of insulated enclosure using GTFS is also shown in the paper.

Keywords: *Geometric Typed Feature Structures, SEED, Design Space Explorer, geometric design information*

The Australian branch of the SEED project (Akin et al. 1997) created a new formalism for design spaces in which the fundamental structuring operator is information specificity, formally characterised as subsumption. Here design space navigation is composed as combinations of the primitive operators of resolution, unification, anti-unification, search, query and hysterical undo. The structures needed to support such a view are highly constrained in a mathematical sense and it is in these constraints that the problems for representation of geometry arise. Like all representations, those for geometry are tuned to support particular operations. In the case of geometry these typically are the geometric composition operators, for example, the Boolean operators of solid modelling. The research challenge is to add the formal design space exploration constraints into an existing geometric representation scheme or alternatively to discover a new scheme in which the constraints are realized.

In essence, we have done both. Based on Typed Feature Structures (TFS) (Carpenter 1992), Geometric Typed Feature Structures (GTFS) are a representation scheme and method for performing the basic design space exploration operations of resolution, unification, anti-unification, search, query and hysterical undo on geometric objects (Chang 1999; Chang and Woodbury 2000; Woodbury et al. 2000). The crucial insight behind extending TFS to geometry is to discover useful algebraic structures of geometric objects affording the mathematics required of TFS. Surprisingly there are many such structures each giving a way to sculpt geometric form within the tight constraints of the TFS theory. In this paper we describe Geometric Typed Feature Structures through four exemplary algebras of form. Our method of exposition is both mathematical and graphical: for each structure we will demonstrate both how it meets the necessary formal conditions as well as the sorts of form-sculpting operations it enables.

What Typed Feature Structures demand of geometry

Typed Feature Structures are a system comprising three main kinds of objects. ***Types*** provide generic accounts of objects and are arrayed in inheritance hierarchies. ***Typed feature structures*** are the objects of the system. These may or may not be instances of particular types. Typed Feature Structures are a kind of directed graph in which nodes represent domain objects and edges represent functional links between objects. Multiple references to the same node depict reference to the same object. The structure as a whole models partial information about a domain object. ***Descriptions*** are textual utterances in a ***description language.***

Two principal operations over Typed Feature Structures are ***subsumption*** and ***satisfaction.*** Subsumption provides a decision procedure for comparing two structures for information specificity, that is, it determines if one structure is a more specific version of the other. Satisfaction provides a means for calling out collections of feature structures that satisfy a description. In Typed Feature Structures both subsumption and satisfaction admit efficient (linear) algorithms.

Our extension, Geometric Typed Feature Structures, begins with the observation that geometric objects have indefinite numbers of sub-parts, none of which can be privileged over another. In other words, we are representing shape in Stiny's sense (Stiny 1980). Thus there can be no general ***a priori*** assignment of features to geometric objects. This excludes the use of typed feature structure feature as a representation device and leaves only the type hierarchy to carry geometric information.

Without features, type hierarchies are simply lower semilattices with the additional required property that, whenever upper bounds exist, there is a least upper bound amongst them. All lattices meet these conditions, as do other structures.

What this means for geometry is that, at a minimum, we can use as a type hierarchy any geometric alge-

bra whose operators induce a lattice onto the algebra. Algebraic operations correspond to movements to new types in the hierarchy. It suffices to have the following:

- A suitable algebra.
- A way or ways of calling out elements of the algebra.
- Efficient algorithms for comparing and combining objects.

In this paper we present an example type hierarchy that meets all of the conditions above. This is the set ***IOPSet*** in which objects have both an outer and inner boundary.

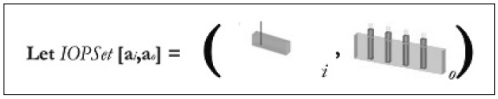
IOPSet

The example we provide here is called ***IOPSet, [Pset$_i$, Pset$_o$]*** under relation ⊙ where ***PSet*** is the set of all point-sets and ***Pset$_i$, Pset$_o$*** are ***IOPSet's*** inner and outer boundary. An instance g satisfies an ***IOPSet*** [a_i, a_o] if g is "larger"[1] than its inner boundary (a_i) and "smaller" than its outer boundary (a_o). In another words, ***IOPSet*** is a flexible geometry that grows from its inner to outer boundary. The most general case of ***IOPSet*** is its bottom that has an empty set as the inner boundary and the set of all point-sets (the universe) as the outer boundary. All point-sets are valid instances of this ***IOPSet***. The efficiency algorithms for comparing and combining objects are based on a so-called cell-based non-manifold representation (details of this representation shown in Chang1999, Chang and Woodbury 1997) that admits efficient comparison and combination operations after the merge step has occurred. The formal definition of an ***IOPSet*** is

Let ***a*** ∈ ***PSet***, ***b*** ∈ ***PSet***, [a, b] ∈ ***IOPSet,*** if ***a*** ⊆ ***b.***
Several instances of ***IOPSet*** are shown in the following example.

Conditions of inner boundary and outer boundary

As we know from definition above that ***IOPSet*** is the flexible geometry with inner boundary and outer boundary conditions. In other words, any forms and shapes of geometric entities can be included as long as they are within this range of boundary conditions. Supposed there is a solid concrete wall used as a diagrammatic representation of IOPSet [a_i, a_o] where a_i indicates the inner boundary and a_o indicates the outer boundary conditions for all point-sets.



The possible instance of above ***IOPSet*** is shown in the middle part of Figure 1.
Thus, the possibilities of the flexible geometry of ***IOPSet*** [a_i, a_o] are infinite as long they are within the range.

Concept of subsumption

We say that ***a*** subsumes (ô) ***b*** if a is a more general structure than ***b***. In our case, the set of instances of ***a*** must include the set of instances of ***b***. The formal definition is.

Let ***a, b*** ∈ ***IOPSet, a*** ô ***b*** if $a_i \subseteq b_i$ and $a_o \supseteq b_o$.

For example, there are two ***IOPSets a, b***, and ***a*** is more general than ***b***; which means inner boundary of ***a*** (a_i) is "smaller" than inner boundary of ***b***, and outer boundary of ***a*** (a_o) is "larger" than outer boundary

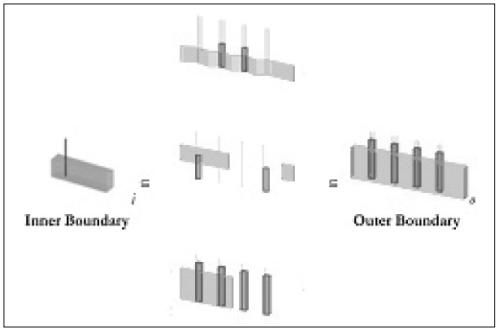


Figure 1
Possibilities of geometry between inner boundary and outer boundary conditions.

of ***b***. A 3D model representing such constraints is shown in Figure 2.

Let ***IOPSet a*** = [a_i, a_o] and ***IOPSet b*** = [b_i, b_o];

If [a_i, a_o] ô [b_i, b_o], then $a_i \subseteq a_o$, $b_i \subseteq b_o$, $a_i \subseteq b_i$, $a_o \supseteq b_o$.

Based on the description above, the subsumption operation over this structure is to specify the point-set inclusion relation of both inner/outer point-sets.

A way or ways of calling out elements of the IOPSet

There must be a way to specify members of the ***IOPSet*** using just the concept of subsumption described above. Primitive parametric objects, transformations of these and combination of both under the un-regularized or regularized Boolean operation can stand as names of inner/outer bounds of elements of ***IOPSet***. We choose the un-regularized forms as these are more general: the regularized forms work just work as well as a mechanism, but give a more narrow domain.

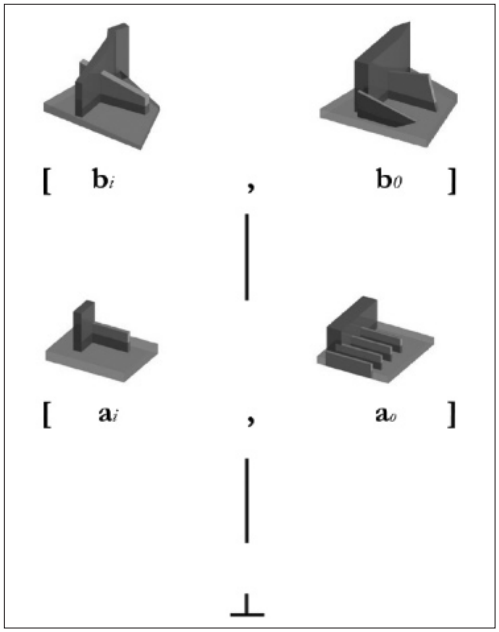


Figure 2
Concept of subsumption.

We define two operations ∀ and ✪ as:

$a \forall b = [(a_i \mid b_i), (a_o \cap b_o)]$, ***a, b*** ∈ ***IOPSet***, $(a_i \mid b_i) \subseteq (a_o \cap b_o)$

$a \text{ ✪ } b = [(a_i \cap b_i), (a_o \mid b_o)]$, ***a, b*** ∈ ***IOPSet***, $(a_i \cap b_i) \subseteq (a_o \mid b_o)$

such that a ô $(a \forall b)$, b ô $(a \forall b)$ and $(a \text{ ✪ } b)$ ô a, $(a \text{ ✪ } b)$ ô b

The set of such expressions using these two operations be called ***Exp***. Other algebras, for example, sweeps are also possible. An advantage of using these two operations is that it is possible to write expressions that name subsumed elements of ***a*** ∈***IOPSet***. These comprise any expression that can be reduced to ***a*** ∀ ***b***, ***b*** ∈ Exp.

To say it more explicitly in terms of hierarchy: Let ***IOPSet a*** = [a_i, a_o] and ***IOPSet b*** = [b_i, b_o], if ***a*** and ***b*** both subsume ***c*** and ***d*** subsumes both ***a*** and ***b***, then ***IOPSet, c*** = [$(a_i \mid b_i)$, $(a_o \cap b_o)$] and ***IOPSet d*** = [$(a_i \cap b_i)$, $(a_o \mid b_o)$]. Such hierarchy could be arranged via the 3D models as below.

Thus, ***a*** can be described as ***d*** ⊙ bottom (⊥), ***d*** can be ***a*** ✪ ***b***, and ***b*** = ***d*** ∀ ***e***. With this Exp, ***c*** can be ***a*** ∀ ***b*** and then ***a*** ∀ ***d*** ∀ ***e***. This demonstrates a simple expression based on the operation ∀. Each member of this hierarchy shown in Figure 3 can be described using this operation.

Architectural example: Insulated Enclosure

In this section, we will describe the mechanism described above using an architectural example - the insulated wall enclosure (Brand 1990). By specifying four layers of enclosure: continuous support, air barrier, insulation and sun/rain screen, for a given abstract massing, enclosure will create and modify

the geometry for each layer according to their constraints and relations among them.

The main reason for specifying the insulation enclosure is its demands for a multi-dimensional representation for geometry. As the representation of geometric objects, feasible geometry with constraints on both inner and outer boundaries is often required in specifying the enclosure technology. While constraints over geometry can be implemented with constraint resolution or propagation techniques, the exploration over these mechanisms is tedious and complex. However, with ***IOPSet*** described in this paper, specifying an enclosure that belongs to an abstract surface and its enclosure instances is just a subsumption operation over these two geometries. In addition, with strong typing system of GTFS, the multi-dimensional information of enclosure will be instantiated as one single order type under a unified hierarchy while not satisfies the geometric operations needed. An example operation is shown follows.

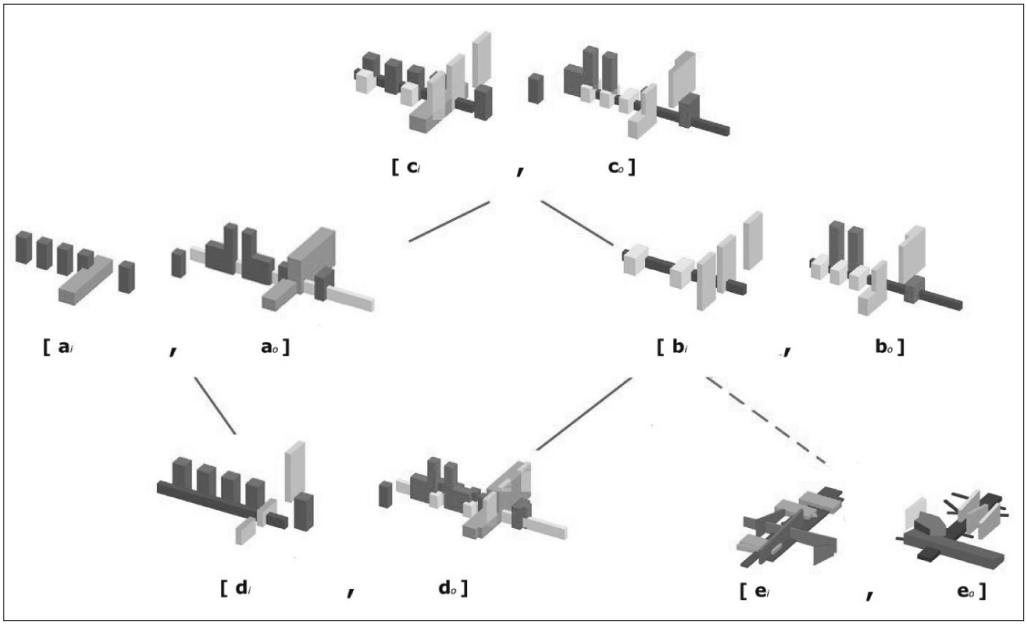


Figure 3
A diagram for demonstrating the way to arrange the IOPSet onto a hierarchy.

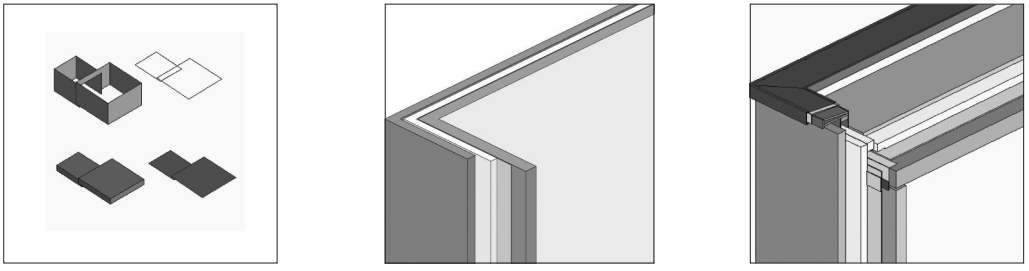


Figure 4
From left to right: abstract massing, four layers of wall enclosure, and the complete insulated enclosure.

```
fu_wall_encl
 AIR_BARRIER fu_air_barrier
          ATTACHMENT fu_attachment
                   DU du_attachment
                     FACE_OF iopset
                     GEOM iopset
                     THICKNESS real
                     TORCHED ballast
          DU du_material
            FACE_OF [0] iopset
            GEOM [0]
            MATERIAL [1] rubberised_asphalt
            THICKNESS [2] 1
          MEMBRANE fu_membrane
                 DU du_membrane
                   FACE_OF [3] iopset
                   GEOM iopset
                   MATERIAL [1]
                   THICKNESS [2]
 CONTINUOUS_SUPPORT fu_continuous_support
               ATTACHMENT fu_attachment
                        DU du_attachment
                          FACE_OF iopset
                          GEOM iopset
                          THICKNESS real
                          TORCHED ballast
               DU design_unit
                 FACE_OF [0]
                 GEOM [0]
                 THICKNESS real
               MEMBRANE fu_membrane
                      DU du_membrane
                        FACE_OF [0]
                        GEOM iopset
                        MATERIAL concrete_block
                        THICKNESS 3
DU du_wall_encl
    AREA real
    BOTTOM_CHORD [6] iopset
    FACE_OF [0]
    GEOM [0]
    THICKNESS real
    TOP_CHORD [5] iopset
```

```
INSULATION fu_layers
          ATTACHMENT fu_attachment
                   DU du_attachment
                     FACE_OF iopset
                     GEOM iopset
                     THICKNESS real
                     TORCHED ballast
          DU design_unit
            FACE_OF iopset
            GEOM [3]
            THICKNESS real
          MEMBRANE fu_membrane
                 DU du_membrane
                   FACE_OF [4] region
                   GEOM iopset
                   MATERIAL fibreglass
                   THICKNESS 2
SUN_RAINSCREEN fu_sun_rainscreen_wall
          AIR_SPACE fu_air_space
                 DU design_unit
                   FACE_OF iopset
                   GEOM iopset
                   THICKNESS real
          ATTACHMENT fu_attachment
                   DU du_attachment
                     FACE_OF iopset
                     GEOM iopset
                     THICKNESS real
                     TORCHED ballast
          DRAINAGE_HOLES fu_drainage_holes
                   DU design_unit
                     FACE_OF iopset
                     GEOM iopset
                     THICKNESS real
          DU design_unit
            FACE_OF iopset
            GEOM [4]
            THICKNESS real
          MEMBRANE fu_membrane
                 DU du_membrane
                   FACE_OF iopset
                   GEOM iopset
                   MATERIAL material
                   THICKNESS real
[5]  =\=  [6]
```

Figure 5
One exemplary description of a possible wall enclosure.

Conclusion

As stated before, design spaces are not adequately structured simply by derivation and that going beyond derivation to a richer set of structuring relations is a very hard problem, especially when geometry is involved. The approach described in this paper presents a way that allows geometry to be comprehensively included in highly structured design spaces.

The example ***IOPSet*** demonstrates a kind of geometric representation based on its point-set constraints. With an expression as Exp to describe the members of the set in an algebraic term, Geometric Typed Feature Structures unleash a new dimension in the field of design exploration and a promising potential in terms of incorporating knowledge with geometry. This approach resolves the first step of design space exploration. However, the behaviors and actions under the subsumption/satisfaction operations of Geometric Typed Feature Structures still require further investigation and exploration.

Reference

Akin, O., Aygen, Z., Chang, T.-W., Chien, S.-F., Choi, B., Donia, M., Fenves, S. J., Flemming, U., Garrett, J. H., Gomez, N., Kiliccote, H., Rivard, H., Sen, R., Snyder, J., Tsai, W.-J., Woodbury, R., and Zhang, Y. (1997). SEED: A Software Environment to support the Early phases of building Design. The International Journal of Design Computing.

Brand, R, (1990) Architectural Details for Insulated Building, New York, Van Nostrand Reinhold.

Carpenter, B. (1992). The Logic of Typed Feature Structures with applications to unification grammars, logic programs and constraint resolution. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Chang, T.-W. and Woodbury, R. F. (2000). Geometric Typed Feature Structures: Carrying Geometric Information using Typed Feature Structures. In Gudni Gudnason editor, Construction Information Technology 2000: taking the construction industry into the 21st century. Volume 1, Pages 166-177. Iceland. Icelandic Building Research Institute.

Chang, T.-W. and Woodbury, R. F. (1997). Efficient design spaces of non-manifold solids. In Liu, Y.-T., editor, CAADRIA 97, volume 2, pages 335-344, Hsinchu, Taiwan. Computer Aided Architectural Design Research in Asia, National Chia Tung University.

Stiny, G. (1980). Introduction to shape and shape grammars. Environment and Planning B: Planning and Design, 7(3): 343-352.

Woodbury, R.F., Burrow, A., Datta, S., and Chang, T.-W. (1999). Typed Feature Structures and Design Space Exploration. Artificial Intelligence for Engineering Design, Analysis and Manufacturing. 13:287-302.

Footnotes

1. "Larger" and “smaller" as described here are refer to the point-set inclusion/containment relation. A is larger than B if and only if every point of B is within A, that is $A \supseteq B$. A is smaller than B if and only if every point of A is within B, same as $A \subseteq B$.