EXPLORING AND COMPARING DESIGNS: THE SAME GAME?

Exploring and comparing designs

R.F. WOODBURY, S. DATTA and T-W. CHANG
School of Architecture, Landscape Architecture and Urban Design
The University of Adelaide, SA 5005 Australia
A.L. BURROW
Department of Computer Science
The University of Adelaide, SA 5005 Australia

Durability of Building Materials and Components 8. (1999) *Edited by M.A. Lacasse and D.J. Vanier*. Institute for Research in Construction, Ottawa ON, K1A 0R6, Canada, pp. 2640-2649.

Abstract

Design space explorers are computer programs that play on an *exploration* metaphor to support design. They assist designers in creating alternative designs by structuring the process of design creation in a space of alternatives. Subsidiary metaphors relevant to design space explorers are *generation*, *navigation* and *re-use*. The SEED project has developed a knowledge-level representation for a design space explorer. This paper sketches typed feature structures as a formal system in which a design space explorer and its knowledge level might be implemented. First, informal and abstract properties of typed feature structures suffice to build a sketch of the behaviour of a design space explorer. Second, using an example based on single-fronted cottages (a common Australian housing type), we outline the typed feature structure machinery most relevant to design space exploration. Finally we argue that the same mechanisms that enable design space exploration are applicable to evaluation.

Keywords: representation, evaluation, generative design

1 Introduction

This paper is about the representation of designs through the *t*yped feature structures formalism. It presents an informal sketch aimed at developing intuition about the representation and its fundamental computations. The representation was developed in the context of creating a *d*esign space explorer in which designers would search a space of possibilities for designs meeting a possibly changing set of criteria. Its relevance to the present conference theme arises because it addresses, in a particularly rigourous way, the comparison of representations for structural similarity. It is well-known that

in representing, for instance, a building envelope, a rich set of relations encompassing both form and function must be presented in order for an automatic evaluation to proceed. It is perhaps less well recognised that much evaluation can be accomplished, or at least pre-processed, by direct comparison between the symbol structures of representations. This is what the deemed-to-comply provisions of codes do in an informal way. We proceed with a sketch of the design space explorer knowledge and symbol level representations, arriving finally at a re-use mechanism that is based on structural similarity of representations.

In studying the phenomenon of design we use models (metaphors) to envision mechanisms by which computers might support design. One such mechanism is variously called search, exploration, discovery (and other terms)-under any name it is a guided movement through a space of possibilities. Design space explorers are computer media that engage designers in exploration. With them, designers explore possibilities. This encompasses discovering new designs, as well as recalling, comparing, and adapting existing designs. In creating design space explorers we represent the possibilities in the form of symbol structures called states and seek efficient representations for these states and algorithms over them. We represent spaces of possibilities as a relation over states and seek definitions of the space-generating relation which are both relevant to the exploration metaphor and lend themselves to tractable computations.

The aim of the SEED project(Flemming and Woodbury, 1995; Akin et al., 1997; Woodbury and Chang, 1995) is a comprehensive design space explorer exhibiting the above properties. SEED is targeted at the early phases of building design, where it is intended to encourage an exploratory design process by significantly shortening the time taken to generate and evaluate alternative designs. A common knowledge-level representation structures each of the main modules of SEED, which in turn further develop that representation in their own contexts. This knowledge level stands as a hypothesis of a complete, if abstract, description of SEED-Config as experienced by its users. It makes no commitments to its implementation, which might be realised in a number of different ways. This paper sketches *typed feature structures* as a formal system in which this knowledge level might be realised. Typed feature structures provide firstly formal properties useful to the knowledge level behaviour of SEED-Config, and secondly a mechanism for expressing knowledge level concepts.

Typed feature structures make the presumption, common but by no means universal, in the design literature(Newsome et al., 1989; March, 1995) that the entities of interest can be structured into a composition of objects and features. Thus representations such as shapes(Stiny, 1980) in which a definite representation comprises an indefinite set of parts would seem, at first glance, to be excluded from the typed feature structures game.

From the SEED-Config experience using typed feature structures, it turns out that the requirements posed on a representation for design space exploration are akin to those of evaluation in general. The strong relation between function and form, the ability to compare representations meaningfully and representation re-use are all crucial to evaluating designs.

A deep understanding of typed feature structures requires an appreciation of its formal mechanics, which, at least in the first author's experience takes fair effort to acquire. In this paper we give one sketch of typed feature structures emphasising their

behaviour as a representation for a the evaluation aspects of a generative design system. Woodbury et al.(Woodbury et al., 1999) gives a more detailed account of how typed feature structures model design space exploration and Carpenter(Carpenter, 1992) provides the definitivie description of typed feature structures, including many proofs for those readers wishing to tackle the formalism in full gory detail.

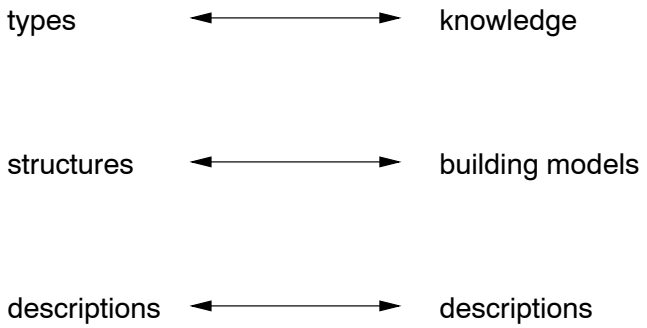

Fig. 1: Abstract elements of a typed feature structure representation

2 Typed feature structures as a representation

As a first approximation, consider a system made out of three sets of components: types T, structures F and descriptions D, put into a representation scheme with a building design domain as shown in Figure . The types comprising T stand for expressed knowledge of the domain of interest, in this case building design. Structures from F represent models of particular designs, in this case, buildings and/or their components, either physical or conceptual. Structures from F are expressed in terms of the information expressed in T. Descriptions from D are utterances in a formal textual language and stand for themselves in the representation scheme, but correspond symbolically and *inter alia* to sets of structures from F. The essential act of generation corresponds to the discovery of models of buildings that are simultaneously compatible with a description and the knowledge from the domain of interest—in typed feature structure terminology generation is the computation of structures compatible with a given set of types and a given description. All of this is ordinary in the realm of current knowledge based system theory and can be found, in many variations, in the literature. In the sequel we describe aspects that typed feature structures provide to a generative system. For now, as much as possible, we eschew the abstract symbolic terms of type, structure and description, relying instead on their domain counterparts, namely knowledge, model and description.

We use, as an example, a house type, common in Australia, known as a single-fronted cottage. Single-fronted cottages are simple houses, comprising a single-loaded hallway, rooms off of the hallway and a collection of spaces at the end of the hallway. The rooms along the hallway are typically generic in function and can be used as bedrooms, lounges, dining spaces, studies, etc. he hallway and its adjacent rooms are typically housed under a single roof, which is either a gable or a hip. The collection of rooms at the end of the hallway is typically configured as a skillion (space under a shed roof addition) or historically as separate structures (typically a kitchen and toilet). A porch along the front of the cottage completes the typical picture. Single-fronted cottages are a

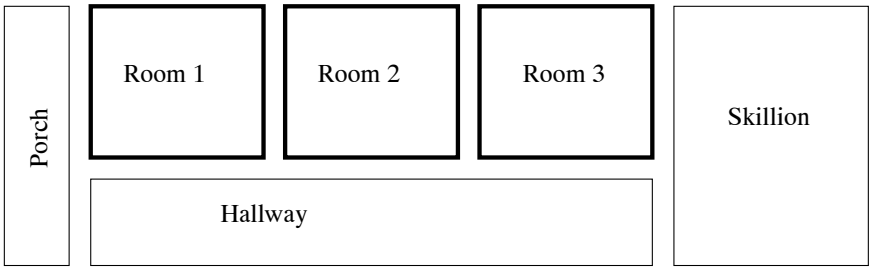

Fig. 2: Rooms 1, 2 and 3 can be identical in all aspects yet remain distinct

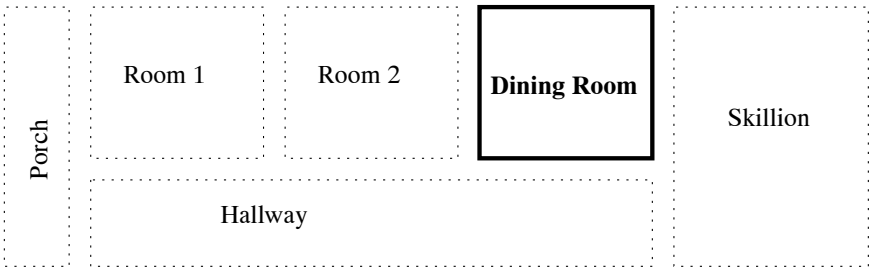

Fig. 3: The dining room and its location is all that is known

historic housing type, filling a niche of small, mostly inexpensive housing. Latterly many have been upgraded with particular attention being given to elaborations of the skillion addition, usually into a combination family room, dining room and kitchen. New houses in the single-fronted genre are still being constructed in inner suburban areas.

2.1 Structures are both intensional and partial

Intensionality in a representation scheme means that two symbol structures can be identical in all aspects yet remain distinct. In an intensional representation actual identity of two structures must be explicitly called out, for example, by a declaration that two compatible structures are, in fact, one and the same. From a domain point of view, we wish to be able to distinguish between models that differ only by the fact of their separate existences. As shown in Figure , in our example realm of single-fronted cottages, the three rooms of the room row can be identical in all aspects, yet remain distinct.

Partialness means that a structure need not contain all information that might be inferred about it from its associated types. In effect, a partial representation A stands for all more complete representations which contain at least the information in A. Appealing to a domain perspective, a model may contain less information than we can infer from our knowledge of it. Figure shows that a model of single-fronted cottage with its dining room in the room row can exist without holding any other information on single-fronted cottages.

Partialness and intensionality swim together. We can know very little of two structures, except that they are different—or that they are exactly the same structure. In terms of cottages, a we may know that the dining room is or is not the third room in the row. Conversely, within a computation, a given structure is not necessarily the same as another that contains consistent information, but might be made so at some future point in the computation. For example, Figure shows that two separate cottage models may

at some point become a single composite model, containing all information from both. In this figure and in all figures demonstrating information inclusion we depart from the normal convention in artificial intelligence of drawing specialisation relations with the most general concepts at the top. We do this for two reasons: (1) to cohere with the intuition that a structure containing more information than another is somehow "greater" than the other, and (2) the typed feature structures literature follows this convention.

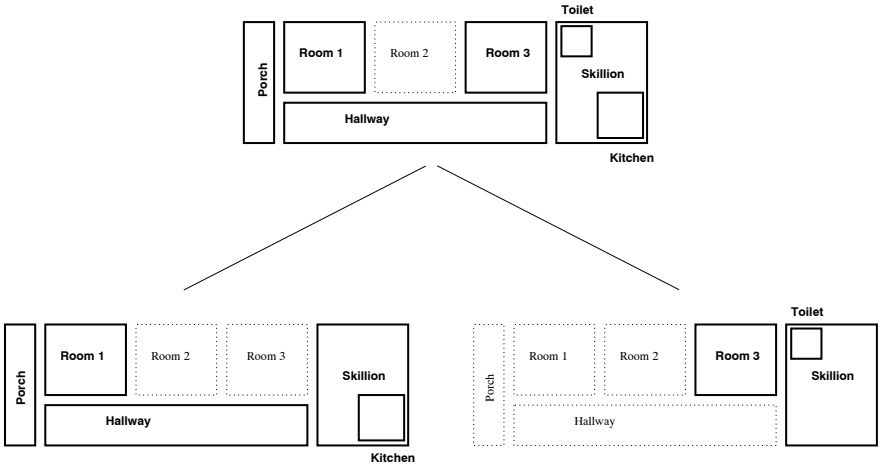

Fig. 4: Two cottage models that are consistent yet incomplete, can be joined into a single cottage model containing the information from both models

2.2 Partialness is in the eye of the beholder

Whether a structure is partial or not depends on the content of its associated types. In domain terms a model might be partial against one set of knowledge but complete with respect to a subset of the knowledge. For example, if knowledge of single-fronted cottages is limited to their spatial organisation, as might be the case for an urban geographer, a complete model of a cottage would assign functions to physical spaces. Such a model would be partial with respect to a larger set of knowledge, containing, for example, knowledge of how to construct single-fronted cottages. In terms of generative systems, partialness in a model indicates that more exploration is possible from that model. Conversely, adding new knowledge opens up new exploration possibilities for previously completed models.

In the domain of single-fronted cottages, Figure shows that a cottage about which we know only the location of the dining-room is consistent with alternatives for the placement of other rooms. If our knowledge of the building type consists of only spatial organisation, a model is complete once we have allocated all functions to specific rooms.

2.3 A generative system without rules

Recall that our formal system comprises three sets: types T, structures F and descriptions D, with elements of D standing for subsets of F. Further, imagine, firstly that there is a relation of satisfication M linking the structures from F that are consistent with a given description, secondly that there is an efficient procedure P for computing M, and thirdly that a type from T itself has a corresponding description in D. Then

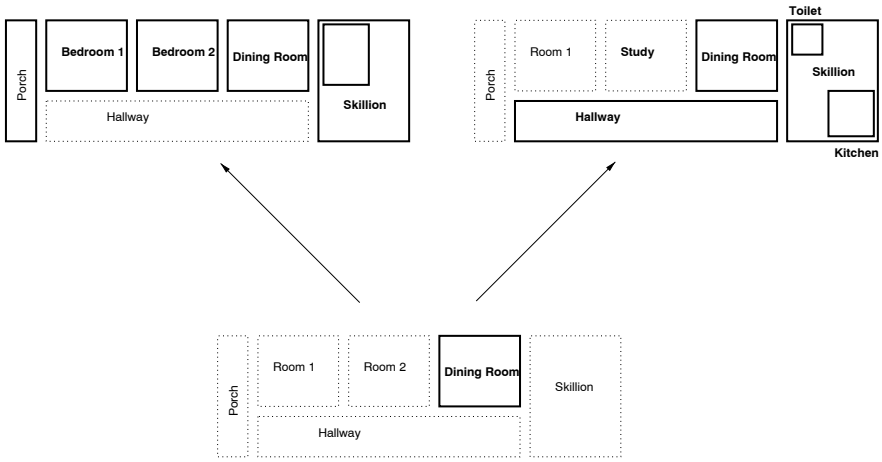

Fig. 5: A partial model of a single-fronted cottage is consistent with multiple complete models

the same procedure that generates structures consistent with a description can generate structures consistent with the set of types. In effect, the type hierarchy itself becomes what Carlson describes as a design space description mechanism(Carlson, 1994). In domain terms, from knowledge we can infer designs.

2.4 Monotonic generation

The fact of partial structures provides, in itself, a notion of what might comprise generation, namely the alternative completions of partial structures into more specific partial structures and finally to complete structures. This amounts to a claim that the generating procedure P must act incrementally, i.e., it must progressively generate more and more complete structures as it works through its input description. P thus stands as the design space generator in this scheme—by its action it traces out a derivation relation from state to state. Under such a regime, generation is monotonic with respect to the structures it creates. In other words, if a fact is known in a model, it remains known in all models that can be created from that model. It is well-known that monotonicity is a double-edged sword. While it enables search strategies based on branch-and-cut, it restricts the set of allowable operators and possible derivation sequences.

Appealing to our example domain, Figure shows that a partial model of a single-fronted cottage in consistent with multiple models, both partial and complete, and these models are related by the action of the procedure that generates one from the other. In this example, we are clearly recovering part of the subset relation over the set of rooms in the single-fronted cottage—general structures are more complex.

2.5 Design spaces are structured by subsumption

Monotonic generation implies that the derivation relation is a subsumption relation, i.e., an model B derived from an model A contains strictly more information than A. With respect to single fronted cottages, Figure shows that, in any derivation chain of models, each model contains strictly more information than its predecessor.

Derivation defines only a subset of the full subsumption relation. If we presume an efficient way of comparing structures from F for their information specificity, i.e.,

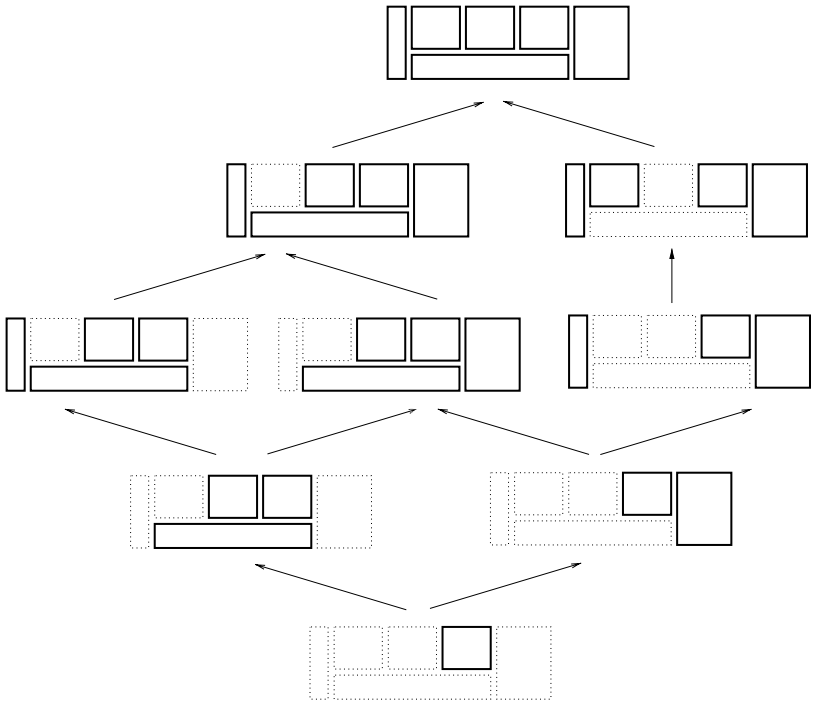

Fig. 7: From a partial model of a single-fronted cottage can be generated a graph of consistent partial and complete models

for the subsumption relation, then the full subsumption relation for a set of discovered structures can be recovered in the process of generation. Further, if structures are thought of as themselves comprising structures this subsumption relation can be recovered for all the parts of a structure.

Having access to the full subsumption relation in a design space enables forms of exploration richer than *branch-and-backtrack* usual in generative systems. The branch aspect of generation remains largely unchanged. Backtracking becomes more rich, in that the derivation relation along which backtracking typically takes place is replaced by the full subsumption relation—from a given state, it is possible to move to less specific designs that have previously been discovered and to less specific designs for parts, irrespective of the process by which the designs were generated.

2.6 Re-use and evaluation-by-structure are intrinsic

The aspirations of design space exploration trade heavily on an ability to re-use information already discovered, for example, to find and adapt for the present single-fronted design that clever island-feature kitchen design someone in your firm did in the Darwin office sometime in the last two years.

Together, partialness and the subsumption ordering over structures provide a seductive view on re-use of designs, the required actions for which are retrieval and adaptation. Recall that a structure is a partial specification of an object, that it is consistent with all objects that contain at least the information specified in it, and that structures exist in a subsumption ordering. It follows that any structure effectively indexes those structures more specific than it in the subsumption ordering—the same mechanism by which de-

signs are represented can be used for retrieval. Once retrieved, the design space explorer operations (monotonic generation, enriched backtracking and information removal) are available to adapt the design to a new context.

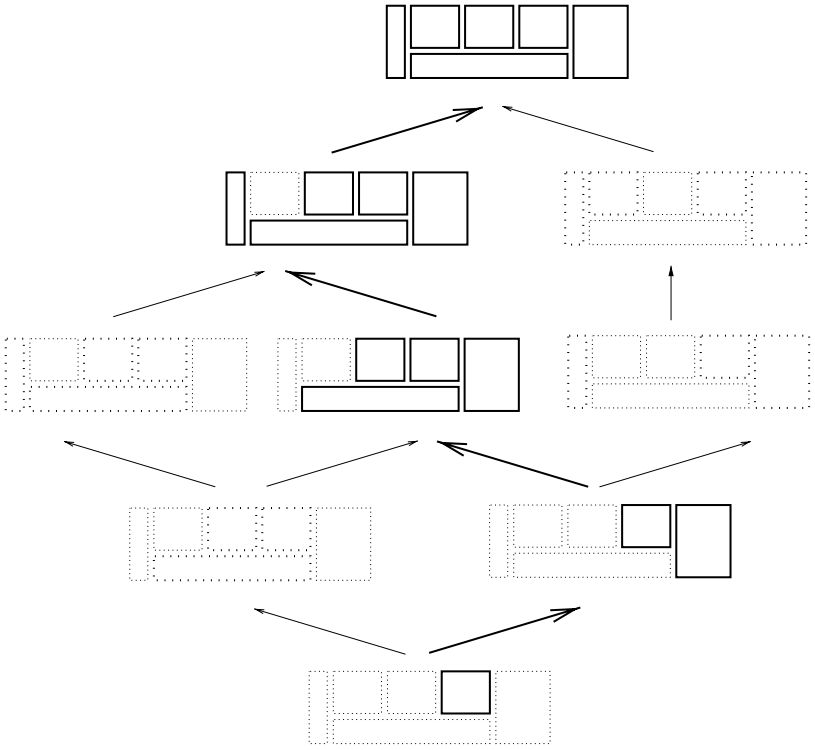

Fig. 7: The derivation relation is a subset of the full subsumption relation over models

In their support of re-use, typed feature structures would appear to do some of the duties of a case-based design system, for which the required mechanisms are representation, indexing, retrieval and adaptation. In the typed feature structures view, both cases and their indexes are indistinguishable from a structure, case retrieval indistinguishable from navigation in a subsumption relation of designs and case adaptation indistinguishable from design space exploration.

If we consider a case index as a structure, then indexes, like structures, have their own place in the subsumption relation of structures. More specific structures are instances of the case; less specific structures are a form of partial match of the case. This view on cases corresponds to the two kinds of matches that Flemming describes for cases in the SEED system(Flemming, 1994). Given the availability of the subsumption relation over structures and their parts, the index vocabulary problem described by Kolodner(Kolodner, 1993, Chapter 6) would appear to disappear—the index vocabulary is the full vocabulary from which structures are constructed and indexes are implicit in the subsumption ordering.

With respect to evaluation, this re-use ability means that exemplar designs demonstrating given properties, either in themselves or in their subparts, can directly compared to candidate designs to determine if the same properties hold. Further, the candidate designs can be extended without the chosen properties becoming invalid. Such an ability

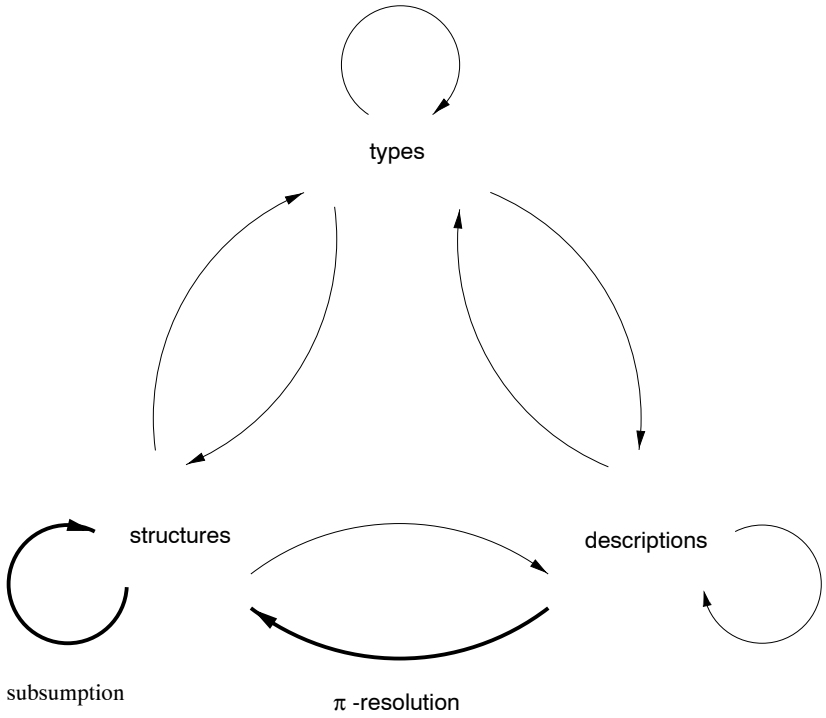

Fig. 8: Principal computation paths in typed feature structures

would appear to be a mechanism for certain types of evaluation, for example, compliance with deemed-to-comply provisions.

2.7 Summary

In the light of the above features, to the sketch of the typed feature structure representation scheme in Figure can be added two principal relations corresponding to computational paths amongst its symbolic components as shown in Figure .

The generating procedure P (called π-*resolution*) that captures a relation from descriptions to structures $M : D \rightarrow F$ is the main exploration mechanism in the system.

Access to the subsumption relation $S : F \rightarrow F$ for structures and their parts provides a design space structure which includes the derivation relation as a subpart. Case indexing and retrieval are essentially search in the subsumption relation. An episode of design space exploration would employ both P and movement in the subsumption relation.

Several computation paths would appear to be excluded from the family of computations available in this scheme. Types especially seems to be estranged. A more complete, and technical, presentation of the typed feature structures formalism would show that a description can call out a type, thus bringing types back into the fold. Most other paths have corresponding well-formed relations, but these are subsidiary in design space exploration and in design evaluation.

Typed feature structures are a model for design space exploration in which both the action of exploration and the structure of a design space are given a sound theoretical basis. As a representation of designs, typed feature structures provide well-founded support for an object-and-relations view of representation and within that view support

intensionality, partialness, monontonic generation, re-use and evaluation by structural comparison.

3 Acknowledgements

The authors would like to acknowledge the support given by the Australian Research Council Large Grants Scheme, The Department of Industry, Science and Tourism Major Grant Scheme, the Australian Postgraduate Award Scheme, The Commonwealth of Australia Overseas Postgraduate Research Scholarship Scheme, The Key Centre for the Social Applications of Geographic Information Systems and The University of Adelaide School of Architecture, Landscape Architecture and Urban Design.

4 References

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. http://www.arch.usyd.edu.au/kcdc/journal/index.html.

Carlson, C. (1994). Design space description formalisms. In Gero, J. and Tyugu, E., editors, *Proceeding of Formal Design Methods for CAD*, number 18 in B, pages 121–131, North-Holland. IFIP WG5.2, Elsevier Science Publishers B.V.

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.

Flemming, U. (1994). Case-based design in the SEED system. *Automation in Construction*, 3:123–133.

Flemming, U. and Woodbury, R. F. (1995). Software Environment to support Early phases in building Design SEED: Overview. *ASCE Journal of Architectural Engineering*, 1(4):147–152.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

March, L. (1995). The smallest interesting world? *Environment and Planning B: Planning and Design*, 23(1):133–142.

Newsome, S., Spillers, W., and Finger, S. (1989). *Design Theory '88*. Springer, New York.

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. L., Datta, S., and Chang, T.-W. (1999). Typed feature structures in design space exploration. accepted in AIEDAM Special Issue on Generative Systems in Design.

Woodbury, R. F. and Chang, T.-W. (1995). Massing and enclosure design with SEEDConfig. *ASCE Journal of Architectural Engineering*, 1(4):170–178.