# NAVIGATING SUBSUMPTION-BASED DESIGN SPACES

ROBERT F. WOODBURY, ANDREW L. BURROW, SAMBIT DATTA*
*School of Architecture,*
*Adelaide University, Adelaide, Australia 5005.*
*rw@arch.adelaide.edu.au, alburrow@cs.adelaide.edu.au*
**School of Architecture and Building,*
*Deakin University, Geelong, Australia 3217.*
*sdatta@deakin.edu.au*

**Abstract.** Design space explorers are information-rich environments conceived for providing effective support for electronic design processes. A subsumption-based design space structures the partial designs in the environment by a relation of information specificity. It provides formal exploration operators for predictive, goal-directed movement in the underlying space of designs and an interaction model for open-ended exploration. This paper focusses on the forward moving operator based on incremental pi-resolution and discusses the topic of information removal through the erasure operator. It describes the possible usage of these operators and the entry points for mixed-initiative human-computer interaction in the SEED-Config explorer .

## 1. Introduction

Design space explorers have their genesis in implementations of shape grammar editors, generative design systems and the human-computer interfaces to them (Giraud 1986; Carlson, 1993; Heisserman, 1989; Harada, 1997; Chien, 1998). This paper describes two exploration operators within a subsumption-based design space and the entry points for human-computer interaction. The navigation of a simple, abstract model of the design space is described. The space is shown with the initial node at the bottom in order to capture the intuition that more specific objects are, in an informational sense, greater than less specific objects. At any time in a computation, a finite part of this space will have been traced out by the action of navigation operations. This *discovered* part of the space is treated as incomplete both "above" and "below" a certain state. The following sections provide a discussion of our understanding of two distinct forms of navigation that arise in the exploration of a subsumption-based design space.

## 2. Subsumption-based Design Exploration

The use of order theory for information management and presentation and the logic of typed feature structures[1] forms the basis for the formal representation[2] of a subsumption-based design space(Carpenter 1992). Additionally, the ordering of design states based on a relation of information specificity must satisfy several formal and operational criteria(Woodbury 2000). Namely,

- *knowledge level,* the representation must correspond closely with the knowledge level of the domain,
- *partiality*, the representation must be able to express a collection of partial views of a single domain object at varying levels of specificity,
- *operators* there must exist operators to refine a partial view to a consistent and more specific view,
- *comparision,* there must exist an efficient algorithm to determine whether one view is more specific than another and
- *match*, there must exist an efficient algorithm to determine whether two views are consistent. Views that possibly represent the same object must be recognisable.

A design space representation satisfying the above criteria has been developed using the logic of typed feature structures(Woodbury 1999). With the representation in place, operators for navigation are developed for applying the formal ideas. In this paper, we discuss a mixed-initiative environment (M. Burstein, 1999; Manuela M. et al 1997) for exploring such a design space representation.

## 3. Mixed-initiative in Exploration

We treat design space explorers as information environments intended to provide novel and effective means of support for generative design processes based on electronic media. Consequently, a user interacts with the formal generative operators of the explorer and in doing so constructs a *design space* recording the states. In our view, while the operators have a formal basis, the exploration process itself is open-ended. Hence, it is both necessary and desirable for the user to control in several ways the automatic generation of designs.

In the field of human-computer interaction, researchers have identified an interaction model based on the dynamic sharing of initiative and control during

---

[1] Broadly, *feature structures* are much like the frames of AI systems, the records of imperative programming languages like C or Pascal, and the feature descriptions used in standard linguistic theories of phonology, and more recently, of syntax.

[2] While the seminal ideas are presented broadly, the reader is referred to the research reported on subsumption-based exploration, namely,(Carpenter 1992; Burrow, 1999; Woodbury 1999; Chang, 1999; Woodbury 2000) for a detailed exposition on the topic.

a collaborative session between the stakeholders in the session. A system in which the machine is collaborating with external users to solve a problem is called mixed-initiative. Mixed-initiative models of interaction form the basis for investigating navigation operations in subsumption-based design spaces. In this paper, we deal with the incremental Pi-resolution operation, the use of mixed-initiative in exploration and design hysteresis, an analogue of erasure or information removal.

## 4. Incremental Pi-resolution

Pi-resolution(Carpenter 1992), the fundamental feature structure construction process, is a non deterministic search for solutions to a query description. A solution is a feature structure, which satisfies a description and the constraint system}. It is the result of a sequence of extension steps corresponding to the satisfaction of constraints, which are organized into an inheritance hierarchy of types.

Incremental Pi-resolution(Burrow 1999) provides a natural entry point for human-computer interaction in the resolution process. Since Pi-resolution proceeds by extension, the resultant search space can be order embedded into the informational ordering over feature structures. Since constraints are drawn from an inheritance hierarchy, alternatives may be organized according to notions of abstraction. For example, if feature structures represent functional decompositions, Pi-resolution non-determinism allows exploration in terms of alternative versions of functional decompositions.

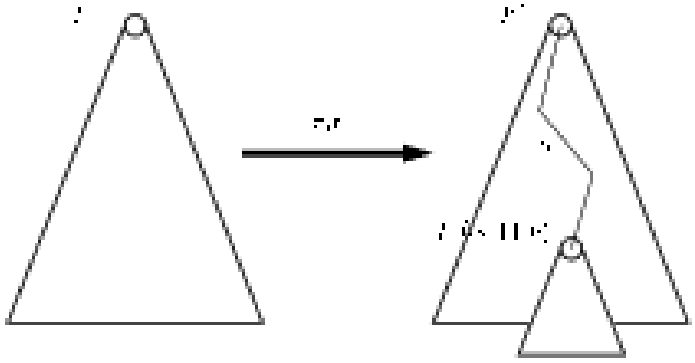

*Figure 1. A navigation step using the Pi-resolution operation.*

The incremental process allows a two-stage dialogue between the designer and the resolution process. First, given a query description, the explorer traces a path through the sub space of possible states that are consistent with the description and presents the results for user intervention. The designer then resolves the non-determinism in the operation and presents the explorer with the next legal operation. Second, the explorer retrieves the last element in the path

and presents its root node to the user. The designer then chooses one of the sub nodes of the structure and requests an incremental pi-resolution operation on this structure. The explorer then builds the set of types and associated recursive type constraints to which the selected node can be refined and presents them to the user. If the structure is fully resolved, the explorer marks the structure as resolved with respect to the type constraints, and requests the user for the next operation. Given a set of legal types, the designer can choose one of the subsuming types and present the explorer with a legal operation.

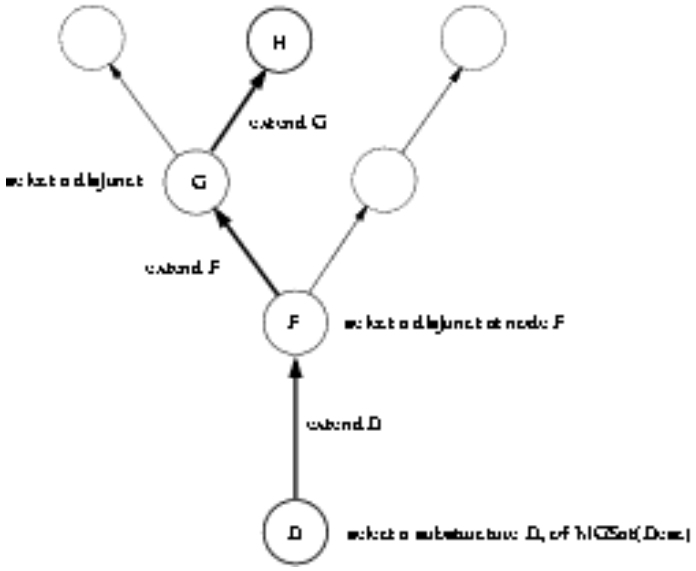


*Figure 2.* A schematic representation of the Extend operation

In both scenarios described above, the incrementality of the generative processs permits the designer to decide what the next forward operation in the satisfier space should be and thus have local control over the paths. Design decisions are not subject to some global inference strategy, but are goal-directed. And each resolution step introduces new constraints and opens up possible spaces via designer-explorer interaction.

## 5. Erasure

In knowledge-based sytems, individuals co-exist in complex relations with other individuals and hence present unique problems for destructive operations such as information deletion, common in CAD systems. Moreover, deletion has connotations in subsumption-ordered information structures that presents challenges in its conception as well as implementation. In the monotonic

subsumption ordered design space we propose, no information deletion is possible[3]. However, in implementing our subsumption-based explorer, we addressed the question, can a notion of information deletion be formulated within the formal properties of the satisfier space ?

Based on the properties of the space and the possible formal operations, we develop a view of *erasure* as not so much a removal of *(parts of)* a state, as the uncovering of *subsumers* of a state implicit or explicit in the design space. This formulation of deletion presents a clean view of information removal in our context and is presented as the erasure operator.

Thus, one can state that information removal lies in moving to less specific states from a given current state. Further, if the underlying order structure into whichthe current object is embedded is taken into account, it is formally possible to move *backwards* into states that are "new" states. Thus, erasure implies navigation that corresponds to movement in the down set of the space, from the current object to less specific implicit or explicit objects, that are *subsumers* of the current object.

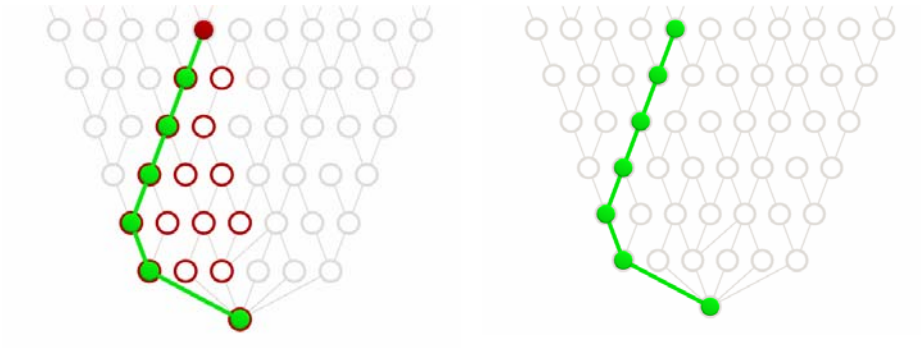

*Figure 3.* Hysteresis as an analogue of Erasure

The state from which information is *"erased"* would remain unchanged in the design space, even though the user's perspective would be of an altered state. Operationally, if a selected object is referenced by a single path and is in an explicitly discovered state, it may be "deleted" by retracting that path. This has the effect of moving to a more general object that does not reflect the forward refinement operation performed by incremental pi-resolution. Since, the reversal is performed on a currently selected context, this form of erasure can be performed across the width of the design space. If a current object is referenced by a multiple paths, and is in an explicitly discovered state, only one of the paths may be retracted, and the user presented with a set of feature paths that

[3] once information is committed to the space, satisfaction monotonicity restricts its removal. Information can only be added to the structure.

may be retracted to arrive at a previous explicit state. Figure1 shows the resulting space that might be uncovered by this form of erasure.

The user interaction with the erasure operator involves a mixed-initiative dialogue, similar to the previous operator.. However, the mixed-initiative interaction with the erasure operation must deals with hysteresis, a term that characterises the possibilities of opening up undiscovered implicit states while computing the least specific subsumers[4].The erasure operator renders the complex interactions between the explicit and implicit design spaces visible. Intuitively, design hysteresis is the following of back edges in the implicit design space, edges which may not exist in the explicit design space. The name refers to the lagged entry of these states into the explicit design space. Figure 3 shows an abstract example of states that might be uncovered due to hysteresis.

Once again the key to the design hysteresis operator is the manipulation of derivation steps and their record in the partial satisfier. It is computationally expensive to decide whether a particular feature structure is a partial satisfier as a matter of routine, and recognising possible back edges is a closely related problem. However, it is possible to recognise the surface steps as being a subset of the incremental Pi-resolution steps involving the maximal types. Once a surface step is identified, a search process similar to that of design anti-unification seeks out the resulting design state. During this process additional states will enter the explicit design space below the operand.

Thus, design hysteresis is a novel backwards navigation technique that conceptually extends simple backtracking. However, it is a close analogue of the forward generative operator. It makes explicit the states that are in the implicit design space order structure. From the point of view of the designer it requires similar input operations. Having activated a subproblem, the designer is prompted with a collection of surface steps in the form of type names. No selection of a structural variant is required, since the structural variant is committed to in the forward direction.

The user interaction with the Erasure operator involves a mixed-initiative dialogue, similar to Pi-Resolution. The user selects the Erasure operator and the explorer prompts the user to select a partial satisfier in the design space. The explorer then determines the erasure possibilities based on the queries outlined above.

## 6. The SEED-Config Explorer

The design of the software environment is based on the requirements of the SEED(Akin 1997, Woodbury and Flemming 1995) Knowledge Level. It is used

[4] A detailed discussion of Erasure and the role of hysteresis in design evolution is forthcoming in (Woodbury 2000).

for the specification and implementation of the SEED-Config design space explorer. SEED-Config, is a sub-module of SEED [5] and supports three dimensional schematic design of building forms and technical systems. It provides the generative machinery that enables a user to operate at a higher level of abstraction than conventional CAD systems by focussing on configuration design rather than drawing descriptions.

Presently, the explorer contains four modules, EntryPort, KryosConfig DesignState and DesignSpace. The EntryPort is the container module that loads all the context necessary for design space exploration.. It is envisaged that the other modules of SEED, as well as external libraries will be available to the explorer through this module as plug-ins. The KryosConfig module provides

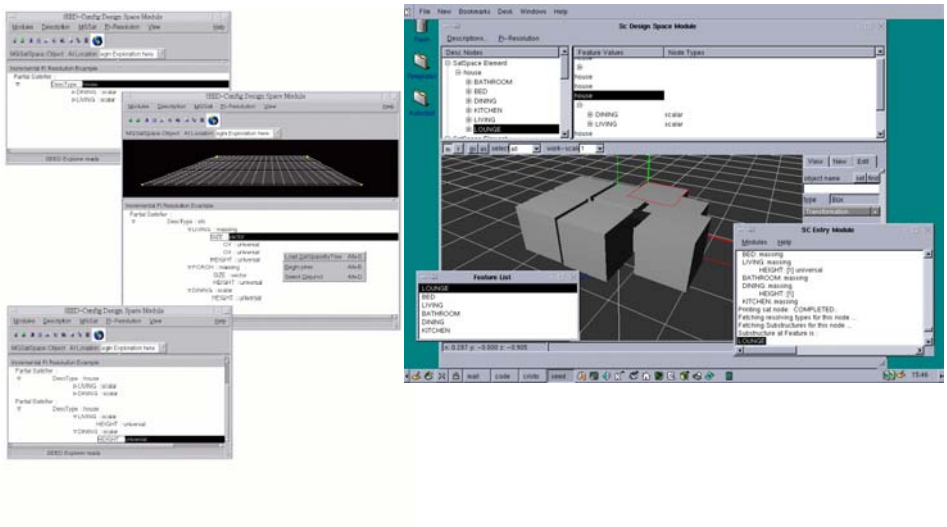

*Figure 4.* The design space explorer interfaces

functionality for choosing the types, features, constraints and descriptions necessary for building a representation of a design space. The DesignState module provides a view of a single partial satisfier state. The DesignSpace module affords the user three views of interaction with the design space exploration machinery. First, the satisfier space can be constructed and the minimal satisfier displayed. The substructures of the satisfier can be unfolded by iterating the elements from the trivial empty path to the last element. Secondly, the satisfier nodes can be selected for Pi-resolution. If the node has disjuncts, a popup window with the list of disjuncts at that point are provided, else a list of resolving types are provided. Selecting a disjunct or a type resolves the choice at the node and a new element is introduced into the space. Thirdly,

[5] The SEED project and its various modules are accesible online from *seed.arch.adelaide.edu.au* and *seed.edrc.cmu.edu*

the user can view and inspect the results of the resolution in the graphical canvas. The interfaces of the design space explorer are shown in Figure 4.

## 7. Conclusion

This paper presents two operations for navigation in a subsumption ordering of partial design states, incremental Pi-Resolution and Erasure. We describe how a user interacts with the incremental Pi-Resolution operator for monotonic forward moving navigation. The second operation described in the paper is based on the notion of information removal during navigation, namely, erasure. Each of the operations are described with reference to their formal properties and their role in a mixed-initiative environment.

The most surprising fact is that only incremental Pi-resolution steps that have been performed to reach the state may be undone. This reflects the direct relationship between a partial satisfier and the set of steps that reach it. All paths to the partial satisfier contain the same steps, only the sequence changes. This "set value" is the crux of the design space operators. The structure of the implicit design space is in fact a consequence of the freedom the designer experiences in directing the incremental Pi-resolution evaluation.

Other operations in a subsumption-based design space, such as Join, Meet, Reuse and Revisit are under investigation. The exploration operators are designed and implemented in the Kryos class libraries, a C++ implementation of typed feature structures. The user interfaces utilise the Qt GUI-toolkit and OpenGL Graphics libraries while the geometric data use the Shapes libraries. Collectively, they comprise the implementation environment for navigating in subsumption-based design spaces.

## References

Akin, O., Z. Aygen, T. Chang, S. F. Chien, B. Choi, M. Donia, S. Fenves, U. Flemming, J. Garrett, N. Gomez and H. Kiliccote, H. Rivard, R. Sen, J. Snyder, W.Tsai, R. Woodbury and Ye Zhang: 1997, SEED : A Software Environment to Support the Early Phases of Building Design, The International Journal of Design Computing Volume 1. http://www.arch.usyd.edu.au/kcdc/journal/index.html.

Burrow, A., R. F. Woodbury. : 1999 Pi-resolution and Design Space Exploration, Proc of the eight International Conference on Computer-Aided Design Futures: 291--308.

Carlson, C. : 1993, Grammatical Programming: An Algebraic Approach to the Description of Design Spaces, PhD Thesis, Department of Architecture, Pittsburgh, Carnegie-Mellon University.

Carpenter, B.: 1992, The Logic of Typed Feature Structures with applications to unification grammars, logic prorgams and constraint resolution, Cambridge University Press.

Chang, T.-W.: 1999, Geometric Typed Feature Structures, PhD Thesis, School of Architecture, Adelaide, Adelaide University.

Chien, S.-F.: 1998, Supporting Information Navigation in Generative Design Systems, PhD Thesis, School of Architecture, Carnegie-Mellon University.

Giraud, C., R. Krishnamurti.: 1986, Towards a shape editor: the implementation of a shape generation system."Planning and Design **13**(4): 391--403.

Harada, M. : 1997, Discrete/Continuous Design Exploration by Direct Manipulation. PhD Thesis, Department of Architecture. Pittsburgh, Carnegie Mellon University.

Heisserman, J.: 1991, Generative Geometric Design and Boundary Solid Grammars. PhD Thesis, Department of Architecture. Pittsburgh, Carnegie Mellon University.

M. Burstein, A. M. a. S. D.: 1999, An Approach to Mixed-Initiative Management of Heterogeneous Software Agent Teams. Thirty-second Annual Hawaii International Conference on System Sciences, Maui, Hawaii.

Manuela M. et al.: 1997, Rationale-Supported Mixed-Initiative Case-Based Planning. Proceedings of IAAI-97.

Wegner, L. C. a. P.: 1985,On understanding types, data abstraction and Polymorphism. Computing Surveys **17**(4): 471--522.

Woodbury, R., A. Burrow, S. Datta and T. Chang : 1999, Typed Feature Structures and Design Space Exploration. AI in Design, Engineering and Manufacturing **13**(4), pp. 287-302.

Woodbury, R. and U. Flemming, R. C., S. Fenves and J. Garrett: 1995, The SEED Project: A Software Environment to Support the Early Phases in Building Design. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings of the Eighth International Conference, Melbourne, Australia, Gordon and Breach Publishers.

Woodbury, R., S. Datta, A. Burrow : 2000, Erasure in Design Space Exploration. Accepted for publication in AI in Design, 2000.

.