Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2006), *20*, 129–142. Printed in the USA.

DOI: 10.1017/S0890060406060124

RESPONSE TO KEYNOTE

Modeling dialogue with mixed initiative in design space exploration

SAMBIT DATTA

School of Architecture and Building, Deakin University, Geelong, Victoria, Australia

(Received June 30, 2005; Accepted December 16, 2005)

Abstract

Exploration with a generative formalism must necessarily account for the nature of interaction between humans and the design space explorer. Established accounts of design interaction are made complicated by two propositions in Woodbury and Burrow's Keynote on design space exploration. First, the emphasis on the primacy of the design space as an ordered collection of partial designs (version, alternatives, extensions). Few studies exist in the design interaction literature on working with multiple threads simultaneously. Second, the need to situate, aid, and amplify human design intentions using computational tools. Although specific research and practice tools on amplification (sketching, generation, variation) have had success, there is a lack of generic, flexible, interoperable, and extensible representation to support amplification. This paper addresses the above, working with design threads and computer-assisted design amplification through a theoretical model of dialogue based on Grice's model of rational conversation. Using the concept of mixed initiative, the paper presents a visual notation for representing dialogue between designer and design space formalism through abstract examples of exploration tasks and dialogue integration.

Keywords: Design Spaces; Exploration; Generation; Mixed Initiative; Navigation

1. INTRODUCTION

Computational design exploration with description formalisms can be explicitly modeled as a form of movement in a structured space of designs. The design space exploration complex presented by Woodbury and Burrow is one such formalism.

In their Keynote Article, Burrow and Woodbury explain the three premises underlying the metaphor of exploration. All three have important ramifications for the work presented in this paper. The first premise posits that the cognitive limits of humans in representing and searching problem spaces can be addressed by modeling designer action as a form of exploration. The second premise posits that a model of design space exploration can amplify designer action through formal mechanisms. The final premise is that the representation of computational exploration is feasible. The need to account for and understand the nature of designer action, amplification of design activities, and the feasibility of encoding formal and structured spaces remains a significant research problem. Woodbury and Burrow present two propositions, namely, the primacy of design space in this discussion, and secondly, the need to support computational amplification.

In the context of this framework for computing exploration, namely, the need to develop a better understanding of what we mean by design space (as opposed to a search or solution space) and how human designers can communicate, control, and coordinate the formal process of exploring such structured space of designs, this paper will address these issues through a discussion of interaction paradigms for designing, Grice's model of rational conversation and mixed initiative.

The formal machinery of design space exploration is explained in the literature (Burrow & Woodbury, 1999; Woodbury et al., 1999, 2000) and is relatively well understood and exemplified in the work of Woodbury and Burrow. However, the problem of incorporating this formal machinery with designer action for amplifying human intention remains an open one.

Reprint requests to: Sambit Datta, Level 4, School of Architecture and Building, 1 Gheringhap Street, Deakin University, Geelong, Victoria 3217, Australia. E-mail: sambit.datta@deakin.edu.au

Interaction paradigms provide a mechanism for introducing human design intent into computational exploration. Several paradigms for human–computer interaction during design exploration have been proposed in the literature (Kochhar, 1994). These are classified as manual, semiautomated, automated, and cooperative, reflective of the degree and extent of the division of labor between human and machine. However, with the shift of emphasis implied by the work of Woodbury and Burrow, this neat division of labor model of design interaction is not sufficient to address the complex issues of designer action and amplification. For example, the mimicking of manual human action, such as sketching or sculpting as an interaction model, while compelling, is unsuitable for design space exploration. Manual action necessarily deals with singleton states, not collections or threads of states.

In our research, the mixed-initiative interaction paradigm presents a more fine-grained division of labor, allocating and sharing control over the same task jointly between the user and the system, allowing for multiple threads, recovering from conflict through disambiguation and many other strategies for interaction known from conversational structure. This paper argues that a useful starting point for addressing this problem lies in conceiving the interaction between the formalism and the designer as a rational conversation and proposes a model of dialogue. The concept of mixed initiative is known from studies of human dialogue, applied artificial intelligence, and the design of autonomous goal-directed systems (Ferguson & Allen, 1994; Allen et al., 1996). Such a paradigm provides a systematic exposition of how communication, coordination, and control strategies enable a designer to interact with a formal system.

The need for a fine-grained division of tasks, acquiring, sharing, and relinquishing initiative requires a model for synchronizing the input and output modalities between the designer and the exploration formalism. An interaction model addressing these requirements has been proposed in the literature (Datta & Woodbury, 2002). The central feature of this model is the maintenance of exploration structure based on conversational dialogue and its computational representation.

This paper describes the representation of dialogue between the designer and state, structure, and moves in the formalism. It presents a visual notation for representing dialogue through abstract examples of exploration tasks that are specifically machine or human driven in an asynchronous fashion. It describes examples of dialogue integration, where the two distinct modalities are combined in a single frame, in a synchronous fashion. It concludes with a discussion of the results from the modeling of the nature of dialogue in design space exploration.

1.1. Background

The representational device of feature structures (Knight, 1989; Kasper & Rounds, 1990) and attribute value logic (Pollard & Moshier, 1990; Franz & Jrg, 1994) known from linguistic theories of generation and from the constraint programming literature underpin the notation for modeling language in Carpenter's typed feature structures (Carpenter, 1992). As a representation, typed feature structures are similar to frame-based (Minsky, 1975) and terminological knowledge (Borgida et al., 1989) representation systems. The analogy between feature structures and knowledge representation schemes comes from associating a collection of features or attributes with each node or frame. Each feature represents a slot label and the arcs themselves point to the fillers, creating a network of associations. Feature structures comprise a set of nodes, each of which is labeled by type information. Most knowledge-based systems underpinned by symbolic representation support the "matching of descriptions." This operation, called unification in the field of deduction systems provides the "addition and multiplication" of descriptions.

A transparent exposition of the modalities of the formalism in a visual manner is one way of expressing and integrating the input and output modalities of both user and formalism. This approach is similar to the work of Piela (1989) in the ASCEND modeling system (Piela et al., 1993). ASCEND provides a visual and direct manipulation interface for developing and testing incremental constraint programs in the domain of process engineering. In the ICE project (Zeller & Snelting, 1995; Zeller, 1997) an interactive front end enables the user to construct configuration threads through the addition and modification of configuration constraints. The Oz Explorer (Schulte, 1997) is another visual constraint programming tool for supporting the user driven development of constraint programs. A tree visualization of the constraint problem is the central metaphor for exploration of any constraint node in the tree. The user can tailor and program user guided search engines over this tree for the development of constraint programs. FEGRAMED (Kiefer & Fettig, 1995) employs a fully interactive front end that presents the user with a customized view of feature structures. This feature structure editor can be used for developing and maintaining feature structures in constraint-based systems.

1.2. Dialogue requirements in design space exploration

The theory of design space exploration (Woodbury et al., 1999) posits an explicit formal substrate for computing exploration that employs and develops extensions to Carpenter's typed feature structures (Carpenter, 1992) to represent and reason over intermediate states, exploration moves, and an ordering over explored states.

First, the formalism supports the representation of an exploration state that retains the feature structure properties of intentionality and partialness. Types, features, and descriptions taken together provide well-founded support for representing problem and solution states. The state repre-

sentation supports useful formal properties such as intentionality, partialness, and structure sharing. Second, the inference algorithms over this representation support a principled notion of exploration moves that enumerate partial states under a subsumption ordering. Moves are operators that can generate new states, and navigate and modify existing states. Third, the exploration formalism supports several movement operations over state and space. Underpinning the exploration of partial satisfiers is an ordering based on subsumption.

The entities of state, structure, and move identified above are formally represented using typed feature structures as the formal substrate as follows:

1.2.1. Dialogue with exploration states

The formalism supports the representation of an exploration state through the concepts of *types, features, descriptions, and feature structures*. The formalism represents exploration states through three elements from the feature structures machinery. These elements are a type hierarchy, a set or sets of feature structures, and a description language for specifying constraints on types and structures. Types comprising T stand for domain knowledge of the allowable universe of discourse expressed in terse form. Structures from F represent exploration states, in this case, physical and conceptual attributes associated with the design of buildings. Descriptions from D are constraint expressions in a formal attribute-value description language. Descriptions are used for problem formulation, constraints on types, and generated structures.

1.2.2. Ordering of exploration structure

The structure of exploration is represented through the ordering relation of subsumption. The concept of an ordered design space underpins the description formalism. In it, the collection of exploration states is ordered by the relation of subsumption over exploration states (feature structures). Subsumption as a formal inference operation is widely used for reasoning.[1] The subsumption ordering relation can answer what superclass a given class has according to its set of attributes. In the context of the exploration formalism, subsumption is a relation of implication that relates more specific to more general states of exploration. Thus, like inheritance over types, subsumption defines a partial ordering over exploration states (feature structures). This ordering of feature structures is represented as a directed graph. The hierarchical graph defined by subsumption over feature structures is such that a child node may have more than one parent node. The exploration formalism provides a structuring relation based on subsumption to order collections of exploration states. This ordering relation provides an invariant structure to the design space. Here, the subsumption relation may be seen as a generalization relation. Thus, in a given design space structured by subsumption, a subsumer state expresses a generalization over the subsumed state. In it, two design states (and recursively their subparts) are related if one subsumes the other, that is, if one contains strictly less information than the other. This subsumption-based design space structuring mechanism is reported in Burrow's thesis (Burrow, 2003). The key feature of this structuring mechanism is that the subsumption relation captures a more generic relation than the derivation relation that structures the spaces of designs generated by a grammar. A detailed view of this formulation is set out in (Woodbury et al., 1999).

1.2.3. Algorithms for computing exploration moves

Exploration moves are cast in terms of moves in a design space upward or downward in an information ordering. The formalism provides a set of operators, namely, incremental π-resolution (Burrow & Woodbury, 1999), indexing and reuse, design unification, design antiunification, and hysterical undo (Woodbury et al., 2000) for the generation of new exploration states, modification of existing states, and movement between states.

Given this representation of state move and structure in a description formalism, the requirements for user interaction with the design space exploration are addressed.

2. CONVERSATIONAL MODEL OF DIALOGUE

The key notion underpinning the model of dialogue is conversational structure. This establishes a shared representation of discourse that enables the participants (human and computational) in the dialogue to negotiate, reference, and confirm mutual understanding.

A conversational model of dialogue enables the possibility of extended interaction (Allen, 1999) between the user and the system. Grice's (1975) maxims of rational conversation is one such formulation.[2] Admitting Grice's conversational maxims is one way of achieving a tight coupling of user actions with the formal substrate and implementing a model of dialogue.

Grice (1989) observes that human dialogue is characterized by rationality, cooperation, common purpose, and direction. He states,

> Our talk exchanges do not normally consist of a succession of disconnected remarks, and would not be rational if they did. They are characteristically, to some degree at least, cooperative efforts; and each participant recognizes in them to some extent, a common purpose or set of purposes, or at least a mutually accepted direction.

[1] For example, description logics (Lambrix, 1996) provide a reasoning operation called subsumption. Subsumption also provides a powerful tool for case-retrieval and standards processing (Hakim & Garrett, 1993).

[2] The literature on conversational structure is large and is review is beyond the present scope. It suffices here to have a model suitable for organizing mixed-initiative interaction with a computer. For this, rational conversation is a good model and Grice an exemplar.

Based on the above observation on human conversation, Grice formulates a set of conversational maxims. The first is *quality*, implying truthful and accurate information. The second is *quantity*, neither more nor less information than is required for the dialogue. The third is *relation*: only information appropriate to the task is considered. The fourth is *manner*: clear and unambiguous information is necessary for conversation. Grice's model of rational conversation forms the basis for addressing communication, coordination, and control issues in the interaction model.

Given a sound representation of communication and coordination, it is necessary to address the sharing of control over the thread of exploration. An incremental model of turn taking between the user and the formalism enables both participants in dialogue to acquire, shift, and allocate control of the exploration process. The dialogue layer must support a robust structure of turn taking between the user and the formalism. Incrementality and turn taking enable the best joint interpretation of input and output modalities between the designer and the formalism during exploration.

3. THE FEATURE NODE REPRESENTATION

A common symbolic representation is proposed for supporting the modalities of input and output during mixed-initiative exploration. Exploration dialogue between the user and the formalism is represented through a visual representation of a feature node.

The dialogue layer must support communication and coordination between the designer and the formalism. Representing the input and output modalities of dialogue between the user and the formalism is a key requirement of mixed initiative in the dialogue layer. The user or the formalism must be able to communicate and coordinate the input and output modalities during dialogue. The input and output modalities of the description formalism are clearly defined in the terms of the formal substrate. Descriptions and partial satisfiers, the base representation of input and output from the formalism are given, *ipso facto*, in terms of feature structures. The dialogue layer must provide an account of the input and output modalities of the designer. A common representation that can unify both modalities of exploration is necessary.

The conception of a design state is made transparently visible to the user through a visual feature node, VNode. Through the construct of the visual feature node, a principled formulation of mixed-initiative conversational structure can be established between user and formalism. These properties of a VNode are based on Grice's (1989) model of rational conversation. The relationship between the domain contruct of an exploration state and the visual feature node is shown in Figure 1. Here, the representation of dialogue based on rationality, cooperation, common purpose, and direction through the visual feature node is addressed.

The concept of a visual feature node enables a principled formulation of dialogue representation. As shown in Figure 2, the visual feature node, VNode provides a common frame for representing mixed-initiative dialogue in the interaction model. Two distinct views are mapped onto the same representation. First, the results of exploration initiated by the description formalism are available as partial satisfiers (incorporating types, features, descriptions) of a VNode. Second, the representation of the results of user manipulation are available as feature nodes (incorporating problem states, choices, and functions, interaction history) through the extrinsic attributes of the VNode.

3.1. A visual notation

The visual notation expresses the input and output modalities from both the user and the formalism. The visual notation extends the representation of attribute-value feature structures.

The attribute-value matrix or AVM notation visually describes feature structures. A visual feature node maps the intrinsic and extrinsic attributes of a feature node, FNode onto elements of the AVM notation. This mapping annotates the visual feature node with the type, feature names, feature values and coreferences taken from the underlying partial satisfier. The connection between a visual feature node, VNode and an FNode is given as follows:

$$\text{VNode} \equiv \text{VNode} \cdot \text{FNode}. \tag{1}$$

The VNode composes and aggregates elements of the underlying representation. As shown in Eq. (1), a VNode can be decomposed into an FNode. The link between the visual notation and the underlying representation is expressed with a dot notation. In an FNode, the values of a feature may be atomic, complex, or another feature node. The value of a feature node FNode, is given either by a *feature-value pair* or *feature-value map*. The smallest element of the visual feature node is the feature-value pair. The feature-value pair represents the relation between a feature and its value. This is shown in Figure 3.

The feature-value map specifies the relation between a feature node and its subnodes. For example, in Figure 3, the feature-value map represents the functional relationship between the GEOM and its value. A feature-value map is enclosed by the delimiters "[" and "]" and annotated by the type label drawn from its partial satisfier. In the example, the partial satisfier is of type, geometry. Feature nodes support recursive containment. Thus, the value of a feature node may be another feature node. The attribute-value notation is easily adapted for a visual representation of a feature-value map as follows: the feature-value map can be conceptualized as a recursive container of entities of type feature-value pair. In Figure 3, the feature-value pair represents the functional relation between the feature MASS_EL and its value, which is minimally the type *massing*. The value of MASS_EL may also be complex, such as a query description, resolution step, or function application or exter-

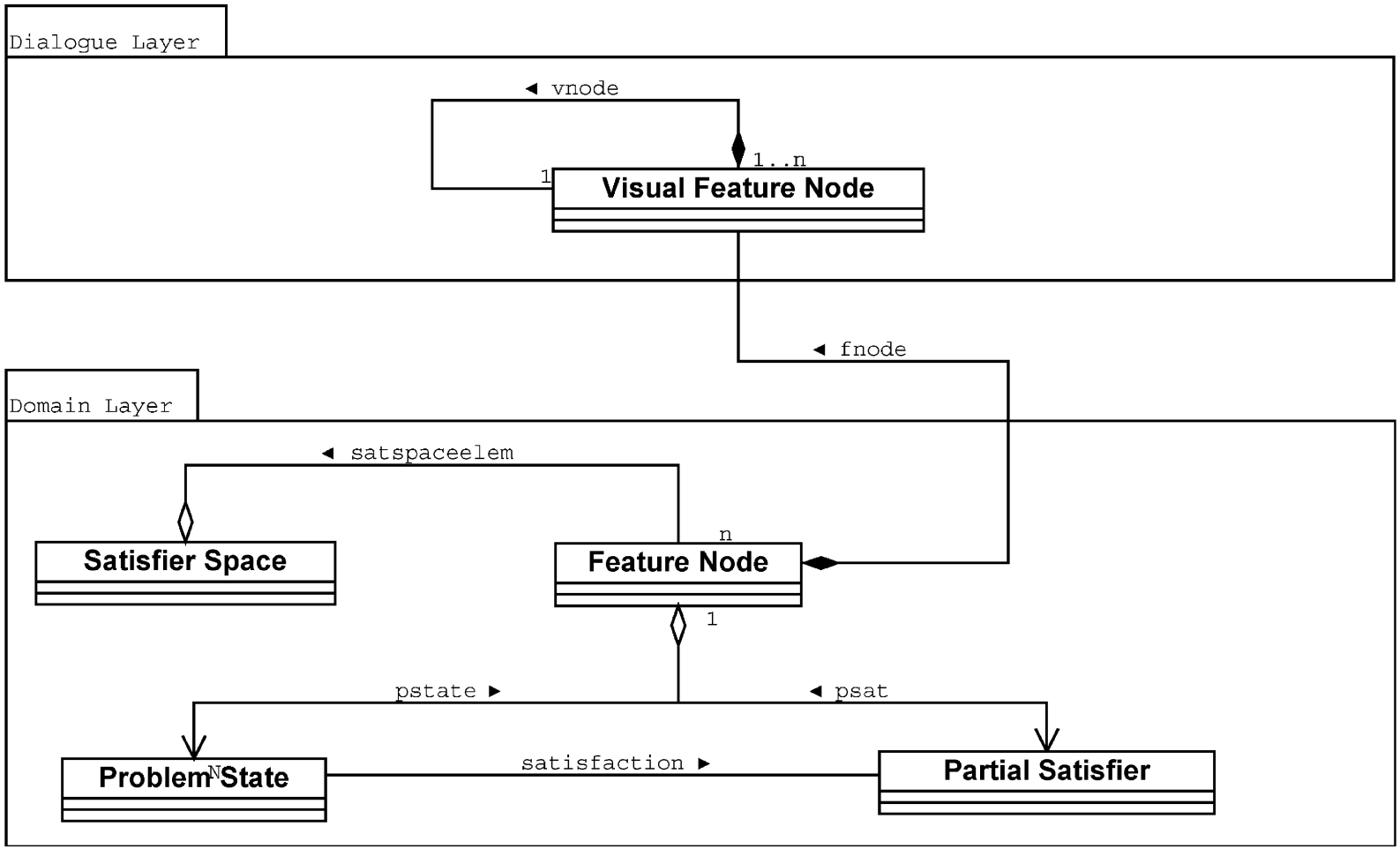

Fig. 1. Mapping domain layer constructs to the dialogue layer through the visual feature node.

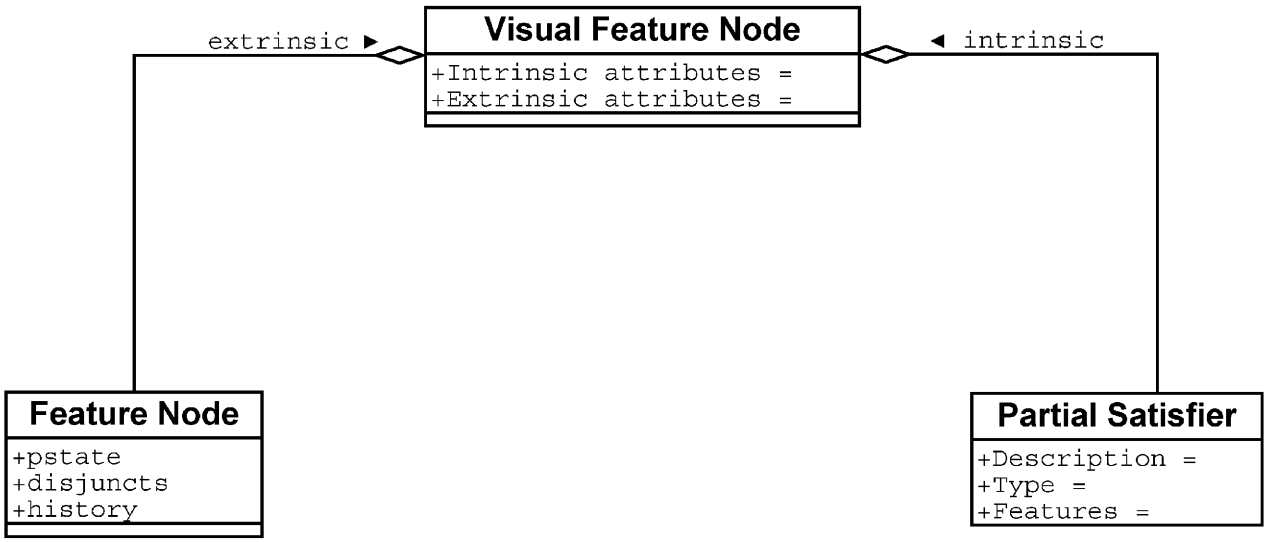

Fig. 2. The elements of the visual feature node map onto the domain layer constructs. The intrinsic ATTRIBUTES represent the formal features along which the exploration may proceed. The extrinsic attributes represent designer moves.

nal complex datatype. The value of MASS_EL may also be another feature structure.

In a visual feature node, VNode, two or more paths can share the same information. This is called *structure sharing*. Paths engaging in structure sharing are called reentrant. Shared structure in a visual feature node is represented by *coreferences*, also called *tags*. The coreference *n* denotes an index value where *n* is the identity of the node that is shared between one or more feature structures. Coreferences and their denotation by indices is straightforward in the AVM notation. As shown in Figure 4, reentrancy, or structure sharing is indicated by reference tags such as 1. The slots are the features and the values are written next to them.

A model of turn taking is proposed for synchronizing the modalities of action available to both the user and formalism during exploration. Through interaction with the intrinsic and extrinsic attributes of a visual feature node, the user and the formalism are able to acquire, relinquish, shift, and allocate initiative during the exploration process.

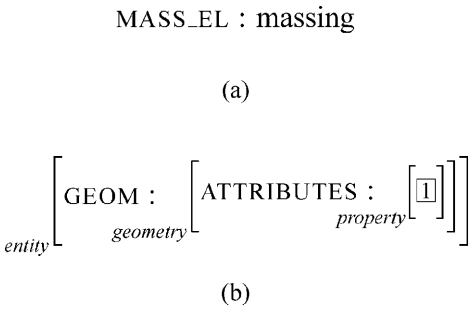

Fig. 3. (a) A feature-value pair and (b) a feature-value map are shown. The pair ATTRIBUTES:property is indicated by a coreference tag, indexed by the number 1, to a node outside of the diagram fragment shown.

3.2. Choices

The notation supports user choices in the VNode through the representation of alternatives, resolution steps, and function applications.

The notation incorporates the operators of the description language: *conjunctions* and *disjunctions*. Conjuncts and disjuncts in feature structures are denoted using the same notation as feature value pairs. In place of the feature labels, the labels *conjunct* and *disjunct* are used, with the values as feature structures. This common representation can be scaled to represent the conjunction of disjuncts and the disjunction of conjuncts. In linguistic attribute-value formalisms (Pollard & Sag, 1987; Pollard & Moshier, 1990), conjuncts and disjuncts are denoted by special delimiters, such as "{" and "}," and their edge names are either omitted, suppressed, or obscured. This is not necessary in this interactive representation.

An example of a conjunct of disjunctive visual feature nodes is shown in Figure 5. User interaction is necessary to resolve the structures associated with the features MASS_EL and GEOM of the feature structure of type *entity*. The disjunctive nodes of type disjunct are represented as a feature-value map. Each feature-value map is accessed by a feature CONJUNCT, of type conjunct. Each feature-value map has four possible disjuncts, which are represented as a feature-value pair, whose features are defined by DISJUNCT *n* where *n* is an index over disjuncts.

The user can choose a single conjunct for each of the features of the node entity shown in Figure 5. For example, if the designer selects the disjuncts DISJUNCT_2 and DISJUNCT_7, shown in Figure 6, as the appropriate values of MASS_EL and GEOM, respectively, the node that results will carry the structure shown in Figure 6. Through the dialogue

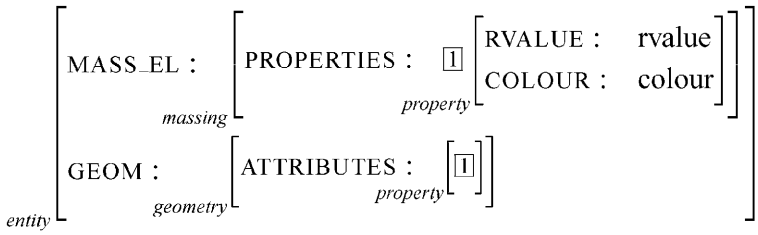

Fig. 4. A visual feature node incorporating coreference notation. The shared feature structure of type property is indicated by the coreference tag, indexed by the number 1.

layer construct, VNode, it is possible to expose the internal representation of descriptions (problem states), partial satisfiers (solutions), and alternatives (choices) for user interaction. Because the VNode is recursively defined, a collection of choices and user interaction history is expressed as a collection of visual feature nodes. Thus far, the notation has shown how the intrinsic attributes of a feature node FNode can be visually represented for direct manipulation by the designer. It is also possible to represent the attributes of a FNode that are extrinsic to the formalism, using the same notation.

For example, a function can be represented in the visual feature node. Functions can be encoded within the feature-value map representation such that the functor annotates the feature-value map and the arguments are features. An example of the duality of a function and its arguments with a feature node representation is shown in Figure 7. This representation allows the feature structure to encode traditional command languages found in geometry-based design systems. The specification of a command or function then returns a value, which can be atomic, complex, or a feature structure. The use of feature structures to encode functions can also be used to pass commands.[3]

The expressiveness of feature structure command representations needs to address the possibility of cyclic feature structures and structure sharing. A cycle arises when following a nonempty sequence of features out of a node leads back to that node, which is a useful property in the finite modeling of knowledge (Carpenter, 1992, pp. 51–52). A recognition mechanism is necessary to interrupt infinite loops

[3]Programming languages like LIFE (Aït-Kaći & Cosmo, 1993) use types for commands.

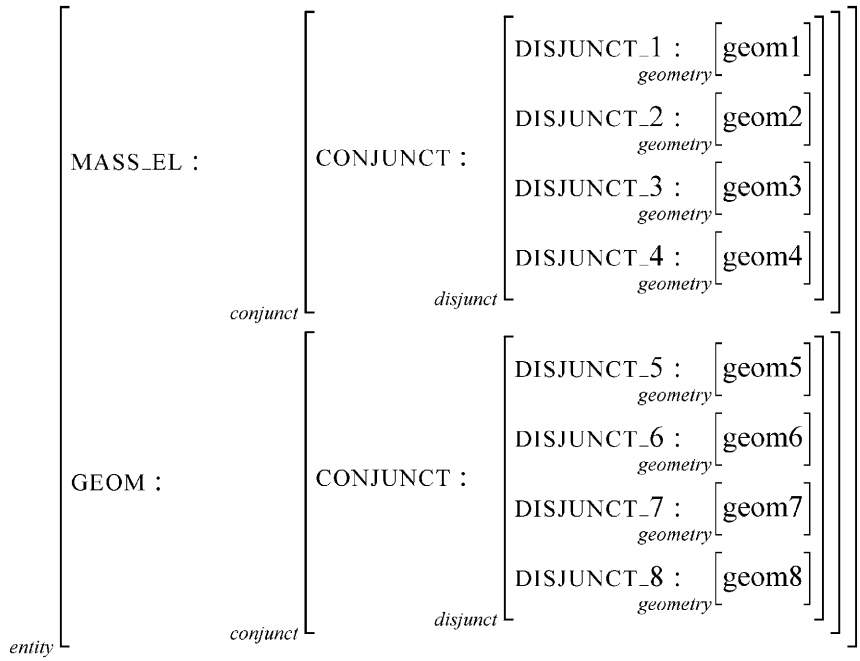

Fig. 5. An example of a conjunct of disjunctive visual feature nodes. The disjunctive nodes are accessed by a node of type conjunct and represented as a feature-value map and each disjunct is represented as a feature-value pair, whose features are defined by DISJUNCT *n* where *n* is an index over disjuncts.

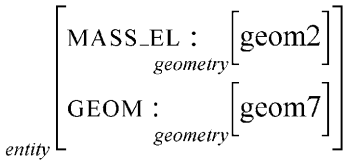

Fig. 6. The resultant visual feature node arising out of the resolution of disjunctive nodes by user interaction.

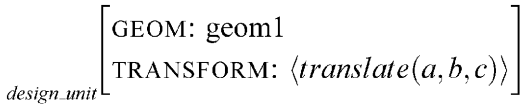

Fig. 8. The function translate(a, b, c) is represented as a feature-value pair and contained within a visual feature node, with feature TRANSFORM and value translate(a, b, c).

in a visual feature node representation for commands. The restriction on commands is that structure sharing is not considered a valid part of the command syntax. If coreferences do occur, the structures they represent are copied uniquely within each command.

Another way of visualizing functions within feature structures is to encode the functional definition as the value of a feature-value pair. In this scheme, for a function append(X, Y) with two arguments, a type *function* exists, such that its value is a function definition with the syntax append(X, Y).

An example of a procedural function in visual form is shown in Figure 8. User interaction on this node involves three possible behaviors. First, the application of the function to an appropriate node results in a new node, consistent with the application. Second, the unification of a functional node with an appropriate feature, results in a new feature structure, following the laws of unification for typed feature structures. Third, the function can be unfolded into its constituent subparts following the interaction defined above and its values subject to exploration. An example of the latter is shown in Figure 9. The unfolding of a functional representation shows that the feature node representation of the function translate(a, b, c) is of type translate and the arguments are the three feature-value pairs, TX, TY, and TZ.

3.3. Interaction with visual feature nodes

The visual feature node representation is extended to incorporate behaviors that admit user level actions extrinsic to the formalism. Interaction with a large collection of visual feature nodes requires functionality for panning, zooming in and out of context, search, and the expansion of tags. Visual feature nodes are nested entities. User navigation of a large collection of feature nodes is enhanced by functionality for zooming in and out of nodes, imploding nested nodes, and the expansion of coreference tags. If the nesting is very deep or broad, panning functionality provides the ability to scroll through the whole feature node. Zoom and implode interaction behavior provide functionality for controlling depth nesting. By using this functionality the user can fold (implode) and unfold (zoom) feature nodes. The user obtains information about substructures of a node by zooming into them. Further, zooming into the selected substructure enables a reorientation of context such that the selected node becomes the new *root node* of exploration. An example of unfolding substructures by user interaction is given in Figure 10.

3.3.1. Zooming and imploding nodes

Feature nodes can be nested recursively to arbitrary levels containing many substructures. The interactive mechanism accounts for folding the nested substructures of a feature-value map to hide their underlying notation and for unfolding the imploded structure to see the details of a feature value map. In the visual representation, this is realized by allowing users to open or close substructures visually through the symbol + (Fig. 11)

The unfolding symbol + shows up in two different situations. A restriction may be placed on the depth of display of a feature-value map. Any substructure in a feature-value map that exceeds that depth, is represented by the symbol +. This is automatically managed by the dialogue layer and the nesting levels set through preferences. The user can

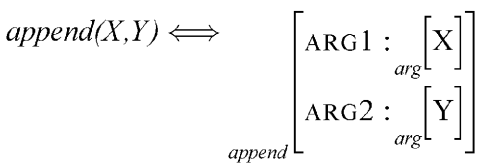

Fig. 7. Encoding a function as a feature-value map. The function append (X, Y) which concatenates values can be represented as the feature-value map of type append and the two features ARG1 and ARG2. The features, ARG1 and ARG2 encode the values X and Y as two feature-value pairs.

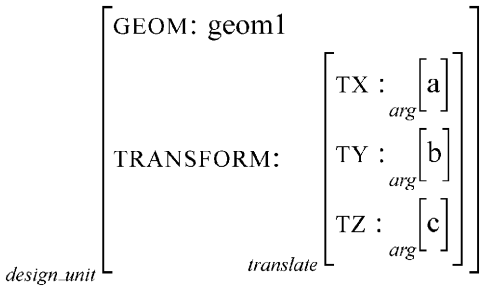

Fig. 9. An unfolding of a functional representation shows the feature structure notation of the function translate(a, b, c). The type of the function is translate. The arguments are unfolded into the three feature-value pairs, TX, TY, and TZ.

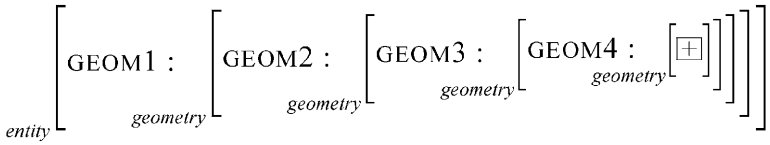

Fig. 10. An example of an imploded feature node hiding the contained substructures. The symbol + indicates that the substructures of the feature-value map of type entity are closed. User interaction on this node is necessary to open these hidden structures.

also manipulate the feature-value map interactively. The feature-value map will be shown as folded, until it is explicitly unfolded. Thus, the + symbols on the feature-value map coming from depth restriction are generated and removed dynamically while the user navigates a feature structure. In contrast, the maps that are unfolded manually need explicit interaction to change their display. This enables the user to control the level of detail shown, while zooming and imploding very large feature node collections.

Pan functionality is provided by enclosing the feature structures in scroll bars. This is a standard means for providing canvas real estate. User interaction with the scroll bars allows context to be shifted horizontally and vertically.

3.3.2. Interaction with coreference tags

Path equality in structures is one of the oldest information structuring concepts in computational design, the first instance being Sutherland (1963). This notion is captured through structure sharing between feature structures at the formal substrate.

From the perspective of the designer, path equality in the underlying formalism is visually displayed through coreference tags. Thus, the appearance of coreference tags in the visual representation indicate nodes that are strictly structure shared partial satisfiers. In the context of exploration, these correspond to entities that are feature values that, by definition, are feature structures.

Coreference tags are used in two ways, both shown in Figure 12. First, a coreference is used to annotate a feature-value pair that structure shares a feature-value map. Second, it is used to simplify the visual representation of partial satisfiers, by simple substitution of the shared feature-value map by the coreference tag, denoted by *n*. The coreference tag can be substituted by the partial satisfier it denotes by user interaction. If the partial satisfier is represented, the coreference appears outside the feature-value map, as shown in the value of PROPERTIES. If the coreference is used to refer to the partial satisfier, it appears inside the feature-value map as shown in the value of ATTRIBUTES.

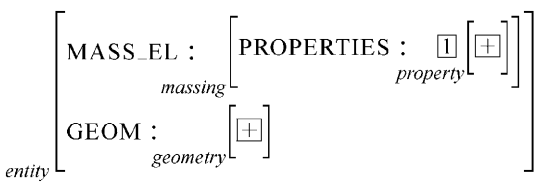

Fig. 11. Another example of an imploded feature node. The symbol + indicates that the feature-value map of type property and geometry contains nested subnodes that can be unfolded by user interaction.

The user can move within a collection of nodes using the find operation. Search for a nested node using the find operation is supported by the use of tags. A successful search results in the matching feature-value pair being panned into focus. The user can specify the *find* function to search only for features, types, coreferences, or atomic values. In the case of coreference search, the tag number is used to locate the partial satisfier corresponding to the specified tag. This is useful in dialogue situations when structure shared partial satisfiers are widely separated. Tags mark structure shared nodes and these can be expanded *in situ* following the general convention of appearing in the first location in which they are introduced. When tag expansion occurs during dialogue, the coreferences are updated and the partial satisfiers that they represent are redisplayed. These extensions are discussed in the next sections.

4. INTEGRATION OF DIALOGUE

The second requirement, dialogue integration, combining two modalities into a common frame, is described in this section. Having defined the visual feature node, its representation and its support for interaction, it is now possible to address how this construct supports dialogue integration.

Dialogue integration utilizes the unification operation. Unification is an appropriate basis for mixed-initiative integration as it can combine complementary input or redundant input from both modalities of exploration. Further, in the case of contradictory inputs, unification can rule out the possibility of integration. A feature node consists of a collection of feature-value pairs. The value of a feature may be an atom, a variable, or another feature-value map. When two maps are unified, a composite map containing all of the feature specifications from each component structure is formed. This is subject to the restriction that any feature common to both feature structures must not clash in value. If the values of a common feature are atoms, they must be identical. If one is a variable, it becomes bound to the value of the corresponding feature in the other feature structure. If both are variables, they become bound together, constraining them to always receive the same value. If the values themselves are feature structures, then the unification operation is applied recursively.

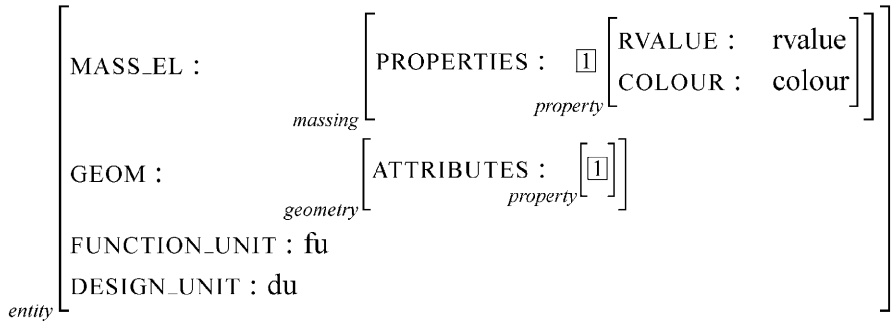

Fig. 12. An example of substitution of a feature-value map with a coreference tag 1. The coreference tag is an index to the nested partial satisfier of type property that is shared by the features PROPERTIES and ATTRIBUTES.

4.1. Supporting partiality

As shown in Section 3.1, a visual feature node (VNode) inherits key behavioral properties of typed feature structures. Thus, given the well-defined semantics of types, features, and descriptions, the visual feature node supports partiality of input and output. The representation framework of visual feature nodes, type annotated frame delimiters, feature-value pair/feature-value map, and coreference tags carry the specification of partial information. Partial information during dialogue provides the opportunity for both user and formalism to underspecify, relying on the extension of dialogue through turn taking. A partially specified exploration move is represented as an underspecified visual feature node. In this situation, a subset of feature-value pairs is not instantiated, following the intentional nature of feature structure representation. Instead, they are assigned a certain type, corresponding to the semantics of the move. This bears explanation through an example of turn taking on a partial specification.

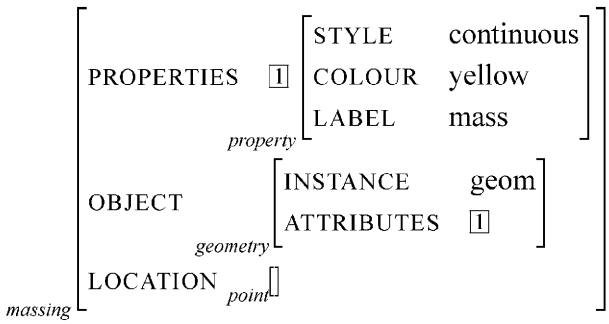

Fig. 13. The visual feature node for an underspecified entity of type massing. Substructures can be shared in a visual feature node. For example, the PROPERTIES of massing are the same as the ATTRIBUTES of geometry. Further, the specification of property can be changed either through the PROPERTIES of massing or through the ATTRIBUTES of geometry.

An example of supporting partiality in mixed-initiative dialogue is explained in Figure 13. In this example, a given description can be integrated with a massing of type geometry, the resultant is assigned an underspecified location feature, whose value is required to be of type point. The description, massing, is assigned to the visual feature node shown. In this scheme, it is possible to state during exploration that there exists an entity of massing with three features PROPERTIES, OBJECT, and LOCATION.

In the visual feature node representation, it is possible to specify the features PROPERTIES of type property and OBJECT of type geometry but not the feature LOCATION, which is assigned a generic type, point. Thus, the visual node captures the idea that there exists a geometric object of type massing and that its feature LOCATION is constrained to be of type point. Given the equivalence of structures and descriptions, the underspecified visual feature node shown in Figure 13 can be interpreted as a formal command for the generation of massing elements. More importantly, the possibilities for locating the massing element are open to mixed-initiative specification by the user's action on a visual feature node or through the resolution of formal constraints.

This is explained in Figure 14 with another underspecified feature node of type command, whose feature LOCATION is constrained to be of type point with specified coordinates. Note that this is only true under the assumption that feature nodes can carry maximal values such as the

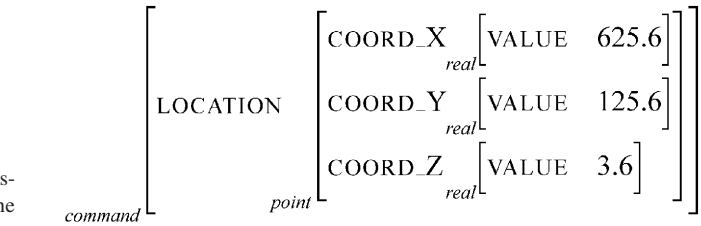

Fig. 14. The visual feature node for an underspecified entity of type command.

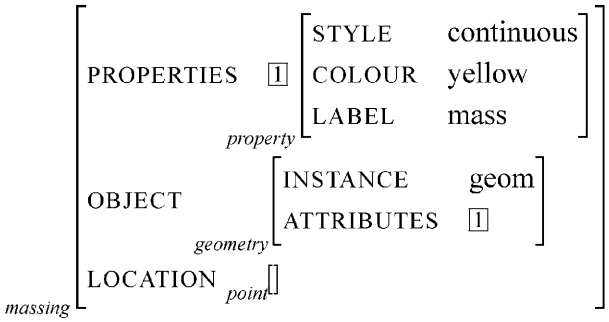

Fig. 15. The substructures can be shared in a visual feature node. For example, the PROPERTIES of massing are the same as the ATTRIBUTES of geometry. Further, the specification of property can be changed either through the PROPERTIES of massing or through the ATTRIBUTES of geometry.

coordinates of a location.[4] It follows from the above, that if "massing" from Figure 13 is compatible with command from Figure 14, the unification of the two will provide a feature structure whose location feature LOCATION will result in the more specific of the two values. To be compatible in type, the result must be the meet or a subtype of the meet of the two argument types. Hence, the result of a typed unification is a more specific feature structure or atom drawn from the type hierarchy. Thus, in a mixed-initiative scenario, the user might provide a location value for an underspecified geometry generated by the formalism. Alternatively, the user might specify a geometry for which the description formalism provides one or a number of possible locations in a current problem state. Through turn taking, both the location and geometry of an element in a solution state can be resolved.

4.2. Supporting structure sharing

Structure sharing is another fundamental property of typed features that plays a significant role during mixed-initiative dialogue. Visual feature nodes enable the user to develop specific exploration paths in great detail and then provide the resultant feature structure to the generator (Fig. 15). The generator can reuse the resultant feature structure multiple times in other feature node contexts through structure sharing. Recall that feature nodes compose feature structures (Section 3.1). Thus, at the visual feature node level, dialogue constructs can take advantage of structure sharing in their underlying feature structures.

Feature structures can be shared across a design space. By reusing shared feature structures, the user can converge information from other exploration paths into the current path of exploration. This bears explanation. For example, there may be two bathrooms in two distinct house designs that are identical. The feature node collections that represent the exploration steps of the first and second designs are distinct paths in the satisfier space. The problem states and choices made by the designer in both paths are also distinct states. However, a portion of the solution state, subsequent to exploration, is identical, in this case, the bathroom design.

In design space, this fact would result in the existence of two distinct feature structures containing the same information (feature structures are intensional). The feature node allows the designer to structure share the bathroom solution of the second design with the first. Once this equivalence is declared, the exploration structure of the first is converged into the exploration structure of the second design. It is important to note that two distinct visual feature nodes can represent two distinct exploration threads in satisfier space. Through structure-sharing dialogue, feature structures can be shared, reused, and converged in design space.

In Figure 13, the PROPERTIES of massing are the same as the ATTRIBUTES of geometry. Further, the specification of type property can be reformulated either through the feature PROPERTIES of type massing or through the feature ATTRIBUTES of type geometry. During exploration, the generative component can instantiate a massing element, with an unspecified feature ORIENTATION. Subsequently, the user might specify an orientation by direct manipulation, by drawing an arrow that has a value direction for its feature ANGLE. The process of typed feature structure unification enables the explorer to structure share the value of feature ANGLE of the feature structure of type direction with the value of ORIENTATION. Thus, when the two features are unified successfully, the resultant feature node of massing has a new value for the feature ORIENTATION. This value is now given by the more specific value of the feature ANGLE. Note that the property of structure sharing can be nested. Feature nodes are recursively contained through the feature-value map and feature-value pair relations of the notation. Structure sharing and its representation using coreference tags presents the notation with the ability to avoid redundant substructures.[5]

4.3. Supporting dialogue disambiguation

Mutual disambiguation is another property supported in the visual feature node. An exploration move that is partially specified is open to multiple interpretations. In such a situation, a collection of many feature-value pairs may be available for unification. For example, if a given description can be integrated with a massing of type GEOM, it can be assigned an underspecified LOCATION feature, whose value is required of be of type GEOM as shown in the previous discussion of partiality in Figure 13. In the description node, massing can

[4] In the design of GENESIS, Heisserman (1991, p. 133) notes that the inability to specify an intensional model of geometric information remains a major drawback for interactive systems for generative design.

[5] In the design of GRAMMATICA, Carlson (1993) notes that the ability to detect and represent duplicates remains a major challenge for interactive exploration of design spaces.

also be assigned a feature node of type location from a number of sources. For instance, the location might be constrained to be adjacent to a previously created entity of massing. Thus, it is possible that there exists not one, but a number of feature nodes of compatible type with type massing with the feature, LOCATION. The dialogue layer provides a mechanism for mutual disambiguation, such that it is possible to specify the feature LOCATION of any compatible type at the current state of exploration to disambiguate the choice. Given multiple options for interpreting the value of the feature node of type location shown in Figure 13, a disambiguation process is a necessary attribute of dialogue to resolve nondeterminism. The same process of dialogue disambiguation applies to disjunctive nodes. Given a number of possible alternative choices, the designer can disambiguate a disjunction by interaction with the visual feature node representing the disjuncts.

For example, in a user-driven query, the formalism might compute a range of possible feature nodes that are extensions of type location. The dialogue layer makes them available to the user as a collection of visual feature nodes. Alternatively, for an explorer-driven step, for instance, a command to create an entity of type massing, the type location can be disambiguated by the user through interactive browsing of the current state, selection of a current point, command input, structure sharing, or a constraint specification.

It follows from the above that when conflicts or multiple choices arise during exploration, mixed initiative in the dialogue layer can provide for mutual disambiguation through the visual notation.

In the visual feature node, this process of disambiguation is mutual. It can be interpreted either as a formal command for the generation of massing elements by the explorer or a user-driven process wherein the possibilities for locating the massing element are open to exploration through the specification by the user's action on the visual feature nodes.

Figure 16 shows a feature node of type massing, whose feature LOCATION is constrained to be of type point. The feature value can be resolved either by the user or by the formal generator. It follows that if the location feature of massing is compatible with one or more locations of compatible location, the unification of the two can be resolved through a process of mutual disambiguation.

In the example of disambiguation shown here, there are several partial interpretations: one for massing, and two for location. Because either of the two locations might be equally valid for the unification to succeed, only a process of mutual disambiguation can isolate the valid choice of location. The dialogue layer allows either of the above and does not distinguish between the modality of exploration, direct specification by the user or constrained search by the generator. In each case, the unification-based integration strategy ensures that mixed-initiative exploration compensates for exploration nondeterminism through type constraints on the values of features. Further, the restrictions imposed on these values ensure that the exploration maintains integrity and consistency during the process of disambiguation.

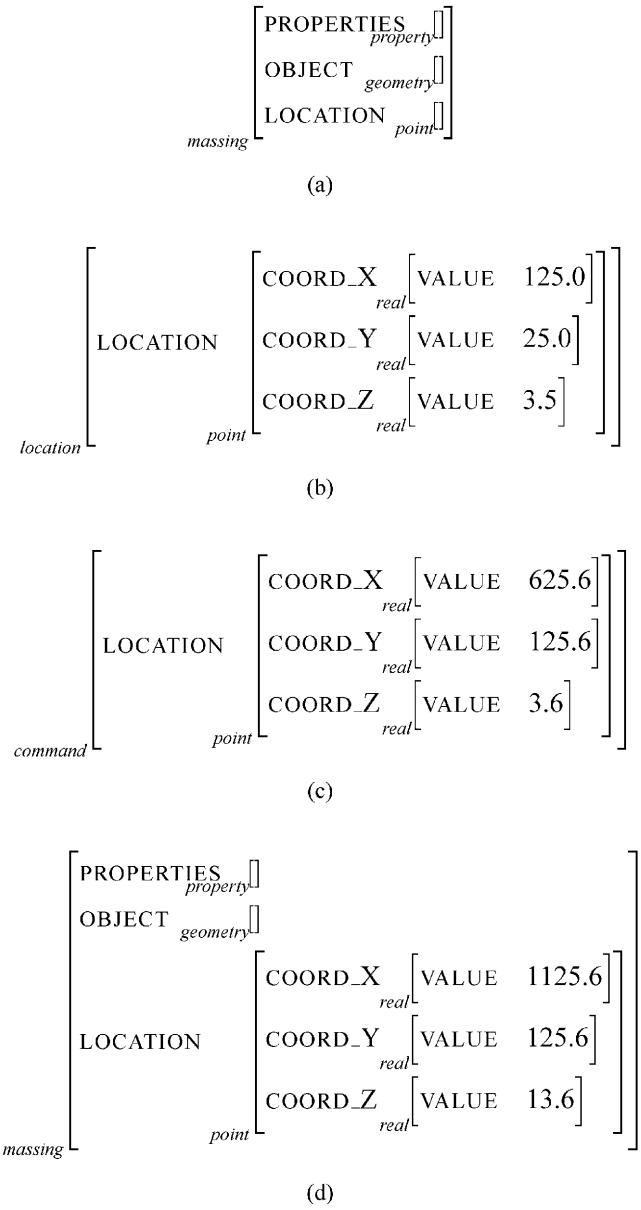

Fig. 16. (top) The feature node of type massing with two options location and command for disambiguation of feature LOCATION. (bottom) The feature node showing LOCATION after disambiguation.

5. DISCUSSION

The introduction of conversational structure through mixed initiative to design space exploration enables the designer to maintain exploration freedom, preserves the underlying structure of exploration, and permits a finer granularity of dialogue. The characteristics of the model of dialogue presented here can be summarized as follows:

1. Maintenance of exploration freedom: The dialogue model maintains freedom for incorporating the intentional actions of the designer at any state of exploration. The visual feature node provides a unified model for representing the set of problems, subproblems, problem revisions, and associated designs that a designer actually considers. Problems need not be fixed. Designs can be partial or complete with respect to the initial problem formulation. A designer may make varied choices that imply different kinds of design space operations.
2. Preservation of order: The dialogue model enables order preserving exploration. The structure of exploration is represented through the ordering relation of subsumption. The concept of an ordered design space underpins interaction between the designer and the description formalism. In it, the collection of exploration states are ordered by the relation of subsumption. Exploration moves are cast in terms of moves in a design space upward or downward in an information ordering. Mixed initiative provides a principled way for keeping track of additions, deletions, and other forms of change as the exploration progresses without negating the underlying subsumption ordering of possible states.
3. Granularity of dialogue: The dialogue model permits a finer granularity of interaction between the designer and the formalism. It supports incrementality and turn taking in the process of exploration dialogue. Through incrementality, emphasis is shifted from the final results of exploration to its intermediate constructive steps. Through turn taking, the formal movement algorithms are made accessible to the designer. The incremental, turn taking model of interaction permits a sound treatment of exploration nondeterinism, where disjunctions in description queries, alternative constraints, conflicts, and errors can be resolved by user intervention.

6. SUMMARY

This exposition of interaction with formal systems for design support has focused on conversational structure as a model of dialogue. However, as Woodbury and Burrow show, integrating human users and computational formalisms remain a key problem for design space exploration. As borne out by the examples in their paper, the visual feature node presented here demonstrates the case for designer action but remains a limiting case for exploration. The visual feature node representation must allow for what Woodbury and Burrow term "amplification."

The two requirements posed in developing the above dialogue model, dialogue representation and dialogue integration, are addressed through the development of an intrinsic and extrinsic logic for unfolding the components of the visual notation. The feature structure representation and its interaction logic are brought together through the construct, the visual feature node. Framed by Grice's maxims of conversational rationality, the designer is able to participate in a limited dialogue with the description formalism to construct problems, navigate solutions, make choices, and record the history of exploration.

As Woodbury and Burrow state, the design space must be admitted as a primary object of research in the field. To do so, the configuration, structure, and visualization of design space requires further work and need to be addressed. The findings at this stage also indicate a clear demarcation between the discrete modes of thinking about generative design (rules, types, features) and continuous modes of manipulation known from interaction with real-world geometric design problems. This is hinted at on several occasions in Woodbury and Burrow's paper, but no clear direction has emerged. Dialogue needs to be extended from a conversational mode (as described in this paper) to a language designers understand best, visual representations, and geometry. Thus, although some advances have been made in understanding joint responsibility over discrete moves between highly structured states, a clear formulation of mixed initiative (the rationality of dialogue between a formalism and a human designer) in the joint exploration of nontrivial design geometry remains an open problem.

ACKNOWLEDGMENTS

The author acknowledges the continuing contribution of Robert Woodbury (Simon Fraser University, Canada), Tengwen Chang (Ming Chuan University, Taiwan), and Andrew Burrow (RMIT University, Melbourne) in the development of the interaction model for design space exploration reported in this paper.

REFERENCES

Aït-Kaći, H., & Cosmo, R.D. (1993). *Compiling Order-Sorted Feature Term Unification*. Technical Report No. PRL-TN-7. Paris: Digital Paris Research Laboratory.

Allen, J. F. (1999). Mixed initiative interaction. *IEEE Intelligent Systems 14(6)*, 14–23.

Allen, J.F., Ferguson, G., & Schubert, L.K. (1996). Planning in complex worlds via mixed-initiative interaction. In *Advanced Planning Technology: Technological Achievements of the ARPA/ROME Laboratory Planning Initiative* (Tate, A., Ed.), pp. 53–60. Menlo Park, CA: AAAI Press.

Borgida, A., Brachman, R.J., McGuinness, D.L., & Resnick, L.A. (1989). CLASSIC: a structural data model for objects. *Proc. 1989 ACM Special Interest Group on Management of Data (Sigmod) Int. Conf. Management of Data*, pp. 58–67.

Burrow, A.L. (2003). *Computational design and formal languages*. PhD Thesis. University of Adelaide, Department of Computer Science.

Burrow, A.L., & Woodbury, R. (1999). π-Resolution and design space exploration. *Computers in Building: Proc. CAADF '99 Conf.* (Augenbroe, G., & Eastman, C., Eds.), pp. 291–308. Dordrecht: Kluwer Academic.

Carlson, C. (1993). *Grammatical programming: an algebraic approach to the description of design spaces*. PhD Dissertation. Carnegie Mellon University.

Carpenter, B. (1992). *The Logic of Typed Feature Structures with Applications to Unification Grammars, Logic Programs and Constraint Resolution*. New York: Cambridge University Press.

Datta, S., & Woodbury, R. (2002). A graphical notation for mixed-initiative dialogue with generative design systems. In *Artificial Intelligence in Design '02* (Gero, J.S., Ed.), pp. 25–40. Dordrecht: Kluwer Academic.

Ferguson, G., & Allen, J.F. (1994). Arguing about plans: plan representation and reasoning for mixed-initiative planning. *Proc. 2nd Int. Conf. AI Planning Systems*, pp. 43–48. Chicago.

Franz, B., & Jrg, S. (1994). Unification theory. In *Handbook of Logic in Artificial Intelligence and Logic Programming* (Gabbay, D., Hogger, C., & Robinson, J., Eds.). Oxford: Oxford University Press.

Grice, H. (1989). *Studies in the Way of Words*. Cambridge, MA: Harvard University Press.

Grice, H.P. (1975). Logic and conversation. In *Syntax and Semantics* (Cole, P. & Morgan, J., Eds.), Vol. 3, pp. 41–58. New York: Academic.

Hakim, M., & Garrett, J.H.J. (1993). Using description logic for representing engineering design standards. *Journal of Engineering with Computers 9(1)*, 108–124.

Heisserman, J. (1991). *Generative geometric design and boundary solid grammars*. PhD Dissertation. Carnegie Mellon University, College of Fine Arts, Department of Architecture.

Kasper, R.T., & Rounds, W.C. (1990). The logic of unification in grammar. *Linguistics and Philosophy 13(1)*, 35–58.

Kiefer, B., & Fettig, T. (1995). *FEGRAMED—An interactive graphics editor for feature structures* (Research Report No. RR-95-06). Saarbrücken, Germany: German Research Center for Artificial Intelligence (DFKI).

Knight, K. (1989). Unification: a multidisciplinary survey. *ACM Computing Surveys 21(1)*, 93–124.

Kochhar, S. (1994). A paradigm for human–computer cooperation in design. *Computer Graphics and Applications 17(16)*, 54–65.

Lambrix, P. (1996). *Part-whole reasoning in description logics*. PhD Dissertation. Lingköping University.

Minsky, M. (1975). A framework for representing knowledge. In *The Psychology of Computer Vision* (Winston, P.H., Ed.), pp. 211–277. New York: McGraw–Hill.

Piela, P. (1989). *ASCEND, an object-oriented computer environment for modeling and analysis*. PhD Dissertation. Carnegie Mellon University, Department of Chemical Engineering.

Piela, P., McKelvey, R., & Westerberg, A. (1993). An introduction to the ASCEND modeling system: its language and interactive environment. *Journal of Management Information Systems 9(3)*, 91–121.

Pollard, C., & Sag, I. (1987). Information-based syntax and semantics. In *Fundamentals*, Vol. 13. Stanford, CA: Stanford University, Center for the Study of Language and Information.

Pollard, C.J., & Moshier, M. (1990). Unifying partial descriptions of sets. In *Information, Language and Cognition* (Hanson, P., Ed.), Vol. 1, pp. 285–322. New York: Oxford University Press.

Schulte, C. (1997). Oz Explorer: A visual constraint programming tool. In *Proc. Fourteenth Int. Conf. Logic Programming* (Naish, L., Ed.), pp. 286–300. Leuven, Belgium: MIT Press.

Sutherland, I.E. (1963). Sketchpad—a man–machine graphical communication system. *Proc. Spring Joint Computer Conf.*, Vol. 23, pp. 328–346.

Woodbury, R., Burrow, A., Datta, S., & Chang, T. (1999). Typed feature structures and design space exploration. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 13(4)*, 287–302.

Woodbury, R., Datta, S., & Burrow, A. (2000). Erasure in design space exploration. In *Artificial Intelligence in Design '00* (Gero, J., Ed.), pp. 521–544. Dordrecht: Kluwer Academic.

Zeller, A. (1997). *Configuration management with version sets—a unified software versioning model and its applications*. PhD Dissertation. TU Braunschweig.

Zeller, A., & Snelting, G. (1995). Handling version sets through feature logic. *Proc. 5th European Software Engineering Conf. (ESEC)* (Schfer, W., & Botella, P., Eds.), *Lecture Notes in Computer Science 989*, pp. 191–204.

Sambit Datta is presently a Lecturer in the School of Architecture, Landscape Architecture, and Urban Design at Deakin University. He holds a Diploma in architecture from the School of Architecture, CEPT, India, a Master's in architecture from the National University of Singapore, and a PhD from the University of Adelaide. Dr. Datta's current research is on developing interaction techniques for generative formalisms, with a particular focus on discrete representations.