Foundations for Active Multimedia Narrative: Semiotic spaces and structural blending

Joseph A. Goguen and D. Fox Harrell
Department of Computer Science & Engineering
University of California, San Diego

Abstract: Multimedia technology is advancing rapidly, and already supports powerful forms of interactive media experience. Yet algorithms and theory to support the real-time production of new content remain primitive. To address this gap, we develop computational models for application to new genres of gaming, entertainment, and art, drawing on research from socio-linguistics on how humans structure narrative, from cognitive linguistics on metaphor and blending, and from algebraic semantics and semiotics, on how to represent and manipulate such structures. We describe our formal models, report results of experiments with our blending algorithm, discuss applications such as active poetry and interactive song, and sketch some directions for future research.

1 Introduction

Narrative is the backbone of most entertainment, and of many games and art works, affording more involving experience, by providing a sense of direction and purpose. Narrative links together events that would otherwise be isolated, into a chain of temporal succession, causal connection, and shared values; note that narrative in this sense need not be explicit. It has been argued that narrative is fundamental in creating social roles and institutions, and in maintaining our sense of community and even our sense of self [25].

Computational narratives are media objects with narrative structure enabled by computer technology. Computational narratives can emulate previous media, but they also present new possibilities. We are especially interested in *active media*, which generate new content on the fly. A digitized film viewed with DVD technology is only trivially interactive, in allowing scenes from a predetermined sequence to be viewed out of order, and it is also not particularly interesting to observe that texts are understood at least slightly differently each time they are read. A more productive challenge is to create coherence during the unfolding of narrative structure in real time.

Unfortunately, "multimedia research" is often considered only to refer to hardware research. Although current multimedia hardware is very powerful, e.g., for video games, support for new narrative content lags far behind, despite the fact that it is necessary to realize the full potential of hardware. It will help us to distinguish among multimedia hardware, genres for using such hardware, and particular works. A *genre* is a style of using some input/output capabilities, and a *work* is an instance of a genre. This paper is concerned with developing new theory and technology to support the production of works in new genres for active media. For this, we apply Labov's theory of narrative [21], algebraic specification theory [13], the algebraic semiotic theory of representation [9], and (a generalization of) the conceptual blending theory of Fauconnier and Turner [7, 8], which studies how concepts combine to form new concepts. An algorithm to compute such blends is described in Section 3 below.

We are especially interested in computational narratives where user interaction drives the meaning, rhetorical slant, content, and even the configuration of a work. In such narratives, user actions are basic elements that authors can include in composition, allowing direct user participation in constructing both the form and the content of a particular instantiation of an active work. One example is digital games that can be played each times with a different story line, based on user actions having meaningful effects in the game world. Another example is multimedia works in which user interaction triggers the generation of new metaphors within a world defined by the author. Thus, Section 4.2 describes an interactive poem with a fixed network of places and events, presented using metaphors generated on the fly, as users navigate the network. An interesting possibility is the reconfiguration of form, since the structure of a computational narrative can be developed more flexibly than allowed by usual techniques such as traversing a tree structure organized by plot or character point of view.

The OBJ code in this paper has evolved, first running under OBJ3 and later BOBJ; we found that it is easy to make errors, even very simple errors, when only informal notation is used, and it can be very difficult to uncover subtle errors. This is an important motivation for using a formal, executable notation. In addition, when the notation has a formal semantics, it enables the underlying theory to be debugged, as well as the examples, by raising questions about the notation; our experience shows that this can be very useful, since sound theory is essential for sound implementation. One interesting phenomenon discovered this way is the relation of type casting to metaphor, as discussed in Section 2.8. Despite this use of formal logical notation, we are not interested in, and are not doing, what is sometimes called "good old fashioned AI." On the contrary, we oppose logical reductionism, and subscribe to current views in cognitive linguistics, ethnomethodology and elsewhere, that language and thought are situated, embodied, and enacted [33, 34, 8].

It has been difficult to write this paper, and difficult to do the research which it reports, due in part to the number and diversity of fields involved. These include linguistics, psychology, sociology, computer science, mathematics, and art. Each field has its own interests, methods, concepts, and results, which impose constraints on research that builds upon it. For example, our blending algorithm is based on a mathematical model that must consistently generalize results of cognitive linguistics, must be mathematically correct and if possible even elegant, must be applicable to user interface design, and tries to be faithful to the spirit of semiotics. In addition, the algorithm itself must be capable of efficient execution in order to support applications, even if this means it only computes approximations. Socio-linguistics and ethnomethodology impose another set of constraints, and in particular, rule out the solipsistic models of traditional artificial intelligence.

Section 2 of this paper contains theoretical foundations, with Section 2.1 describing Labov's theory of narrative, which includes both structure and values [21, 25]; we provide a grammar that formalizes the structural aspects of this theory. Section 2.2 reviews some work of Lakoff and others [23, 22] on metaphor and image schemas, and Section 2.3 reviews work of Fauconnier and Turner [7, 8] on conceptual spaces and conceptual blending. Image schemas and metaphors provide texture for stories, while conceptual spaces provides content, and blending helps to maintain coherence and generate new content. Section 2.4 summarizes the ideas from algebraic semantics that we need [13, 15], and introduces the BOBJ algebraic specification language [14, 17, 13], in which some of our examples are expressed. Then Section 2.5 describes algebraic semiotics [9, 10], a formal theory of complex signs based on algebraic semantics. Two main concepts here are semiotic spaces and semiotic morphisms. Semiotic morphisms are treated in Section 2.6; these generalize the conceptual spaces and conceptual mappings of Fauconnier and Turner, to take account of structure and give a theory of representation, with some principles for comparing the quality of representations [9].

Section 2.7 extends conceptual blending to take account of structure, i.e., of how complex things are built from simpler parts. Section 2.8 gives an extended example of blending using the BOBJ language, based on an example in [9], and Section 2.9 shows how to view narrative and natural language generation as structural (not conceptual) blending.

Section 3 describes our research on blending algorithms, with Section 3.1 on the algorithm, and Section 3.2 on the results. Our experiments have shown the need for principles that identify high quality blends and reduce the search. These "optimality principles" include the degree to which various features are preserved (Section 2.7). Section 4 describes some active media projects, including in Section 4.1 an experiment by Fox Harrell with active poetry generation [19], and proposing in Section 4.2, an interactive multimedia poem based on a conventional (but powerful) poem by Pablo Neruda. We compare our work with some early work on computer generated poetry in Section 4.3, and conclude the paper in Section 5 with a summary of what we have learned so far, and some problems that remain.

2 Foundations

This section briefly reviews several areas that provide foundations for the work of this paper. We begin with narrative, then turn to conceptual spaces and conceptual blending, algebraic semantics and algebraic semiotics, and conclude with a treatment of structural blending.

2.1 Narrative

Narrative provides the basis for a deeper and more satisfying involvement, for most entertainment, and for many games and art works. Temporal and causal succession are essential for narrative, but values also play a key role, by connecting events in the story to the social worlds and personal experiences of users. These two aspects of narrative provide the sense that a work is "going somewhere" and that it "means something," respectively. Sociolinguists William Labov [21], Charlotte Linde [25] and others have studied oral narratives of personal experience, in which the narrator is an agent, and the story is told orally to a group of peers under natural conditions. The following briefly summarizes the structure of such narratives:

1. There is an optional **orientation section**, giving information about the time, place, characters, etc. in what will follow.
2. The main body of the narrative consists of a sequence of **narrative clauses** describing the events of the story; by a default convention, called the **narrative presupposition**, these are taken to occur in the same order that they appear in the story. The narrative clauses are usually in the past tense.
3. Narrative clauses are interwoven with **evaluative material** relating events to the narrator's value system, which is presumed shared with the audience. Evaluative material often appears in separate clauses, but may also take the form of repeated words, unusual syntactic or lexical choice, etc.
4. There is an optional **closing section**, summarizing the story, or perhaps giving a moral.

An additional principle is **accountability**, that the person telling such a story must establish to the audience the relevance of the actions reported. This is accomplished by including evaluative material which relates the narrative events to shared social values, and it provides a warrant for the authenticity of the values involved.

These claims are thoroughly grounded in empirical research and linguistic theory, and are relatively precise, especially in comparison with traditions such as critical literary theory and post-structuralism. Although strictly speaking, the theory only applies to naturally occurring oral narratives of personal experience, it can still yield insights into other forms, such as novels or human computer dialogues, because oral narratives of personal experience are foundational to many other forms. It may be surprising that values are an integral part of the internal structure of stories, rather than being confined to a "moral" at the end. But naturally occurring stories really do embody evaluative material, in many different ways, including explicit justifications for the narrator's choice of what to tell, or a character's choice of what to do. Moreover, evaluation also appears more implicitly, for example, via emphatic words, such as "very" or "extremely."

The narrative presupposition is a *convention*, not a necessity; for example, Alton Becker [3] shows that in Balinese narratives, if no special care is taken then the events reported in a sequence of narrative clauses are taken as occurring *simultaneously* rather than sequentially (a computer scientist might say that the default connective for a narrative sequence in English is sequential composition, whereas in Balinese it is parallel composition). The interpretation of narrative also employs the **causal presupposition**, which says that, other things being equal, given clauses in the order A, B we may assume that A causes B. (For example, "You touch that, and you die.") The default narrative presupposition can be overridden by providing explicit markers of other temporal relations, such as flashbacks and flashforwards. Moreover, some narratives may involve multiple times, places, and narrators, but will still be composed of subsequences that conform to the above structure.

The structural aspects of this theory can be formalized as a grammar, the instances of which correspond to the legal structures for narratives. The following uses a grammatical notation called "extended BNF,"

```
((<Abs> + <Ornt>) <Eval>*)* (<Cls> <Eval>*)* [<Coda>]
```

where `[...]` indicates either zero or one instance of whatever is enclosed, where `+` indicates exclusive or, and where juxtaposition of subexpressions indicates concatenation. For example, the subexpression

```
((<Abs> + <Ornt>) <Eval>*)*
```

at the beginning of the first expression defines `<Open>`, the opening section of a narrative. So we can also write this grammar in the form

```
<Narr> ::= <Open> (<Cls> <Eval>*)* [<Coda>]
<Open> ::= ((<Abs> + <Ornt>) <Eval>*)*
```

The sort `<Narr>` is for narratives, and `<Cls>` is for narrative clauses, which we take as potentially including evaluative material, while `<Eval>` is for stand-alone evaluative clauses, `<Open>` is for the opening section, which may include an orientation and/or abstract, and `<Coda>` is for the closing section.

Regular expressions (of which BNF is a variant notation) are not sufficient for describing all aspects of narrative structure; for example, the above grammar fails to address the variety of ways in which evaluation can occur. Some alternatives to explicit evaluative clauses include repetition of words or phrases (which serves to emphasize them), noticeably unusual lexical choices (which may serve to emphasize, de-emphasize, or otherwise spin something), and noticeably unusual syntactic choices (which also may emphasize or de-emphasize). The grammar also does not address the ways in which narratives can be embedded in other narratives, e.g., via flashbacks, flashforwards, or characters telling a story within a story.

2.2 Metaphors and Image Schemas

George Lakoff, Mark Johnson, and others [23, 22] have shown that metaphors come in families, called **image schemas**, having a common theme. One example is MORE IS UP, as in "His salary is higher than mine," or "That stock rose quite suddenly." The source UP is grounded in our experience of gravity, and the schema itself is grounded in everyday experiences, such as that when there is more beer in a glass, or more peanuts in a pile, the level goes up. Some image schemas, including this one, are grounded in the human body, and are called **basic image schemas**; they tend to yield the most persuasive metaphors, and can be useful, for example, in user interface design [9]. Image schemas, and especially basic image schemas, can add depth and interest to narratives.

2.3 Conceptual Spaces and Conceptual Blending

Fauconnier and Turner have developed a theory known as **conceptual blending**, or **conceptual integration** [7, 8]. **Conceptual spaces** are a basic notion of this theory, building on Fauconnier's earlier notion of mental spaces [5]. Conceptual spaces are relatively small, transient collections of concepts, selected from larger domains for some purpose at hand, such as understanding a particular sentence. Mathematically, conceptual spaces are sets of "elements" and relation instances among them [9]; these elements and relations are not typed, and there are also no functions and no axioms. Fauconnier and Turner study the **blending** of conceptual spaces, to obtain new spaces that combine parts of the input spaces [7]. Blending is common in natural language, for example, in words like "houseboat" and "roadkill," and phrases like "artificial life" and "computer virus" are blends, and blending is claimed to be a basic human cognitive operation, invisible and effortless, but pervasive and fundamental, for example in grammar and reasoning.

This research in cognitive linguistics extends work on metaphor described in Section 2.2. For example, to understand the metaphor, "the sun is a king," we blend conceptual spaces for "sun" and "king," resulting in a new, blended space, together with **conceptual mappings** to it from the "king" and "sun" spaces. In contrast to traditional approaches, there is no direct mapping between the two original spaces, although there are some "cross space" identifications, certainly including the identification of the "sun" and "king" elements, so that they are the same element in the blended space, and perhaps including further identifications, depending on the particular language that produced the metaphor. Not all blends are metaphoric; **metaphoric blends** are asymmetric, in that one space, the **target** of the metaphor, is understood using only certain salient concepts from the other "source" space [18]. For example, aspects of "king" are "**blocked**" from mapping to the blend space – usually the sun does not wear a crown or charge taxes. Additional information needed to understand a blend may be recruited from other spaces, as well as from frames, which encode highly conventionalized information. **Conceptual integration networks** are networks of conceptual spaces and conceptual mappings, that are used in blending the component spaces for situations that are more complex than a single metaphor.

Fauconnier and Turner [8] give several principles that characterize optimal blends. Here are six of these "optimality principles," as discussed in [18]:

1. *Integration:* The scenario in the blend space should be a well-integrated scene.
2. *Web:* Tight Connections between the blend and the inputs should be maintained, so that an event in one of the input spaces, for instance, is construed as implying a corresponding event in the blend.
3. *Unpacking:* It should be easy to reconstruct the inputs and the network of connections, given the blend.

4. *Topology:* Elements in the blend should participate in the same sorts of relations as the counterparts in the inputs.
5. *Good Reason:* If an element appears in the blend, it should have meaning.

All these principles require human judgement, and so cannot be implemented in any easy or obvious way. However the Topology Principle, in the special case where the relations involved are identities, only involves structure rather than meaning. Hence it can be implemented, and indeed, it is part of our blending algorithm (see Section 3.1). There are also specialized optimality principles for some particular kinds of blend, including personification, oxymoron, and metonymy, such as:

6. *Metonymic Tightening:* Relationships between elements from the same input should become as close as possible within the blend.

2.4 Algebraic Semantics

Algebraic semantics has its origin in the mathematical foundations of abstract data type theory [16, 15]. This section introduces some basics of that theory, using the BOBJ language. Much more detail appears in the literature, e.g., [13, 15] for algebraic semantics, [14] for BOBJ, and [17] for its ancestor OBJ3[1]. Modules in BOBJ are called **theories**, of which we use two kinds, **data theories** and **loose theories**. Below is a simple data theory named `MEDIUM` which will be used in later examples to provide values for attributes of some objects in conceptual spaces.

```
dth MEDIUM is sort Medium.
  ops land water : -> Medium.
end
```

The keyword `dth` indicates a data theory. The keyword `sort` indicates a type declaration[2]. Next, the keyword `ops` indicates a declaration for operations, in this case, two constants, `land` and `water`, of the sort `Medium`, using the convention that constants are given as operations with no arguments; argument sorts come before the arrow, while the value sort comes after it. The sort and operation declarations together, possibly with some subsort declarations, are known as the **signature** of the theory. BOBJ imports the Booleans into every module by default, so that the two Boolean values and some standard Boolean operations are also included in `MEDIUM`.

We now give a loose theory, which will be used to classify certain items and relations in some other theories that will be blended.

```
th BASE is sorts Person Object.
  pr MEDIUM.
  op person : -> Person.
  op object : -> Object.
  op use : Person Object -> Bool.
  op on  : Object Medium -> Bool.
  eq use(person, object) = true.
endth
```

This theory imports the `MEDIUM` theory, as indicated by "`pr MEDIUM`" ("`pr`" is short for "protecting," which is the usual mode of importation [13]) and it then declares three new constants. The OBJ languages do not provide relations as such, but these can be represented by Boolean valued functions. This module declares two such relations, `use` and `on`, using the keyword `op`; each has

[1] "OBJ" refers to the language family, particular members of which include OBJ3 and BOBJ.

[2] The OBJ languages use the word "sort" to avoid the ambiguities of the word "type."

two arguments, the sorts of which are given before the arrow, while the value sort `Bool` of booleans comes after the arrow. Such a relation holds of its arguments if its value on those arguments is `true`, which can be expressed with an equation, as in the above theory. **Constructors** are operations that build new elements from their arguments; the theories above do not have any constructors, but we will see examples of these later.

The models of a loose theory are all the structures (called **algebras**) that provide a set for each sort symbol, and a function for each operation symbol in the signature, and that satisfy all the equations in the theory. For data theories, there is just one intended model up to isomorphism, consisting of the minimal (or "free") set of terms generated over the signature. In the case of `MEDIUM`, this yields just the two constant terms, `land` and `water`, a so-called enumerated type.

There is a basic duality between theories and models: A semiotic theory determines the class of models that satisfy it, which we may call its **model space**[3]; and a class of models has a unique (up to equivalence) most restrictive theory whose models include it. This duality is the source of some interesting phenomena, as well as some confusion.

2.5 Algebraic Semiotics

Before briefly discussing algebraic semiotics, it may be helpful to be clear about its philosophical orientation. The reason for taking special case with this is that, in Western culture, mathematical formalisms are often given a status beyond what they deserve. For example, Euclid wrote, "The laws of nature are but the mathematical thoughts of God." Similarly, "situations" in the situation semantics of Barwise and Perry, which are similar to conceptual spaces (but more sophisticated – perhaps *too* sophisticated) are considered to be actually existing, ideal Platonic entities [2]. Somewhat less grandly, one might consider that conceptual spaces are somehow directly instantiated in the brain. However, the point of view of this paper is that such formalisms are constructed by researchers in the course of particular investigations, having the heuristic purpose of facilitating consideration of certain issues in that investigation. Under this view, all theories are situated social entities, mathematical theories no less than others.

Algebraic semiotics includes some major insights of the founders of semiotics, Charles Saunders Peirce [28] and Ferdinand Saussure [32]. Peirce emphasized (among other things) that the relation between a given token and its object is not just a function (as in denotational semantics), but a relation that depends on the situation in which the token is interpreted, while Saussure emphasized (among other things) that signs come in systems. Semiotic systems include Saussure's insight, while the blending of semiotic systems includes (and extends) Peirce's insight.

Whereas conceptual spaces, which consist of elements and relations among them, are good for studying meaning in natural language, they are not adequate for computational narrative. For example, conceptual spaces and conceptual blending can help us understand metaphors in poems, but more than elements and certain instances of relations among them is needed for an adequate treatment of structure, e.g., to describe how a particular meter combines with a specific rhyme scheme in a fixed poetic form; music raises similar issues, which again require an ability to handle structure. Conceptual spaces are good for talking about concepts about (e.g., how we talk about) things, but are awkward for talking about the structure of things.

Thus, to use blending as a basis for multimedia narrative, we must generalize conceptual spaces to take account of structure, which requires constructors and axioms; it also helps to have a hierarchical type system. Hence we distinguish conceptual blending from **structural blending**,

[3]This use of the word "space" differs from that in the conceptual spaces of cognitive linguistics, which are actually single models, rather than classes of models. Also, recall that for us, a conceptual space is a theory, not a model.

which we may also call **structural integration**, where the former is blending of conceptual spaces and the latter is blending that in general involves non-trivial constructors.

Algebraic semiotics uses algebraic semantics to describe the structure of complex signs, including multimedia signs (e.g., a music video with subtitles), and to study the blending of such structures. A **semiotic system** (also called a **semiotic theory** or **sign system**) [9] consists of a loose algebraic theory, plus a **level ordering** on sorts (having a maximum element called the **top sort**) and a **priority ordering** on the constituents at each level. Loose sorts classify the parts of signs, while data sorts classify the values of attributes of signs (e.g., color and size). **Signs** of a certain sort are represented by terms of that sort, including but not limited to constants. Among the operations in the signature, some are **constructors**, which build new signs from given sign parts as inputs. Levels express the whole-part hierarchy of complex signs, whereas priorities express the relative importance of constructors and their arguments; social issues play an important role in determining these orderings. The **models** of a semiotic theory are just the models of its algebraic theory. Conceptual spaces are the special case where there are no operations except those representing constants and relations, and there is only one sort. Many details omitted here appear in [9, 10, 11].

We now give a simple example of a semiotic theory for a space of structured objects, namely books. Because our purpose is to illustrate the concept of semiotic space rather than to precisely describe books, the theory is greatly simplified. The data theory for books just provides titles and page numbers:

```
dth BOOKDATA is
  pr QID *(sort Id to Title).
  pr NAT *(sort Nat to PageNo).
end
```

Here `QID` and `NAT` are builtin modules for (quoted) identifiers and natural numbers, respectively, and the notation `M *(sort A to B)` calls for renaming a sort `A` in module `M` to become `B` in a new module. Now we define books as lists of `Chapters`, where a `Chapter` is a pair of a `Head` and some `Content`, and a `Head` is a pair of a `Title` and a `PageNo`. It is convenient to use a `subsort` declaration to say that `Chapters` are `Books` (i.e., one chapter books are allowed).

```
th BOOK is sorts Book Chapter Head Content .
  subsort Chapter < Book .
  pr BOOKDATA .
  op <_,_> : Title PageNo -> Head .
  op <_,_> : Head Content -> Chapter .
  op  _ _  : Chapter Book -> Book .
end
```

Here `Book` is the top sort, `Chapter` is the secondary sort, `Head` and `Content` are tertiary sorts, and `Title` and `PageNo` are fourth level sorts. The constructors are the two pairing operations, both of the form `<_,_>`, and the list forming operation, `_ _`; the OBJ languages use the underbar character as a place holder, showing where the arguments of an operation should go. For example, `<'Semantics, 6>` is a valid term of sort `Head`. Among the constituents of `Head`, `Title` has priority over `PageNo`, and among those for `Chapter`, `Head` has priority over `Content` (see [10] for an explanation of this perhaps strange assertion). Content is deliberately left unspecified here; it could be anything in models.

The grammar for narratives in Section 2.1 can also be described as a semiotic system. The top level sort is of course `<Narr>`; the second level sorts are `<Cls>`, `<Eval>`, `<Open>`, and `<Coda>`, while `<Ornt>` and `<Abs>` are third level sorts. It is a good exercise to write out further formal details of this semiotic system, including its priorities and constructors. It should not be thought that this semiotic system will be blended with some conceptual spaces to give a story; this would not be appropriate because narrative structure is at a different level of abstraction from that of narrative content. However, it would be appropriate to blend a narrative structure space with another space that described some other structure, such as the scene/shot/etc. structure of cinema.

Some other examples of semiotic systems are: dates; times; bibliographies (in one or more fixed format); tables of contents (e.g., for books, again in fixed formats); newspapers (e.g., the *New York Times* Arts Section); and a fixed website, such as the CNN homepage (in some particular instance of its gradually evolving format). Note that each of these has a large space of possible instances (i.e., models), but all these instances have the same structure.

Semiotic systems, like the algebraic theories that they build upon, are *formal* in the precise sense that changing the names used in them does not change their space of models, but only the way that parts of the models are named. Thus, these formal descriptions do not even attempt to capture meaning in any real human sense; however, we do try to choose names that can help the reader's intuition.

Using theories for semiotic systems is better than concrete model-based approaches because it is much more natural to treat levels and priorities in the context of theories, and because defining spaces of models with theories makes it convenient to use axioms to constrain the allowable models. For example, in formalizing the representation that produces indices from books, we might want to impose axioms on the target space of indices, e.g., requiring that indexed items must be phrases of 1, 2 or 3 words, and that the page total for indexed phrases must be not more than 2% of a book's page total. Here is what the first of these axioms might look like in BOBJ syntax:

```
    var I : IndexItem .
    eq # I < 4 = true .
```

where the first line declares `I` to be a variable of sort `IndexItem`, where `# I` denotes the length of `I`, and where `<` is the usual less than relation on numbers, given as a `Bool` valued function.

There is a subtle point about how truth values are handled. We have written equations that explicitly give the value `true` for a relation on some constants (and there can also be variables in such equations, so that they apply to a potentially infinite set of constants). If the given equations do not determine a truth value for some arguments, then the relation may be true for those arguments in some models and false in others. However, it is also possible, if that truth value is used in other equations, to set things up so that undefined values are treated as false; this corresponds to what is called the **closed world assumption**. See [13] for some further details.

The reader may have noticed that the structures described by semiotic spaces, like those of conceptual spaces, are static. Fauconnier and Turner do not attempt to capture the behavior of dynamic entities, with changeable state, in their theory. However (given the necessary mathematics), it is not very difficult to extend semiotic spaces to include dynamic structures; in fact, such an extension is needed for applications to user interface design, and is carried out in detail, with examples, in [11]. However, this is not the place to give those details, although these features are necessary for describing interaction in applications such as multimedia games. The conceptual blending theory of Fauconnier and Turner also does not assign types to elements of conceptual spaces; this makes sense, due to the very flexible way that blends treat types, but it also represents a significant loss of information, which in fact can be exploited in some interesting ways, such as being able to characterize some metaphors as "personifications," and being able to generate more striking and unusual blends by identifying sorts that are far apart (see Section 4.2, and also the discussion of type casting in Section 2.8). Another difference from cognitive linguistics is that we do not first construct a minimal image in the blend space, and then "project" it back to the target space, but instead, we build the entire result in the blend space, which seems more perspicuous.

2.6 Semiotic Morphisms

Mappings between structures became increasingly important in twentieth century mathematics and its applications; examples include linear transformations (and their representations as matrices), continuous maps of spaces, differentiable and analytic functions, group homomorphisms, and much more. Mappings between sign systems are only now appearing in semiotics, as uniform representations for signs in a source space by signs in a target space; user interface design is an important application area for such mappings [9]. Since we formalize sign systems as algebraic theories with additional structure, we should formalize semiotic morphisms as theory morphisms; however, these must be partial, because in general, not all of the sorts, constructors, etc. are preserved in the intended applications. For example, the semiotic morphism from the conceptual space for "king" into the blended space for the metaphor "The sun is a king" discussed in Section 2.3 (most likely) blocks the throne, court jester, queen, and castle. In addition to the formal structure of algebraic theories, semiotic morphisms should also (partially) preserve the priorities and levels of the source space. The extent to which a morphism preserves the features of semiotic theories helps to determine its quality, as discussed in detail in [9, 10, 12, 11].

Note that we take the direction of a semiotic morphism to be the direction that models are mapped. Thus, if B is a semiotic system for books and T one for tables of contents, then books (which are models of B) are mapped to their tables of contents, which are models of T. However, this map on models is determined by, and is dual to, the theory inclusion $T \to B$, which reflects the fact that the structure of tables of contents is a substructure of that of books. We can write this in BOBJ as follows:

```
th TOC is sorts Toc Chapter Head .
  subsort Head < Toc .
  pr BOOKDATA .
  op <_,_> : Title PageNo -> Head .
  op  _ _  : Head Toc -> Toc .
end
```

What are called morphisms in algebraic semiotics are called "views" in OBJ. Here is the inclusion view from the table of contents theory into the book theory:

```
view V from TOC to BOOK is
  sort Toc to Book .
end
```

where defaults (as discussed in the next subsection) are used to omit several obvious mappings, and where the sorts in operations in the shared subspace `BOOKDATA` are also preserved by a default convention. In addition, it is required that all data sorts and operations are preserved by morphisms, so that there are really two reasons for `Medium`, `land`, and `water` to be preserved.

2.7 Blending

The simplest form[4] of blend is shown in Figure 1, where I_1 and I_2 are called **input spaces**, and G is called a **base space**[5]. Elements of the base space may be called base sorts, base constants, and base relations. We call I_1, I_2, G together with the morphisms $I_1 \to G$ and $I_2 \to G$ an **input diagram**. Given an input diagram, we use the term **blendoid** for a space B together with morphisms $I_1 \to B$, $I_2 \to B$, and $G \to B$, called **injections**, such that the diagram of Figure 1 **weakly commutes**, in the sense that both compositions $G \to I_1 \to B$ and $G \to I_2 \to B$ are **weakly equal** to the morphism $G \to B$, in the sense that each element in G gets mapped to the same element in B under them, provided that both morphisms are defined on it[6]. In general, all four spaces may be semiotic spaces; the special case where they are all conceptual spaces gives conceptual blends.

Since there are often very many blendoids, some way is needed to distinguish those that are desirable. Criteria for this purpose will be called **optimality principles**, and a **blend** is then defined to be a blendoid that satisfies some given optimality principles to a significant degree. Section 3.2 defines some **structural** optimality principles, which are based only on the structure of blends, rather than their meaning. These include the degrees of commutativity and of type casting (these terms are defined in Section 2.8); there can still be many blends in this sense, as the example in the next section will make very clear. A more precise, but mathematically difficult and narrow, definition is given in Appendix B of [13], and as discussed in Section 2.3, Fauconnier and Turner [8] have given many powerful but vague principles.

$$\begin{tikzcd}
 & B & \\
I_1 \arrow[ur] & & I_2 \arrow[ul] \\
 & G \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Figure 1: Blending Diagram

We refer to a diagram having the shape of Figure 1 as a **diamond diagram**, and we refer to the composition of the two morphisms on its left as its **left morphism**, the composition of the two morphism on its right as its **right morphism**, to the middle upward morphism as its **center morphism**, to the triangle on its left as its **left triangle**, and the triangle on its right as its **right triangle**. We say that a triangle **commutes** for a sort, constant, or relation iff that sort, constant, or relation is mapped to the same sort, constant, or relation in the blend space by the center morphism as by the left (or right) morphism (depending on the triangle), and similarly for relations.

[4] This diagram is "upside down" from that used by Fauconnier and Turner, in that our arrows go up, with the generic G on the bottom, and the blend B on the top; this is consistent with the basic image schema MORE IS UP, as well as with conventions for such diagrams in mathematics. Also, Fauconnier and Turner do not include the map $G \to B$.

[5] The term "generic space" is used in the cognitive linguistics literature [8], but the term "base space" better describes the role of this space in our approach to interface and active multimedia design.

[6] Strict commutativity, usually called just **commutativity**, means that the compositions are strictly equal, i.e., one morphism is defined on an element iff the other is, and then they are equal.

2.8 An Example

This subsection considers blends of the concepts "boat" and "house," using the notation of BOBJ, extending the discussion in [9]. We first give theories for the conceptual spaces involved, noting that the base theory `BASE` has already been given. Here are theories for the two input spaces:

```
th HOUSE is sorts Person Object.
  pr MEDIUM.
  op resident : -> Person.
  op house    : -> Object.
  op live-in  : Person Object -> Bool.
  op on       : Object Medium -> Bool.
  eq live-in(resident, house) = true.
  eq on(house, land) = true.
endth

th BOAT is sorts Object Person .
  pr MEDIUM.
  op boat  : -> Object.
  op passenger : -> Person.
  op ride  : Person Object -> Bool.
  op on    : Object Medium -> Bool.
  eq ride(passenger, boat) = true.
  eq on(boat, water) = true.
endth
```

The first mentioned sort of an OBJ module is called its **principal sort**; it plays a special role in default views, as discussed below. It is convenient to identify this sort with the top sort of a semiotic theory; a similar convention can be enlisted to express the rest of the level ordering, by listing the secondary sorts using a separate `sort` declaration, and then the tertiary sorts with another, etc. There does not seem to be any convenient way to express priorities on constructors in OBJ, but that does not matter for conceptual spaces, where the only constructors are constants.

Here are the morphisms from the base space to the house and boat spaces:

```
view M1 from BASE to HOUSE is
  op person to resident.
  op object to house.
  op use to live-in.
endv

view M2 from BASE to BOAT is
  op person to passenger.
  op object to boat.
  op use to ride.
endv
```

OBJ allows views to be simplified by omitting certain "obvious" mappings, including those of the forms `sort A to A` and `op a to a`. Using this convention, OBJ is able to deduce that `M1` and `M2` both map the sort `Object` in `BASE` to the sort `Object` in their target theories. Similarly for the operation `on`, which is not mentioned in either view. Moreover, everything in `MEDIUM` is automatically preserved, because `MEDIUM` is a shared subtheory. Principal sorts are also preserved unless this is explicitly overridden. The resulting abbreviated views are called **default views** [17]. In fact, the views `M1` and `M2` above could be replaced by totally default views, with empty bodies.

Default views make it tricky for OBJ to represent cases where something has the same name in the source and the target space of a morphism, but we do not want it to be preserved. One solution is to define a subtheory of the source theory that does not include the unpreserved things, and then define the view from that subtheory. Fortunately, this issue does not arise in the present example.

We say that constants c_1 in input space I_1 and c_2 in I_2 are **merged** or **identified** in a blend iff they map to the same constant in the blend space. In such cases, there is an ambiguity about what the merged constant should be called. A convenient convention for the case where c_1 and c_2 have the same sort is simply to merge the two names with a slash between them, as in `resident/passenger`; if the two names are the same, then we simply use the common name. The case where the sorts are different is more difficult and is discussed later, but even when the sorts are the same, one may still wonder which sort name should appear first in the merged name. Our view is that it does not matter, because the formal name is not a *structural* property; however, we try to choose an order that does not conflict with conventional usage. The same considerations apply to the names chosen for the blend space itself, and to merged relations.

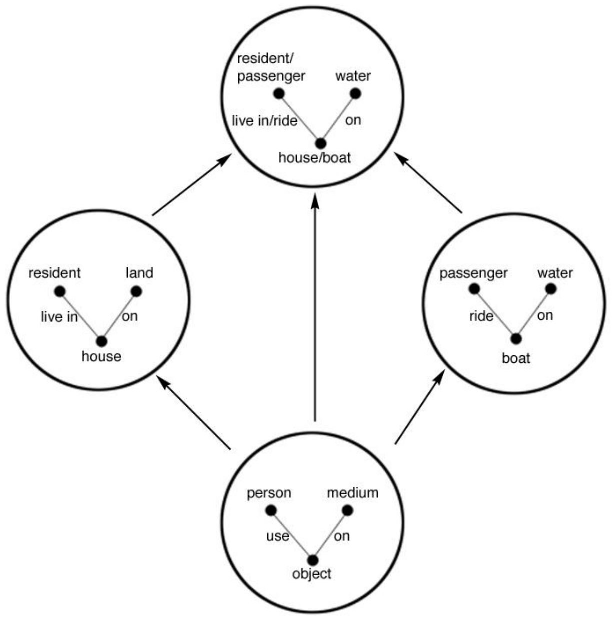

Figure 2: Houseboat Blend Diagram

We now give the theory and three morphisms (three because the center morphism is included) for our first blend of the two given spaces and three given morphisms; see Figure 2. Note that the equation for `on(house, land)` has not been preserved.

```
th HOUSEBOAT is sorts Object Person.
  pr MEDIUM.
  op house/boat : -> Object.
  op resident/passenger  : -> Person.
  op live-in/ride : Person Object -> Bool.
  op on : Object Medium -> Bool.
  eq live-in/ride(resident/passenger, house/boat) = true.
  eq on(house/boat, water) = true.
endth
```

```
view M3 from HOUSE to HOUSEBOAT is
  op resident to resident/passenger.
  op house to house/boat.
  op live-in to live-in/ride.
endv

view M4 from BOAT to HOUSEBOAT is
  op passenger to resident/passenger.
  op boat to house/boat.
  op ride to live-in/ride.
endv

view C from BASE to HOUSEBOAT is
  op person to resident/passenger.
  op object to house/boat.
  op use to live-in/ride.
endv
```

For this blend, the two triangles commute for all three sorts in the base space (and the sort `Medium` is also preserved, because it is a shared data sort); similarly, the two base constants `object` and `person` are preserved. Thus we have commutativity in the strong sense for this blend, so that corresponding elements of the input spaces are identified in the blend; e.g., `house` and `boat` are identified in `HOUSEBOAT`, and the merged element is named `house/boat`. Similarly, the two relations in the base space map to the same relation in the blend via the three paths, so that the relations `live-in` and `ride` are identified. Finally, for each pair of elements in the base space for which a relation holds, the corresponding elements in the blend space satisfy the corresponding relation, which means that all three paths preserve the axiom in the same way.

The reader may have noticed that a significant piece of information is lost in this blend, that `live-in` and `ride` each hold separately for `resident/passenger` and `boat/house`. However, in OBJ, by omitting `use` from the base space, we can avoid identifying the two relations, so that they can hold separately in the blend space, if that is desired.

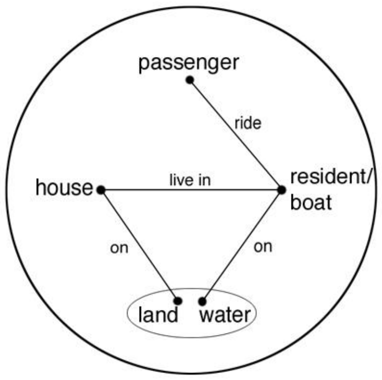

Figure 3: Boathouse Blend Space

Now let's consider a second blend, shown in Figure 3, which in English is called a "boathouse" – in it, the boat ends up in the house. Here are its blend space and three morphisms:

```
th BOATHOUSE is sorts Person Object.
  pr MEDIUM.
  op passenger : -> Person.
  op resident/boat : -> Object.
  op house : -> Object.
  op live-in : Person Object -> Bool.
  op ride : Person Object -> Bool.
  op on : Object Medium -> Bool.
  eq ride(passenger, resident/boat) = true.
  eq live-in(r:Universal>Person(resident/boat), house) = true.
  eq on(house, land) = true.
  eq on(resident/boat, water) = true.
endth

view M3 from HOUSE to BOATHOUSE is
  op resident to r:Universal>Person(resident/boat).
endv

view M4 from BOAT to BOATHOUSE is
  op boat to r:Universal>Person(resident/boat).
endv

view C from BASE to BOATHOUSE is
  op person to resident/boat.
  op object to house.
  op use to live-in.
endv
```

Notice that in the morphism `M3`, mapping `resident` to `boat` would not type check, whereas it does with `r:Universal>Person(boat)`, where `r:Universal>Person(boat)` is a **retract**, an instance of a family of built-in OBJ functions that change the sort of the term in its argument from that the first sort after the "`:`" to become the sort after the "`>`" [13], in this case `boat`, `Universal` and `Person`, respectively, where `Universal` is a built-in OBJ sort such that every other sort is a subsort of it. Without this modification, which is called **type casting** in programming language theory, it would be impossible for the boat to live in the boathouse. This type change is a kind of metaphor, called **personification** in literary theory, in which an object (here, a boat) is considered a person.

For this blend, neither triangle commutes, because the base element `object` is mapped to `boat` in the blend by the right morphism (the composition of morphisms `M2` and `M4`), and to `house` by the left morphism (the composition of `M1` with `M3`), but is not mapped to `boat/house` in the blend. Similarly, since `M4` preserves `passenger`, the central morphism cannot preserve the base element `person`, and the some goes for the base `use` operation. On the other hand, the base relation `on` goes to the same place under all three maps.

There is a third blend which is similar to (in fact, symmetrical with) the above `BOATHOUSE` blend, in which a `house/passenger` ends up riding in the boat. (As noted in [9], there are real examples of this blend, where a boat is used to transport prefabricated houses across some body of water, e.g., for a new housing development on a nearby island.) This blend has similar (in fact, symmetrical) commutativity properties to those of the `BOATHOUSE` blend. It is left as an exercise for the reader to write OBJ code for this blend.

There is a fourth blend, the meaning of which is less familiar than the first three, but the preservation and commutativity properties of which are very good, so that it is a very pure blend of its input, even though its physical existence is doubtful. This is an amphibious RV (recreational vehicle), a vehicle that you can live in, and that you can ride in on land and on water. Here all aspects of `HOUSE` and `BOAT` are realized, with no surprising transpositions, such as a house riding a boat. It is an interesting exercise to write OBJ code for this blend.

In addition, there is a fifth blend, the meaning of which is even less familiar. It is a livable boat for transporting livable boats. Perhaps only a blending algorithm could have discovered it, since it seems rather counter-intuitive for humans.

Finally, a sixth blend gives a boat used on land for a house; it arises omitting the axioms that require a house/boat to be on water and a passenger to ride a house/boat.

It is encouraging that our intuitive sense of the relative purity of these blends, and the degree to which they seem "boat-like" and "house-like," corresponds to principles such as degree of commutativity, and preservation of axioms in input spaces. This suggests that the principles for measuring the quality of blends in [9] are reasonable, and reinforces our hope for good heuristics for finding high quality blends based only on structural properties of the conceptual space network.

2.9 Narrative as Structural Blending

Narrative construction provides a nice example of structural blending. We first define a "narrative space" for the rules in Section 2.1, seen as constructors by reversing their direction and expanding the `*`s; in this context, such constructors are conventionally called "templates." The narrative space also needs additional rules to supply clauses to instantiate the arguments of the Labov constructors. Some arguments would be blended with elements from other spaces to provide particular persons (e.g., a protagonist), places, objects, etc.; this cross-space sharing is indicated by shared generic constants from a generic space. All these spaces would vary from one narrative to another; it is a major task of the artist to choose them appropriately. The Labov structure is not the only possibility here, but it seems likely to be more familiar and comfortable than alternatives.

It is intriguing that the implementation of these ideas closely resembles what a good NLG (natural language generation) programmer would likely do anyway, the main difference being that this deeper understanding of the principles involved supports a clearer system architecture (as shown in Figure 5). In this sense, it is a rational reconstruction of existing practice. However, we also expect that new possibilities will emerge as we gain more experience building active media systems. For example, "optimality" principles for narrative coherence, and more generally for intersentential coherence, need further investigation. Other issues include blending constructors to create new hybrid templates (which in effect would be a novel view of grammar as process), and combining the engines for conceptual and structural blending. Such research problems seem especially relevant for generating new content that seamlessly blends multiple media. Another topic needing more attention is the use of hidden algebra [14] to handle dynamic aspects of interaction; this has been done for user interface design in [11], but narrative raises some new issues.

3 Algorithmic Blending

This section describes our blending algorithm, its implementation, some experiments with it, and then discusses the results. These experiments are not intended to produce comprehensive models of the human mind. Instead, the motivation is to improve the algorithm, the theory, and our intuitions about blending. When results differ from our intuition (as should be expected in the early stages of such a project), it could be due to a bug in the algorithm, in the theory, or in our understanding. Thus, these experiments initiate a highly beneficial cycle of mutual improvements.

A running program is also valuable, because even for relatively simple inputs, the number of blendoids is so large that it is difficult for a human to discover them all. In the houseboat example, the algorithm computes 48 primary blendoids (in which every possible axiom is preserved), and 736 if it also computes those that fail to preserve some axioms. An important conclusion is that efficient techniques for computing high quality blends are necessary for the theory to be useful for content generation and analysis.

3.1 The Algorithm

The blending algorithm is programmed in LISP, and given an input diagram, it either computes one good blend, or else all blendoids (in the sense of Section 2.7 over that diagram). The algorithm uses a graph data structure having four kinds of node, for relations, data sorts, non-data sorts, and constants. For example, the graph for the houseboat example is shown in Figure 4. These graphs have three columns, with relation nodes on the left, sort nodes in the center, and constant nodes on the right. The sort nodes represent sorts after any required non-data sort identifications have been made. An edge from a relation node to a sort node means that the relation is defined on that sort. An edge from a sort node to a constant node indicates that the constant has that sort. An edge from one constant or relation node to another indicates that the connected entities are from different input spaces. This representation makes it clear which axioms could potentially give values to a relation, by following edges from left to right, from relations to constants. It also allows visualization of type casting, as simply changing the sort node to which a constant node connects. Two relations can be identified iff they connect to the same sorts, and to each other.

We can avoid a great deal of computation by identifying relations and constants independently. Hence we can generate the tree of ways to identify relations, and then combine it the tree of ways to identify constants. In the relation identification tree, the branches diverging from nodes arise from whether or not particular elements are identified. The constant identification tree is similar, except when constants have different sorts.

The algorithm is a depth first traversal over the binary trees describing the ways to identify relations and the the ways to identify constants. Different elements from the same input space are never identified. Data sorts and data sort constants are never identified. Non-data sorts are identified iff required by being mapped to from a common sort in the base space. All elements in the input spaces not mapped to from the base space are included in the blend space. When constants of different sorts are identified, both choices for the sort of the blended constant are considered acceptable. The blended constant is then cast to the appropriate sort in each axiom. There is a bijection between the blendoids that can be computed in the two cases, since type casting of the blended constant is necessary in one case iff it is unnecessary in the other. The number of primary blendoids is the number of leaves of the relation subtree multiplied by the number of leaves of the constant subtree. After all possible identifications have been computed, the algorithm concludes by enumerating the power set of axioms. This is necessary due to examples like sixth blend described in Section 3.1 above, although it greatly increases the number of blendoids, which is $2^A P$, where P is the number of primary blendoids, and A is the number of axioms.

Updating axioms after some relations and constants are identified may result in duplicate axioms. For example, under the mappings

`live-in` to `live-in/ride`
`ride` to `live-in/ride`
`house` to `house/boat`
`boat` to `house/boat`

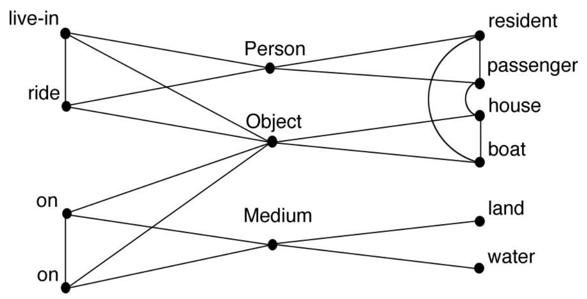

Figure 4: Boat/House Blending Graph

`resident` to `resident/passenger`
`passenger` to `resident/passenger`

both axioms

```
live-in(resident, house) = true
ride (passenger, boat)   = true
```

get updated to

```
live-in/ride(resident/passenger, house/boat) = true
```

In such cases, the duplicate axioms are removed.

3.2 Discussion

Our experiments found that relatively few blendoids correspond to reasonable human concepts, thus motivating a search optimality principles, as discussed in Section 2.7, and carried forward in Section 2.8, with the degrees to which various properties are preserved. The current algorithm uses degree of commutativity as its only optimality principle, but we are considering other optimality principles that are easy to implement, and how to combine them. One potential measure of optimality is the amount of type casting for constants, because the more of these a blendoid has, the more constants get unnatural sorts. Another is axiom preservation. Each can be measured on a numerical scale, as a ratio of actual to possible occurrences. Moreover, such values can be given weights, so that a single number can be computed as a weighted sum, and thresholds can be set, such that blendoids below threshold need not be considered. In the houseboat example, low values would be given to blendoids such as a `passenger/house` that rides a `resident/boat` on water, and to a `passenger/boat` and `resident/boat` that do nothing. The highest value would go to the blend in which a `house/boat` is `ride/live-in` (please excuse the tense here) by a `resident/passenger`. Alternatively, partial ordering relations could be used, as in [9].

In generating a metaphor, one input space is chosen as target, and attributes from the other input space are blocked, which can greatly reduce the search for good blends. Another potentially useful principle is that metaphors do not usually call attention to incongruities among connected entities from the input spaces.

4 Further Case Studies

This section describes an experiment with improvisational poetry, and some plans for active multi-media case studies using the technology described above, especially the blending algorithm. These plans represent feasible directions for exploring the potential of our theory, exposing and correcting its weaknesses, and developing the technology needed for more difficult applications. It is quite possible that these plans will evolve in various ways as our research progresses, and some may even be abandoned.

4.1 Active Poetry

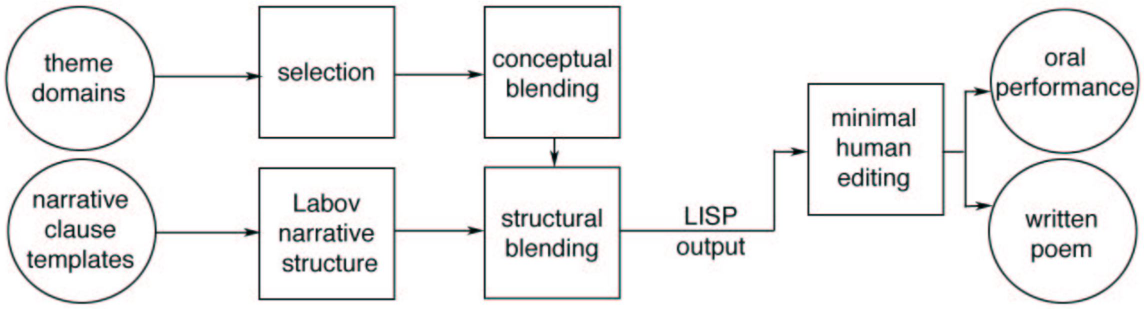

Figure 5: Poetry Generation System

In an initial experiment, Fox Harrell used the blending algorithm in a system entitled "The Girl with Skin of Haints and Seraphs," which generates poems about a girl with skin composed of angels and demons [19]. The LISP program draws on a set of theme domains with concepts such as skin, angels, demons, Europe, and Africa, given as sets of axioms. It constructs input spaces by extracting axioms from two different domains, and then infers relations, sorts, and constants from these axioms. A base space is generated by instantiating shared structure between the spaces. Morphisms from the base space to the input spaces are generated, and the input spaces, base space, and morphisms are passed to the blending algorithm. The generated blends are then placed in poetic phrase templates, which are structured according to the formalization of Labov's narrative discussed in Section 2.1. The system has the following components (see also Figure 5):

1. *Theme Domains:* Its themes are represented by axioms expressing properties specific to that particular poetic system.
2. *Blending Algorithm:* The blending algorithm generates new metaphors from input spaces selected from the theme domains.
3. *Optimality Principles:* Currently only blends with the highest possible commutativity are given as output. Richer and more diverse blends would require different principles from those for conventional blends, which are intended to be easily understood. Any optimality principles used will still reduce blend search.
4. *Labov Narrative Structure:* We used phrase templates based on Labov narrative structure, instantiated with phrases produced by conceptual blending.

A sample poem generated by the system is presented below in two forms, one edited lightly for grammar and formatting, and the other consisting of the original LISP output. Here is the lightly edited output:

her tale began when she was infected with smugnessloveitis.
she began her days looking in the mirror

at her own itchy entitled face.
her failure was ignoring her tormented angel nature.
life was an astounding miracle.
nordic-beauty death-figure vapor steamed from her pores
when she rode her bicycle. that was nothing lovely.
when 21 she was a homely woman. she decided to persevere;
in the rain, she fears only epidermis imperialists.
she believes that evil pride devours and alternates with pride of hope.
it was no laughing matter.
she snuggles in angel skin sheets and sleeps.
inside she was resolved to never find
a smug or paranoid love.

and here is the LISP output on which it was based:

```
((her tale began when she was infected with (smugness / love) -itis)
(she began her days looking in the mirror at her own (itchy / entitlement) face)
(her failure was ignoring her (tormented / angel) nature)
(life was an astounding miracle)
((nordic-beauty / death figure) vapor steamed from her pores when she rode her bicycle)
(that was nothing lovely) (when 21 she was a homely woman)
(she decided to persevere) (in the rain) (she fears only (epidermis / imperialists))
(she believes that (evil / pride devours / alternates-with hope / pride))
(it was no laughing matter) (she snuggles in (angel / skin) sheets and sleeps)
(inside she was resolved to never find an (smugness / paranoia) love))
```

This poem was generated and recited live, on 30 January 2004, at the Powering Up/Powering Down Festival and Conference, organized by the critical arts and technology organization Teknika Radica. It is a commentary on racial politics and the limitations of simplistic binary views of social identity. The dynamic nature of social identity is a central theme of this poetic system, as reflected in the way the program dynamically generates many poems based upon fixed theme domains. We consider that unity between structure and content is an important desideratum for active media design.

4.2 Interactive Poetry

The poem "Walking around" by Pablo Neruda has the form of a narrative. Its first stanza serves as an orientation, introducing the protagonist, the place, and the time (the latter two in a condensed poetic form); the location is perhaps a small city in Chile. Each subsequent stanza explores aspects of some area within that city, using metaphors that are often quite striking. The general theme of the poem is weariness induced by consumerism. Here are its first two stanzas (out of ten), in a fine translation by Robert Bly [27]:

It so happens that I am tired of being a man.
It so happens, going into tailorshops and movies,
I am withered, impervious, like a swan of felt
navigating a water of beginning and ashes.

The smell of barbershops makes me weep aloud.
All I want is a rest from stones or wool,

all I want is to see no establishments or gardens,
no merchandise or goggles or elevators.

We intend to implement an active version of this poem, generated from game like user interaction. The visual interface to the poem could be implemented various ways, ranging from full motion video to a simple textual menu, but a minimal model is appropriate for the initial test, to allow rapid prototyping, testing, and revision. The interface will be a map of the city with icons for various features, such as a tailor shop, barbershop, and nun (each appears in the poem); the user can click on any allowed location, and this will determine narrative structure. Domains for consumerism and for contextual information such as features of the city will be constructed, and metaphors will be generated by blending spaces from the consumerism domain with spaces from the contextual domain, plus guidelines intended to capture aspects of Neruda's style. A natural language generation system will use these blends to create English metaphors and structure the narrative; a history file will prevent excessive repetition and support some cumulative effects. Of course, there is no hope of achieving anything close to the stylistic genius of Neruda; we intend this experiment as inspiration for heuristics that can generate interesting experiences for users. The implementation needs at least the following components in addition to those described in the previous section:

1. *Knowledge Domains:* Neruda draws on a rich background body of imagery, allusion, multi-sensory experiences, objects, and cultural context, which can be (roughly) represented as a set of domains.
2. *Optimality Principles:* Neruda's metaphors often blend concepts in unusual, creative ways (such as "swan of felt" and "water of beginning and ashes"). This will require optimality principles very different from those that produce conventional blends, for example, type casts to very different sorts may be preferred.
3. *Natural Language Generation:* The NLG system produces grammatical English using structural blending of Labov constructors with templates emulating Neruda's style and phrases produced by conceptual blends, again following Neruda's style.
4. *User Interface Technology:* Only modest media processing power is needed, and a simple graphical interface with mouse would suffice, although future versions could include images, real-time video processing, or 3D computer graphics. Audio output, e.g., background noises, would be easy to include, and would fit well with Neruda's multi-sensory metaphors.

This project addresses some shortcomings of the initial active poetry experiment. As in Section 4.1, the thematic content of the poem will be expressed using theme domains. Granularity is increased by using a natural language generation system to reduce the use of precomposed sentences, and different optimality principles will produce metaphors having a very different qualitative feel. Some themes from the first four stanzas of the poem are respectively roughly described by the following axioms, written in a LISP-like notation:

1. `(is-weary-of Person:protagonist  Concept:being-human)`
   `(goes-to Person:human Place:town-location)`
   `(evokes Concept:being-human Emotion:detachment-from-life)`
2. `(evokes Concept:consumerist-life Emotion:despair)`
3. `(is-weary-of Person:protagonist  Concept:being-human)`
4. `(desires Person:protagonist Concept:escape-from-social-convention)`
   `(attacks Person:protagonist Thing:modern-society)`
   `(is-more-important-than Concept:escape-from-social-convention Concept:life)`

Such structures are easily translated into conceptual spaces, since they already have the form of relation instances, and a set of them could also generate a theme domain, such as Consumerism. To blend knowledge domains with theme domains requires selecting appropriate conceptual spaces from these domains. Selecting by priority of sorts and relations gives one approach. Another approach is to select at random and then see how good the resulting blends are, though this will be less efficient.

Knowledge domains provide background context. For example, a `Town-location` is a place such as a tailorshop, movie theater, or barbershop, and would contain `town-objects`, such as goggles, elevators, wool, and stones, where attributes of wool might be heavy and impervious. Knowledge domains could also include entrenched metaphors, many of which occur in the poem. Some that are described in [24] include: DISPASSIONATE IS COLD, LIFE IS A BURDEN, LIFE IS A FLAME, and DEATH IS REST. Some other metaphors in Neruda's poem are PEOPLE ARE PLANTS and its corollary HUMAN DEATH IS PLANT DEATH, e.g., in the funerary symbol of the cut lily. Image schemas are often used in poetry to express themes. For example, in "Walking Around," the LIFE IS A JOURNEY schema underlies metaphors such as "I am withered, impervious, like a swan of felt / navigating a water of beginning and ashes." The experiments of Sections 2.3 and 4.1 suggest further ideas, e.g., implementing personification by casting a constant to be of sort `Person`. Future experiments will no doubt help us discover new principles for generating blends.

4.3 Discussion

Our experiments differ from earlier work on algorithmic generation of stories and poems in some significant ways. Although the major goal of our work is to provide technology for new media genres, it has an empirical basis in cognitive linguistics and is also intended to contribute to that field. In contrast, most previous work has followed the lines of "good old fashioned AI," which assumed that human cognition is computation over logic-based data structures, and which largely ignored (or even denied) the embodied and socially situated nature of being human.

An early relevant work is Raymond Queneau's 1961 *Cent Mille Milliards de Poèmes* (*One Hundred Thousand Billion Poems*) [30], originally published as a set of ten sonnets with interchangeable lines, but later made available in computer implementions. Narrative coherence was not a goal of this work, which exemplifies the experimental literary group Oulipo's often whimsical use of mathematical ideas. James Meehan's 1976 TALE-SPIN [26] is perhaps the first computer story generation system. It produced simple animal fables, with the goal of exploring the creative potential of viewing narrative generation as a planning problem, in which agents select appropriate actions, solve problems within a simple simulated world, and output logs of their actions. William Chamberlain and Thomas Etter's dialogue based program Ractor, and Chamberlain's 1984 book, *The Policeman's Beard is Half Constructed* [31], used syntactic text manipulation to support conversation with users having text input and poetic output. This was not intended as scientific research, but rather as entertainment, with humorous clever output, exploiting the novelty of being "written by a computer program." Charles Hartman's 1996 work in automated poetry generation [20] was presented as literary experimentation, but Hartman realized that it is better not to ask "whether a poet or a computer writes the poem, but what kinds of collaboration might be interesting." Selmer Bringsjord and David Ferruci's 2000 BRUTUS system [4] aimed to explore formalizations for generating stories about betrayal, with the goal of being "interesting" to human readers; they sought Turing test competence, but avoided claims like Chamberlain's, that the system actually authored the texts.

Espen Aarseth's extended analysis [1] of several such systems considers relationships among programmer, system, and reader, as a basis for critical analysis. This focus is useful, because readers' authorial models affect their interpretation of works, causing the approaches of traditional literary criticism to fail. Although there are differences in the theoretical foundations for the templates and rules, the generalizability and soundness of those foundations, and the success of the experience generated, all these systems ultimately amount to some set of precomposed textual templates plus rules for combining and instantiating them, inspired by work like that of Vladimir Propp [29], in the tradition of Russian formalism. In contrast, cognitive linguistics studies the cognitive resources that underlie language, as well as other aspects of thought and social communication; meaning is not considered to reside in the language forms themselves, but to be generated by complex operations involving metaphor, analogy, framing, mental spaces, prototypes, metonymy, conceptual integration (or blending), and more [6]. A science of meaning construction is emerging from this research, which we believe will help us to implement a wide range of narrative models, realized in multiple media, and based on a uniform formalization; we also believe that our experiments can contribute to the refinement of theory in cognitive linguistics.

5 Conclusions and Open Issues

Our experiments on creating active media using a conceptual blending algorithm have yielded insights about further potential applications, and further theoretical research that is needed. One major application area for this work is developing narrative computer games. Many computer games are based in narrative, but despite the fact that users act dynamically within the game worlds, the stories are static. Providing games with the potential to generate story elements on the fly, constrained stylistically and thematically by the game developer's narrative model, can add value to games, in the forms of greater salience for the users' actions, and greater replayability.

Music is another promising area. Hip hop was once countercultural, but is now a commercialized commodity. Considering its use of appropriated media, it seems natural to use it in experiments with the new genre of interactive song. There could be a library of basic rhythms and sampled loops, and a variety of theme domains, e.g., containing stereotypical hip hop themes like cars, money, and guns, which would support an ironic statement about materialistic trends in hip-hop. The problem is then to generate novel rhyming rap on top of the loops, based on metaphors generated from the theme spaces. Novel structure can be produced via structural blending.

Content on websites such as CNN, Google, and Amazon is generated on the fly, based on current events, the state of the web, historical information on the user, etc., but these have little or no narrative structure. Active media techniques like those described in Section 4 suggest ways to structure such information narratively for display. One interesting possibility is the generation of narrative for sporting events. Data from sporting events is often broadcast live. Streaks of point scoring, injuries, lead changes, and similar occurrences can be represented numerically. The same event is currently described differently depending upon who narrates it. It seems plausible to imagine a system that can generate narratives to describe these events from different perspectives, perhaps that of a team, or even a customized perspective based upon the user's preference.

Speculation about possible applications for our research can be exciting, but many important issues remain to be explored, even for the projects described in Section 4. One important problem is finding computationally effective principles for pruning the blend search space. The optimality principles of [8] are suggestive, but not computationally effective; degrees of commutativity, axiom preservation, and type casting (discussed in Sections 2.7, 2.8, and 3.2), are much more practical, but also more limited. Using a real poem, such as Neruda's, provides powerful stimulation for exploring such questions, as well as providing an interesting application.

Some further open problems are better understanding and improving the processes by which input spaces are selected from domains, and by which base spaces are generated; strictly speaking, these problems belong to cognitive linguistics, and so we hope that they will be solved by researchers in that area, though we may have to proceed in a more pragmatic manner in the short term. For media with user participation, such as the Neruda poem system described in Section 4.2, user actions help to determine these choices. Driving metaphor generation with user interaction raises interface design issues quite different from the issues usually explored in cognitive linguistics.

We would like our systems to include more content of greater complexity. Randomness is used in some cases where decisions could potentially be based in part on principles, e.g., selecting input spaces, selecting elements of the blend to output, and selecting narrative clause templates. Also, the granularity of the generated content, as determined by the length of narrative clauses relative to the output, is not very high. This has the effect that the reader could begin to recognize patterns used to generate poetry after reading only a small number of poems. This problem can be resolved by introducing a greater variety of subclauses, and rules for the composition of clauses (e.g., clause A must immediately follow clause B, or clause D may only be included if clause C has). The blends themselves are already more finely grained, and are used to generate words, noun clusters, and short verb phrases that are integrated into the sentences. More directionality could be introduced into blends by blocking some material from one input space. Although the metaphorical quality of a blend is really in the eyes of human interpreters, this strategy should increase the impression of their being metaphorical. Despite such concerns, the program generated interesting and surprising blends and striking phrases having strong thematic coherence. Many models of narrative are used in areas of the humanities such as literary theory, film theory, and cultural studies. The framework developed in this paper is easy to adapt to narrative models other than that of Labov. For example, experimental postmodern narratives could be implemented, as well as traditional poetic forms such as the sonnet.

There are also some more philosophical issues raised by this work. The poetry generation program described in Section 4.1 is suitable as an initial case study, due to the simplicity of its design, but hardcore traditional AI practitioners might consider it cheating because it does not generate content without significant human input. In fact, most text generation projects have been oriented towards total automation and Turing test competence. Our goal was quite different: to demonstrate use of the blending algorithm within a human designed system that generates poetry containing novel metaphors in real-time; just as with computer games, it is desirable and necessary for humans to provide rich content. In any case, we value artistic freedom over dogmatic Turing test reductionism, and desire to explore all the potentialities of our technologies.

The use of results from diverse areas, including formal modeling, socio- and cognitive linguistics, and semiotics, as well as techniques from engineering and the arts, seems necessary in developing conceptual and computational support for the real-time production of new content. The research reported in this paper is an early step towards new ways to address issues such as meaning and narrative, which have traditionally resisted formal representation and implementation, but we believe that significant progress has been made, and that much more will follow, though not necessarily using the terminology or philosophy of this particular paper.

References

[1] Espen J. Aarseth. *Cybertext: Perspectives on Ergodic Literature.* Johns Hopkins, 1997.

[2] Jon Barwise and John Perry. *Situations and Attitudes.* MIT (Bradford), 1983.

[3] Alton L. Becker. Text-building, epistemology, and aesthetics in Javanese shadow theatre. In Alton L. Becker and Aran Yengoyan, editors, *The Imagination of Reality: Essays on Southeast Asian Symbolic Systems*, pages 211–243. Ablex, 1979.

[4] Selmer Bringsjord and David Ferrucci. *Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a storytelling Machine*. Erlbaum, 2000.

[5] Gilles Fauconnier. *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Bradford: MIT, 1985.

[6] Gilles Fauconnier. Methods and generalizations. In T. Janssen and G. Redeker, editors, *Scope and Foundations of Cognitive Linguistics*. Mouton De Gruyter, 2000.

[7] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[8] Gilles Fauconnier and Mark Turner. *The Way We Think*. Basic, 2002.

[9] Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphors, Analogy and Agents*, pages 242–291. Springer, 1999. Lecture Notes in Artificial Intelligence, Volume 1562.

[10] Joseph Goguen. Social and semiotic analyses for theorem prover user interface design. *Formal Aspects of Computing*, 11:272–301, 1999. Special issue on user interfaces for theorem provers.

[11] Joseph Goguen. Semiotic morphisms, representations, and blending for interface design. In *Proceedings, AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15. AMAST Press, 2003. Conference held in Verona, Italy, 25–27 August, 2003.

[12] Joseph Goguen and Fox Harrell. Information visualization and semiotic morphisms. In Grant Malcolm, editor, *Visual Representations and Interpretations*. Elsevier, 2003. Proceedings of a workshop held in Liverpool, UK.

[13] Joseph Goguen and Grant Malcolm. *Algebraic Semantics of Imperative Programs*. MIT, 1996.

[14] Joseph Goguen, Grigore Roşu, and Kai Lin. Conditional circular coinductive rewriting. In Martin Wirsing, Dirk Pattinson, and Rolf Hennicker, editors, *Recent Trends in Algebraic Development Techniques, 16th International Workshop, WADT'02*, pages 216–232. Springer, Lecture Notes in Computer Science, volume 2755, 2003. Selected papers from a workshop held in Frauenchiemsee, Germany, 24–27 October 2002.

[15] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In Raymond Yeh, editor, *Current Trends in Programming Methodology, IV*, pages 80–149. Prentice-Hall, 1978.

[16] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. Abstract data types as initial algebras and the correctness of data representations. In Alan Klinger, editor, *Computer Graphics, Pattern Recognition and Data Structure*, pages 89–93. IEEE, 1975.

[17] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Action*, pages 3–167. Kluwer, 2000. Also Technical Report SRI-CSL-88-9, August 1988, SRI International.

[18] Joseph Grady, Todd Oakley, and Seanna Coulson. Blending and metaphor. In Raymond Gibbs and Gerard Steen, editors, *Metaphor in Cognitive Linguistics*. Benjamins, 1999.

[19] D. Fox Harrell. Algebra of identity: Skin of wind, skin of streams, skin of shadows, skin of vapor. 2004. Performed at *Powering Up/Powering Down*, a Festival of Radical Media Arts, organized by Teknika Radica, 30 January.

[20] Charles O. Hartman. *Virtual Muse: Experiments in Computer Poetry.* Wesleyan, 1996.

[21] William Labov. The transformation of experience in narrative syntax. In *Language in the Inner City*, pages 354–396. University of Pennsylvania, 1972.

[22] George Lakoff. *Women, Fire and Other Dangerous Things: What categories reveal about the mind.* Chicago, 1987.

[23] George Lakoff and Mark Johnson. *Metaphors We Live By.* Chicago, 1980.

[24] George Lakoff and Mark Turner. *More than Cool Reason.* Chicago, 1989.

[25] Charlotte Linde. *Life Stories: the Creation of Coherence.* Oxford, 1993.

[26] James Meehan. TALE-SPIN. In Roger Shank and Christopher Riesbeck, editors, *Inside Computer Understanding: Five Programs Plus Miniatures*, pages 197–226. Erlbaum, 1981.

[27] Pablo Neruda. *Windows that Open Inward (Images of Chile).* White Pine, 1999. With photographs by Milton Rogovin.

[28] Charles Saunders Peirce. *Collected Papers.* Harvard, 1965. In 6 volumes; see especially Volume 2: Elements of Logic.

[29] Vladimir Propp. *Morphology of the Folktale.* University of Texas Press, 1928. Reprinted 1968.

[30] Raymond Queneau. *Cent Mille Milliards de Poèmes.* Gallimard, 1961.

[31] Racter, Thomas Etter, and Joan Hall. *The Policeman's Beard Is Half Constructed.* Warner, 1984.

[32] Ferdinand de Saussure. *Course in General Linguistics.* Duckworth, 1976. Translated by Roy Harris.

[33] Lucy Suchman. *Plans and Situated Actions: The Problem of Human-machine Communication.* Cambridge, 1987.

[34] Francisco Varela, Evan Thompson, and Eleanor Rosch. *The Embodied Mind.* MIT, 1991.

Contents