R Stouffs and R Krishnamurti.
Sortal grammars as a framework for exploring grammar formalisms. *Mathematics and Design'01*, Deakin University, Geelong, Australia July 2001

SORTAL GRAMMARS AS A FRAMEWORK FOR EXPLORING GRAMMAR FORMALISMS

Rudi Stouffs
Faculty of Architecture, Delft University of Technology, Delft, The Netherlands

Ramesh Krishnamurti
School of Architecture, Carnegie Mellon University, Pittsburgh, USA

Abstract: Grammar formalisms come in a large variety and are commonly difficult to implement. To alleviate these obstacles to a more widespread adoption, a framework for developing grammar systems is needed that supports an exploration of alternate and varying grammar formalisms in a rapid prototyping way. We present such a framework based on a formalism for representational flexibility, named ***sorts****. Sorts provide a component-based approach for building grammar systems, utilising a uniform characterisation of grammars.*

Introduction

Grammar formalisms have been around for over 40 years and have found application in a wide variety of disciplines and domains, to name a few, natural language, architectural design, mechanical design, and syntactic pattern recognition. Their implementations, however, have been mostly narrowly focused and sparse. In design, in particular, the expectation of grammar formalisms or similar rule-based systems to pervade design software has so far remained only an illusion. There are three main reasons for this. The first relates to the difficulty stemming from technical considerations of implementing grammars, which we addressed in an earlier paper (Krishnamurti and Stouffs, 1993). The second difficulty pertains to ways of enabling designers to employ grammatical rules in a manner that does not impede their act of designing. In this paper, we consider a third difficulty that affects the rapid development, adaptation, and maintenance of grammar-based systems.

Grammar formalisms come in a large variety, requiring different representations of the objects being generated, and different interpretative mechanisms for this generation. Altering the representation may necessitate a rewrite of the interpretative mechanism, resulting in a redevelopment of the entire system. At the same time, all grammars share certain definitions and characteristics. Grammars are defined over an algebra of objects, U, that is closed under the operations of addition, $+$, and subtraction, $-$, and a set of transformations, F. In other words, if u and v are members of U, so too are $u + f(v)$ and $u - f(v)$ where f is a member of F. In addition, a match relation, $\leq$, on the algebra governs when

an object occurs in another object under some transformation, that is, $f(u) \leq v$ whenever u occurs in v for some member f of F, if u and v are members of U.

Building on these commonalties, we propose a framework for exploring different grammar formalisms, based on a variety of algebra, and match relations (or interpretative mechanisms). We base this framework on *sorts*, a concept for representational flexibility (Stouffs and Krishnamurti, 1998; 1997). Sorts constitute a model for representations that defines formal operations on sorts and recognises formal relationships between sorts. Each sort defines an algebra over its elements; formal compositions of sorts derive their algebraic properties from their component sorts. This algebraic framework makes sorts particularly suited for defining grammar formalisms. Provided a large variety of primitive sorts are defined, sorts can be conceived corresponding to almost every representation. Since sorts can easily be adapted and compared, these provide the basis for exploring alternate and varying grammar formalisms.

In this paper, we describe the concept and implementation of sorts, reflect on a uniform characterisation of grammars and its application to sorts, and present examples of grammar formalisms that can be expressed with sorts. We conclude with a discussion of the scope and flexibility of sorts in exploring new grammar formalisms.

Sorts

Conceptually, a sort specifies a set of similar models; sorts combine algebraically to form new sorts (Stouffs and Krishnamurti, 1998). Consider a sort as a set of models that are described in terms of a set of equations. Then, each individual of the sort corresponds to a distinct assignment of values to the parameters in the set of equations. For example, a point is specified by its tuple of coordinates. Furthermore, sorts can be related by comparing their systems of equations, in a mathematical manner. Since each such equation constrains the values that its parameters may adopt, a sort *subsumes* another sort if its equations form a subsystem of the other's sortal equations.

Representationally, elementary data types define *primitive* sorts, which combine to *composite* sorts under formal compositional operations defined over sorts (Stouffs and Krishnamurti, 1997). For instance, an attribute operator provides for recursively, subordinate compositions of sorts using an object-attribute relationship, in both one-to-one and one-to-many instantiations. For example, the sort of labelled points is specified as a sort of points with one or more labels assigned as attributes to each point in the data form. The operation of sum allows for disjunctively co-ordinate compositions of multiple sorts, under many-to-one and many-to-many instantiations, where each sort may – though not necessarily – be represented in the data form. As an example, a rule has both a *lhs* (left-hand-side) and *rhs* (right-hand-side) component, either of which can be omitted. Other compositional operations can also be considered, such as an array- or grid-like composition of sorts.

The result is a constructive, hierarchical description of sorts as compositions of other sorts, where each leaf node specifies a primitive data type and every other node defines a compositional operation on its operand children nodes (Figure 1). Assigning names to sorts provides for a semantic-like differentiation of sorts that may be, otherwise, syntactically identical, e.g., *lhs* and *rhs* denote equivalent – not identical – sorts.

```
sort conceptrefs : (concepts : [Label]);
sort (hasrefs, isrefs) : [Property] (concepts, conceptrefs);
sort concepttree : concepts ^ concepttree + concepts ^ hasrefs + concepts + conceptrefs ^ isrefs;
```

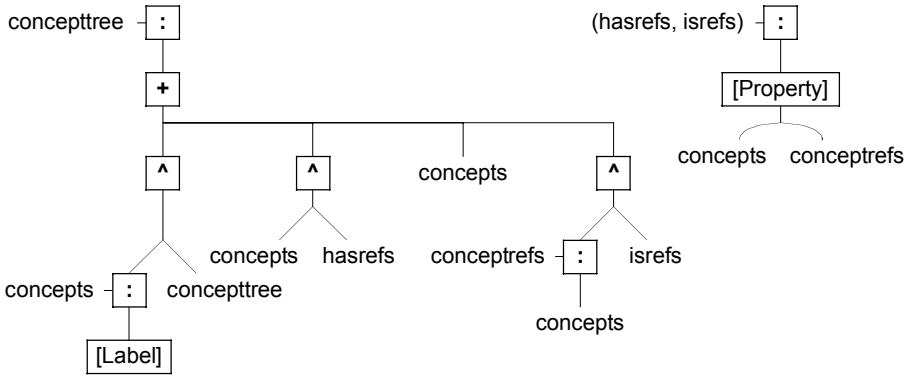

Figure 1. Textual and graphical definition of a recursive 'concepttree' sort. A 'concepttree' may include multiple instances of a single concept, with one instance defined and referenced by all other instances. '+' and '^' denote operations of sum and attribute respectively. ':' denotes the naming of a sort. 'Label' and 'Property' are primitive sorts; the latter defines a property relationship sort between two given sorts.

The definition of a sort includes a specification of the operational behaviour of its members and collections, denoted as *forms*, for the common arithmetic operations of sum, difference, and product (intersection). This behavioural specification enables a uniform handling of forms of different sorts, on the proviso that the universe of all forms of a sort is closed under the respective operations. Additionally, if a match relation exists, that is, there is a partial order relation on the sortal forms, a grammar can be defined over this sort. The simplest behaviour that fulfils these requirements is discrete behaviour, corresponding to a mathematical set, where the part relation reduces to the subset relation and the operations of addition and subtraction correspond to set union and difference, respectively. A contrasting behaviour offers the maximal element representation (Krishnamurti, 1992; Stouffs, 1994), where any element or individual contains indefinitely many, not necessarily disjoint, individuals. This behaviour applies readily to intervals over continuous domains, e.g., line or plane segments, or volumes. Primitive sorts have their behaviours assigned in order to achieve a desired effect, e.g., discrete behaviour for points and labels, interval behaviour for line segments, and an ordinal behaviour for weights such as thickness or tones. On the other hand, a composite sort receives its behaviour from its component sorts, based on its compositional relationship (Stouffs and Krishnamurti, 1997).

Formal relationships between sorts enable the comparison and mapping of sorts as representational structures. The behavioural specification of sorts supports the mapping of information structures onto different sorts, such that the resulting information structures conform to the definition of the respective sorts or representations.

Sortal grammars

Grammars are formal devices for specifying languages. A grammar defines a language as the set of all objects generated by the grammar, where each generation starts with an initial object and uses rules to achieve an object that contains only elements from a terminal vocabulary. A rewriting rule has the form $lhs \rightarrow rhs$; *lhs* specifies the similar object to be recognised, *rhs* specifies the manipulation leading to the resulting object. A rule applies to a particular object if the *lhs* of the rule 'matches' a part of the object under some allowable transformation. Rule application consists of replacing the matching part by the *rhs* of the rule under the same transformation. In other words, when applying a rule $a \rightarrow b$ to an object s under a transformation f such that $f(a) \leq s$, rule application replaces $f(a)$ in s by $f(b)$ and produces the shape $s - f(a) + f(b)$. The set F of valid transformations is dependent on the object type. In the case of geometric entities, the set of valid transformations, commonly, is the set of all Euclidean transformations, which comprise translations, rotations, reflections, and scale. In the case of textual entities, or labels, case transformations of the constituent letters may constitute valid transformations.

The central problem in implementing grammars is the *matching problem*, that of determining the transformation under which the match relation holds for the *lhs*. Clearly, this problem depends on the representation of the elements of the algebra. Sorts offer a representational flexibility where each sort additionally specifies its own match relation as a part of its behaviour. For a given sort, a rule can be specified as a composition of two forms, a *lhs* and a *rhs* (Figure 2). This rule applies to any particular form if the *lhs* of a rule is a part of the form under any applicable transformation *f*, corresponding to the behavioural specification of the form's sort. Rule application results in the subtraction of *f*(*lhs*) from the form, followed by the addition of *f*(*rhs*) to the result. Both operations are also defined as part of the behavioural specification of the sort.

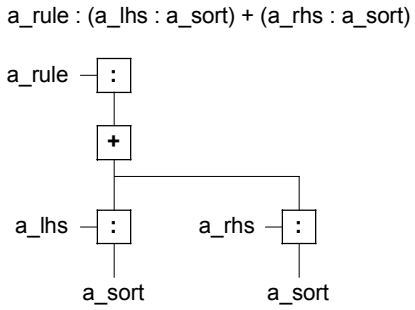

Figure 2. Sort definition for rules applying to a given sort 'a_sort.'
A rule is composed of a lhs and a rhs sort, under the operation of sum.

As composite sorts derive their behaviour from their component sorts, the technical difficulties of implementing the matching problem only apply once for each primitive sort. As the part relationship can be applied to all kinds of data types, recognition algorithms can easily be extended to deal with arbitrary data representations, considering a proper definition of what constitutes a transformation. Correspondingly, primitive sorts can be developed, distributed, and adopted by users without any need for reconfiguring the system. At the same time, the appropriateness of a given grammar formalism for a given problem

can easily be tested, the formalism correspondingly adapted, and existing grammar formalisms can be modified to cater for changing requirements or preferences.

The specification of spatial rules and grammars leads naturally to the generation and exploration of possible designs; spatial elements emerging under a part relation is highly enticing to design search (Mitchell, 1993; Stiny, 1993). However, the concept of search is more fundamental to design than its generational form alone might imply. In fact, any mutation of an object into another, or parts thereof, can constitute an action of search. As such, a rule can be considered to specify a particular compound operation or mutation, that is, a composition of operations and/or transformations that is recognized as a new, single, operation and applied as such. Similarly, the creation of a grammar is merely a tool that allows a structuring of a collection of rules or operations that has proven its applicability to the creation of a certain set (or language) of designs.

Examples

A uniform characterisation for a variety of grammar systems is given in (Gips and Stiny, 1980). Krishnamurti and Stouffs (1993) survey a variety of spatial grammar formalisms from an implementation standpoint. Here, we consider the specification of some of these examples using sorts.

Structure grammars

Structure grammar is an example of a set grammar. "A structure is a symbolic representation of parts and their relationships in a configuration" (Carlson et al, 1991). A *structure* is represented as a set of pairs, each consisting of a symbol, e.g., a spatial icon, and a transformation. The resulting algebra corresponds to the Cartesian product of the respective algebras for the set of symbols and the group of transformations. Both symbols and transformations define sorts with discrete behaviour, i.e., respective sets match under the subset relationship. These combine into a composite sort under the attribute relationship; each symbol in a set may have one or more transformations assigned as an attribute.

Tartan Worlds

The *Tartan Worlds* (Woodbury et al, 1992) is a spatial grammar formalism that bestrides string and set grammars. We consider a simplified string grammar version of the *Tartan Worlds*: each symbol in a string corresponds to a geometrical entity represented as a graphical icon and located on a grid. A rule in these *simplified Tartan Worlds* (Krishnamurti and Stouffs, 1993) consists of one symbol on the *lhs* and symbols on the *rhs* given in their spatial relation. An equivalent sortal grammar may be defined over a sort composed over a grid of a sort of graphical icons. On a fixed-sized grid, the behaviour of the composite sort breaks down into the behaviour of the sort of graphical icons, e.g., ordinal or discrete, over each grid cell. The matching relation is defined in the same way.

Augmented shape grammars

A *shape* (Stiny, 1980) is defined as a finite arrangement of spatial elements from among points, lines, planes, or volumes, of limited but non-zero measure. A shape is a part of

another shape if it is embedded in the other shape as a smaller or equal element; shapes adhere to the maximal element representation (Krishnamurti, 1992; Stouffs, 1994). Shapes of the same dimensionality belong to the same algebra; these define a sort. A shape consisting of more than one type of spatial elements belongs to the algebra given by the Cartesian product of the algebras of its spatial element types. The respective sorts combine under the operation of sum, as a disjunctive composition.

A shape can be augmented by distinguishing spatial elements, e.g., by labelling, weighting, or colouring these elements. Augmented shapes also specify an algebra as a Cartesian product of the respective shape algebra and the algebra of the distinguishing attributes. However, the resulting behaviour can better be expressed with a sort that is a subordinate composition of the respective sorts, i.e., combined under the attribute operator. A sort of labels may adhere to a discrete behaviour, a sort of weights to an ordinal behaviour; a weight matches another weight if it has a smaller or equal value.

Discussion

Technical considerations make grammar systems generally difficult to implement. Part of this difficulty also relates to the appropriateness of a given grammar formalism for a given design problem, and to the representational demands that grammar systems impose on users and developers. Together, these difficulties inhibit the rapid development, adaptation, and maintenance of grammar-based systems. Adopting an existing grammar system may present the user with a system that is not exactly suited for the purpose. On the other hand, it is unreasonable to expect every user to develop a grammar system from scratch or invest the time to analyse and adapt an existing system. Instead, a development environment for grammar systems based on sorts will provide the user with the ability to define a grammar formalism and explore its appropriateness for the problem at hand, and then, to integrate it into a larger application.

The concept of sorts only specifies a common syntax, allowing for different vocabularies and languages to be created, and providing the means to develop conversion and translation facilities between these. For example, a point may be specified with any number of coordinates depending on its dimensionality; its coordinates may constitute integers, reals or rationals; these may be bounded in space, etc. Sorts can be defined accordingly and, based on their compositional structures, compared and related. For example, the operation of sum specifies a subsumption relationship on sorts, where one sort may match a part of another sort, under sum (Stouffs and Krishnamurti, 1997). Compositional structures under the attribute relationship, if not equal, may be fully (or partially) convertible: the attribute relationship is associative though not commutative. Based on the result of this comparison, translation support can be provided for and data loss monitored. For example, partial conversions always result in data loss; complete conversions may result in data loss depending on the behavioural categories of the constituent sorts.

Alternative design representations can be defined as variations on a given sort, by altering the components or the composition. As an example, consider a representation for a collection of drawings given a sort that defines a single drawing. By specifying an attribute composition with a sort of labels, a named collection of drawings is enabled similar to a set of layers in a CAD application (Stouffs and Krishnamurti, 1996). Alternatively, by specifying an attribute composition with a sort of points or rectangles, a layout can be represented for these drawings (Figure 3). One step further, this sort can be modified to

enable drawings to relate to parts within other drawings, allowing for detailing relationships to be specified in this layout.

named_drawings : drawings ^ (labels : [Label])
layouts : drawings ^ (points : [Point])
named_layouts : drawings ^ points ^ labels

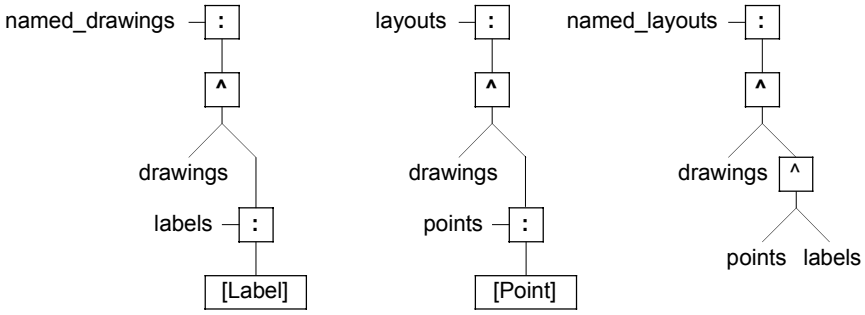

Figure 3. Sort definitions for named drawings, layouts of drawings, and named layouts of drawings, given a sort for a drawing.

There is no imposition of concepts beyond the purely syntactical and the alphabet of building blocks can be readily extended at all times. No language thus created ever needs to be static. Firstly, a vocabulary may be extended from the existing alphabet or using newly developed building blocks. Secondly, representations may be updated by reconfiguring the existing composition of sorts or by extending it using additional component sorts. Far from having to redevelop the data structure and the applicative operations, the concept of sorts aims to provide almost continuous support to evolving representations, providing for an environment that supports exploration and trial, even with respect to the representation. Representational structures can be compared and mapped, data can be readily converted to new and extended (or condensed) representations, and procedural operations remain applicative if such flexibility has been considered.

Grammar formalisms can be explored in a similar way. Depending on the matching characteristics required, a representation or sort could be developed or, preferably, adapted. This representation and its corresponding matching algorithms can be integrated into an exploratory environment. Consequentially the results of this testing, the sort may be further adapted to achieve the desired effect. Such adaptation requires little alteration of the exploratory processes. Forms of different sorts behave in a uniform way with respect to matching and other common operations. This uniform behaviour, together with the flexibility to build a wide variety of representations from a common vocabulary of representational building blocks and compositional operations, makes sorts particularly appropriate for the exploration and rapid development of grammar and other rule-based systems.

Acknowledgment

The research on sorts is funded in part by the Netherlands Organization for Scientific Research (NWO), grant nr. 016.007.007.

References

Carlson, Christopher, R. McKelvey, Robert F. Woodbury, An introduction to structure and structure grammars, *Environment and Planning B: Planning and Design* 18 (1991), 417-426.

Gips, James, George Stiny, Production systems and grammars: a uniform characterization, *Environment and Planning B: Planning and Design* 7 (1980), 399-408.

Krishnamurti, Ramesh, The maximal representation of a shape, *Environment and Planning B: Planning and Design* 19 (1992), 585-603.

Krishnamurti, Ramesh, Rudi Stouffs, Spatial grammars: motivation, comparison and new results, *CAAD Futures '93* (eds. U. Flemming and S. Van Wyk), pp. 57-74, North-Holland, Amsterdam, 1993.

Mitchell, William. J., A computational view of design creativity, *Modeling Creativity and Knowledge-Based Creative Design* (eds. J.S. Gero and M.L. Maher), Lawrence Erlbaum Associates, Hillsdale, N.J, 1993.

Stiny, George, Emergence and continuity in shape grammars, *CAAD Futures 1993* (eds. U. Flemming and S. Van Wyk), pp. 37-54, North-Holland, Amsterdam, 1993.

Stiny, George, Introduction to shape and shape grammars, *Environment and Planning B: Planning and Design* 7 (1980), 343-351.

Stouffs, Rudi, The algebra of shapes, Ph.D. dissertation, Department of Architecture, Carnegie Mellon University, 1994.

Stouffs, Rudi, Ramesh Krishnamurti, An algebraic approach to comparing representations, *Mathematics & Design 98* (ed. J. Barallo), pp. 105-114, The University of the Basque Country, San Sebastian, Spain, 1998.

Stouffs, Rudi, Ramesh Krishnamurti, Sorts: A concept for representational flexibility, *CAAD Futures 1997* (ed. R. Junge), pp. 553-564, Kluwer Academic, Dordrecht, The Netherlands, 1997.

Stouffs, Rudi, Ramesh Krishnamurti, The extensibility and applicability of geometric representations, *3rd Design and Decision Support Systems in Architecture and Urban Planning Conference, Architecture Proceedings*, pp. 436-452, Eindhoven University of Technology, Eindhoven, The Netherlands, 1996.

Woodbury, Robert F., Antony D. Radford, Paul N. Taplin and Simon A. Coppins, Tartan worlds: a generative symbol grammar system, *ACADIA '92* (eds. D. Noble and K. Kensek), 1992.