Towards a Formal Representation for Description Rules

Rudi Stouffs
National University of Singapore / Delft University of Technology
http://www.arch.nus.edu.sg/
stouffs@nus.edu.sg

This paper explores a generalized specification for descriptions and description rules, on the basis of an extensive overview of applications of description grammars in literature. The aim of this research is to establish a formal representation for description rules and the implementation of a grammar interpreter that supports the specification of description grammars, discursive grammars or, in general, shape grammars including textual descriptions.

Keywords: *Description rule, description grammar, shape grammar, representation*

INTRODUCTION

"Designers work with descriptive devices of many kinds. These may be spatial or symbolic" (Stiny 1991, p.171). Descriptions may serve to compare designs to find similarities and descriptions can sometimes be generated. Shape grammars have been used for both: they are a formal rewriting system for producing languages of shapes. When we are describing architecture, we are both interested in the description of the specific architectural object and in its relation to other, similar architectural objects. While shape grammars have been extensively used for this purpose, shape descriptions of architectural objects are lacking, as Stiny (1981, p.257) noted, "main details of the functional elements comprising designs in these languages are provided in the informal, verbal descriptions of the shape [rewriting] rules used." Stiny proposed to augment a shape grammar with a description function in order to construct the intended descriptions of designs. He also illustrated the application of a description function with designs made up of blocks from Froebel's building gifts. Stiny (1981, p.258) indicated that the formal representation of descriptions, together with the descriptions themselves, would "likely have to be worked out on a case-by-case basis". More than thirty years later, quite a few researchers have adopted the idea of a description function or scheme, often specified as a description grammar, in conjunction with a shape grammar, to qualify designs both spatially and descriptionally. As such, we have a number of examples that we can draw generalizations from in order to attempt to establish a formal representation of descriptions that, if not all, will support a significant number of these examples.

An elaborate literature study revealed 116 publications (and reports) referencing Stiny's paper, 37 of which actually include the specification and/or illustration of a description scheme of a textual nature, referring to 16 distinct accounts. Due to space constraints, we present only a subset of these sixteen accounts. Next, we briefly describe these description schemes and their use, present an inventory of the representational components prescribed in these schemes, propose and illustrate a generalized specification for descriptions and description rules, and touch upon its representation.

DESCRIPTION SCHEMES AND THEIR ILLUSTRATIONS

Brown et al. (1996; also, Brown and Cagan 1997) consider a description function that generates process plans for the manufacturing of objects manufacturable by a given process. The objects themselves are generated by a parametric attributed set grammar, but redefining the grammar instead as a shape grammar (with constraint specifications) would not impact the description function as such. Separately, Brown (1997) exemplifies volume calculation as a description function for a grammar specifying a language of stepped grooved shafts.

Agarwal (1999; also, Agarwal et al. 1999) considers a description function that yields cost expressions or equations that can be evaluated to reveal the cost of a design as the design develops through the generation process. This can be used to provide feedback on how design changes affect the cost and thus providing feedback on the generation process; but it can also be used to guide the generation process by cost preferences or constraints.

Li (2001; also, 2004) applies a description function to the specification of a shape grammar for (teaching) the architectural style of the *Yingzao fashi* (Chinese building manual from 1103). The descriptions that are generated are taken from the annotated *Yingzao fashi* (Liang 1983) and, similarly to Stiny's (1981) illustration of description functions, the descriptions reflect on the spatial elements that constitute the design and the way these are combined. Li considers various descriptions (nine in total, specifying measures and descriptions of width, depth, height), as well as drawings (seven, from plan diagram to plan, section and elevation), in parallel.

Duarte (2001; also, 2005a) considers a discursive grammar to incorporate a shape grammar, a description grammar and a set of heuristics, at least from a technical viewpoint. The use of heuristics is intended to constrain the rules that are applicable at each step of the design generation. From an operation viewpoint, a discursive grammar combines a programming grammar generating design briefs based on user and site data and a designing grammar using the design brief(s) to generate designs in a particular style. Both programming grammars and designing grammars utilize description grammars, though only the designing grammar complements the description grammar with a shape grammar. Duarte and colleagues apply discursive grammars, among others, to the Portuguese housing program guidelines and evaluation system (PAHP) and the houses designed by the architect Alvaro Siza at Malagueira (Duarte 2001), to urban design (Beirão 2012) and to housing rehabilitation (Eloy 2102; also, Eloy and Duarte in press [1]). Descriptions, in these applications, take various forms. Duarte (2005b; also, 2001) presents the Malagueira grammar, separately, as a designing grammar only. Here, descriptions represent functional zones and their adjacency relations.

Stiny (2006) presents description rules for Palladian villa plans that count the number of rooms and assign plans to equivalence classes and explores the use of such descriptions to set goals to guide and control the design process. Ahmad (2009; also, Ahmad and Chase 2006) proposes to augment a shape grammar with a style description scheme based on the concept of semantic differential to map the style characteristics of shape rules. Al-kazzaz (2011; also, Al-kazzaz et al. 2010) considers descriptions in shape grammars for hybrid design, where the descriptions provide feedback on rule application based on comparisons between the generated design and the antecedents in the corpus. Additionally he considers a user guide specified as sets of antecedent labels.

Finally, Stouffs and Tunçer (in press) consider a description scheme in the context of the generation of historical architectural typologies, generating an instance of the typology of classical period Ottoman mosques of the architect Sinan from an ontological description thereof. Descriptions come in two forms, as an XML specification and as a set of labels.

REPRESENTATIONAL COMPONENTS OF DESCRIPTIONS AND DESCRIPTION RULES

Few of the publications investigated offer specific, detailed examples that give insight into the construction and manipulation of descriptions and description rules. All avoid the question of implementation with respect to the general application of descriptions and rules, although Duarte and Correia (2006) explain how to implement specific examples. In particular, Duarte and Correia describe the implementation of a description grammar where the description rules are specifically encoded (hard-coded) to handle custom description structures. Duarte et al. (2012, p.84) identified the lack of a (general) description grammar interpreter as one of two reasons for adopting a different strategy considering an ontology to represent urban program formulation rules and an ontology editor as the rule interpreter, the other reason being the complexity of the urban formulation problem.

Below we inventorise the representational components that each of the accounts prescribes. Primarily, these are numbers with operations of sum, difference, etc., strings with the concatenation operation, and lists of any of these components, including lists of lists, with various list operations. Lists can also serve to represent points or vectors, segments, etc.

Stiny (1981) considers descriptions containing multiple sections separated by the '#' symbol. Others consider multiple descriptions handled in parallel (e.g., Li 2001; Duarte 2001). While all schemes consider (a) textual description(s) to be specified in parallel to the shape description(s), Beirão (2012), instead, considers multiple instances of a description, each linked to a particular shape 'object'.

Numbers

Stiny (1981; 2006), Li (2001) and Beirão (2012) all consider descriptions as integers for counting, with operations of addition, subtraction and/or multiplication. Brown (1997), Duarte (2001), Al-kazzaz (2011) and Beirão (2012) consider descriptions as real or floating-point numbers, expressing area, volume and cost, among others, with various mathematical operations, including division and exponentiation. Almost all authors consider numbers, and operations on numbers, as part of more complex descriptions.

Enumerations

Duarte (2001; 2005b), Ahmad (2009), Al-kazzaz (2011), Beirão (2012), Eloy (2012) and Stouffs and Tunçer (in press) all consider enumerations of terms, for example, denoting functions, spaces, qualifications, rule labels, ontological terms, etc. Distinct from strings (as addressed below), enumerated terms are fixed – though a description rule may replace one term by another –, they always form separate entities in a tuple when collected in a description, and they generally do not contain any special characters. While almost all enumerations are grammar-specific, Duarte (2001) proposes an enumeration of 'true' and 'false'.

Lists

Next to integers, Stiny (1981) considers coordinate pairs, tuples (of fixed length) of coordinate pairs (specifying the boundary points of (linear) openings or of spaces or 'rooms'), sequences (of variable length) of coordinate pairs, sequences of tuples of coordinate pairs, and an adjacency matrix, all of which can be represented as lists, of fixed or variable length (Stiny (2006) only considers pairs of integers). Li (2001) considers both tuples and sequences of integers, and triples of textual descriptions (strings). Duarte (2005b) considers a sequence of tuples combining entities of different types: an integer, terms (from an enumeration), and a set of terms (from the same enumeration). Brown et al. (1996), conversely, consider a tuple combining sequences of numbers, a sequence of terms (from an enumeration), and a sequence of pairs of (either) integers or terms, next to a number. Duarte (2001) maps out a large number of parallel descriptions, considering tuples, sequences, and nested variations thereof, of various types, including mixtures of numbers, terms (enumerations and strings) and tuples/sequences. We will denote all nestings of tuples and/or sequences as structured lists, because these can generally be considered as having a predefined tree structure of tuples, sequences, and their entities, where entities within a sequence are commonly of the same type, though entities in a tuple may have different types. Among all description schemes, there are very few exceptions to this rule. Among the schemes here reviewed, Ahmad (2009), Beirão (2012) and Eloy (2012), similarly, consider structured lists of various kinds.

Stiny (1981) considers an append operation on lists, simply using a space to separate the list and the element to be added. Though not explicated as operations, he also considers retrieving the last coordinate pair of a list, determining the number of distinct coordinate pairs in a list, retrieving the distinct number of adjacent coordinate pairs in a list, retrieving loops of coordinate pairs in a list, etc. All these operations, and others, are not specific to lists of coordinate pairs and can easily be generalized to lists of any type, and provided as functions to be applied in description rules. Brown et al. (1996) consider operations to retrieve the first element or elements from a list and to prepend one or more elements to a list, using a shorthand notation borrowed from logic programming in which an initial number of elements of the list, separated by spaces, is followed by a separator '|', and then a parameter for the remainder of the list. Additionally, they consider a function to reverse a list, so as to allow the list to be operated upon from the back as well. Duarte (2001) also uses an append operation on lists but, additionally, considers an addition operation on tuples that have the same structure (i.e., length and entity types). Adding two tuples adds the respective entities: if both entities are numeric they are summed, if both entities are enumerated terms they must be identical, if one entity is a (variable length) list, the other is appended to this list, if both entities are tuples then addition is applied recursively.

Parentheses, angle brackets and square brackets are all used as enclosing brackets to identify lists, even by the same author(s), though parentheses are never used in the case of a list of variable length. Elements within a list are separated with commas or semicolons. Sometimes, enclosing brackets are omitted at the top level of the tree structure and, in a few cases, separation marks as well, leaving only spaces to separate the entities.

Sets

Brown et al. (1996) also consider a set notation, using curly brackets, though their set is in other ways indistinguishable from a variable-length list. Duarte (2005b), Al-kazzaz (2011) and Stouffs and Tunçer (in press), on the other hand, use sets for their ability to identify and remove individual elements from a set without having to be concerned with the size of the set or the ordering of the elements in the set. Eloy (2012) uses a variable-length list notation in the abstract specification of the various, parallel, descriptions, but omits the list and identifies only the individual elements of concern in the specific description rules. A set representation is undoubtedly more appropriate here. Duarte (2001) considers tables as fixed descriptions, containing dimensional and cost information. Here too, each table can be represented as a set, of triples, where each triple specifies the row and column indices and the corresponding cell value.

Strings

Li (2001) and Stouffs and Tunçer (in press) both consider textual descriptions that can be represented as strings with operations of concatenation and replacement. Li considers as a description a triple of strings, each describing a specific aspect, that together form a single statement about the building style. Stouffs and Tunçer consider an ontological description in the XML format represented as a single string that is built up through description rules. Both description schemes necessitate the ability to parametrize parts of the string and rebuild the string from these parts and additional, explicit writing. Li (2001) omits any quotes, any explicit concatenation operator, and distinguishes parameters only through the use of italics. Stouffs and Tunçer (in press) use quotes to identify explicit text from parameter specifications and use the '.' as concatenation operator.

Other schemes also consider entities that can be represented as strings. For example, the enumerations of terms mentioned above. Duarte (2001) considers names of people as an entity type, and Brown et al. (1996) consider labels for tuple entities to improve readability.

Conditionals

Li (2001), Duarte (2001), Beirão (2012) and Eloy (2012) all consider conditional specifications that additionally constrain rule application and cannot simply be captured in an explication of the left-hand side of the rule. For instance, a rule may apply in a number of different cases that correspond to different values for a single description entity. Short of specifying different rules corresponding the different values, which could work in the case of an enumeration but would fail in the case of a real numeric interval, conditional specifications may allow a parameter to be constrained beyond a single value. For example, Duarte (2001), Beirão (2012) and Eloy (2012) present numerous examples where parameters can take a limited set of values. Li (2001) and Eloy (2012) both consider numerical conditions constraining one numeric value in function of another numeric value, or values, all part of the same description. Note that Brown et al. (1996) also consider rule variants that include conditional specifications but these can easily be captured in a further explication of the left-hand side of the rule.

References

Li (2001) and Stiny (2006) consider description rules referencing the current value of other, parallel, descriptions, in the specification of the right-hand-side of the description rule. For example, having one description count the number of rafters, another description describes the disposition of the beams, including the resulting number of rafters. Duarte (2001) similarly considers description rules referencing other parallel descriptions, however, including only a specific entity rather than the entire value of the other description into the specification of the right-hand-side of the description rule. Duarte (2001) sorts this out through conjunctive description rules where the entity in question is identified by a parameter in a description rule that, otherwise, leaves this other description unchanged, such that this parameter can be referenced in the description rule where its value is needed.

Brown et al. (1996), Brown (1997) and Agarwal (1999) consider description rules explicitly referencing shapes and shape rules. Of course, all authors consider description rules to apply in conjunction with shape rules. Such conjunction may imply dependencies. For example, Duarte (2001; 2005b), Beirão (2012) and Eloy (2012) consider shape rules and description rules to use the same labels. Furthermore, they consider conditionals (see above) to apply concurrently to both shape and description rules. The dependency is thus implied in the conditional. In contrast, Stiny's (1981) description rules are specified as independent of the shape rules. Though they collect coordinate pairs specifying boundary points of (linear) openings or of spaces or 'rooms', made up of blocks from Froebel's building gifts, the relative coordinates of subsequent coordinate pairs are hard-coded in the description rules, considering a distance of one between adjacent boundary points. On the other hand, Brown's (1997; and similarly, Brown et al. 1996) description rules for volume calculation require the conjunctive shape rule to provide values for the diameter and length of the section when adding a new section to the shaft, and values for the diameter of the section and the width of the groove, when adding a circumferential groove to a section of the shaft. While Agarwal's (1999) cost equations make explicit reference to characteristics of the shape under rule application, such as its dimensions, these are not evaluated during rule application. However, in order to provide feedback on how design changes affect the cost during the generation process, the cost equations must be able to be evaluated on the corresponding shape at any time.

Others

The inventory above is not complete. In fact, it would likely require a rewrite of the respective schemes, or an in-depth reflection from the scheme's author(s), to identify all variations within the schemes. In this respect, it should be noted that Al-kazzaz (2011) offers no explication of description rules, describing them only conceptually. Lacking even a detailed explanation, we can only guess at how the rules might be explicated. Below, we list other representational components encountered that merit mention, even if briefly.

Both Stiny (1981) and Duarte (2001) define an 'empty' entity, respectively denoted *e* and *nil*. Such entity may denote zero, an empty string, an empty list, or even a tuple of zeros (e.g., zero vector). Li (2001), Duarte (2001; 2005b), Beirão (2012) and Eloy (2012) also adopt the symbol 'Ø' for empty lists.

Li (2001), Duarte (2001; 2005b), Beirão (2012) and Eloy (2012) allow rules to request or necessitate user input. Specifically, Li (2001) and Beirão (2012) identify a series of variables for input by the user, the input for which can be provided beforehand or, if missing, at rule application. In the case of Duarte (2001; 2005b) and Eloy (2012), however, the same rule might be applied more than once, with different input values, therefore necessitating user input at rule application.

Brown et al. (1996) and Duarte (2001) both consider functions as part of description rules that are specific to the grammar under consideration. In the case of Brown et al., these functions are themselves expressed as description rules, though not necessarily operating on the same or similar descriptions. In the case of Duarte, these functions are not explicated, and seem to be an indication of description rules that are too complicated to express otherwise. Without going into detail, we would like to note that Knight (2003) proposes functions encoding algorithms to be embedded in description rules.

A SPECIFICATION FOR DESCRIPTIONS AND DESCRIPTION RULES

Our aim is to establish a formal, generalized representation of descriptions that, if not all, will support a significant number of the variations in descriptions and description rules offered in the schemes presented above. Before we address the actual representation, we will outline the components of a generalized specification for descriptions and description rules. Obviously, descriptions must be allowed to include numbers, strings, lists and sets (of descriptions). Additionally, description rules must be allowed to include various operators and functions as well as variables, acting as parameters, and references to values specified elsewhere (e.g., values of other descriptions, including shape descriptions, and values of variables defined in conjunctive description rules). Within the left-hand-side of a description rule, such references must form part of conditionals on variables. We forego user input values and references within descriptions – as opposed to description rules – here.

Numbers

Numbers can be integers or floating-point numbers; operators on numbers are addition ('+'), subtraction and negation ('–'), multiplication ('*'), division ('/'), modulo ('%') and exponentiation ('^'). The usual operator precedence rules apply and parentheses can be used to override these rules. Numerical expressions can be extended to include other mathematical operators as functions, e.g., square root ('sqrt') and trigonometric functions. Though we do not intend to support logical expressions, observing Duarte (2001), we assume the keywords 'true' and 'false' to represent 1 and 0, respectively.

Strings

Strings must be quoted, using double quotes; the only operator on strings is the concatenation operator ('.'), though it also serves to identify substrings in the matching process. Thus, we adopt Stouffs and Tunçer's (in press) notation for machine readability, though Li's (2001) notation would be preferable from a human reader's point of view. We envision that the more explicit notation can always be parsed and presented in the latter format, if desirable. We reserve unquoted terms, i.e., identifiers, for variables (within description rules) and references. Thus, enumerations are not explicitly considered (except for 'true' and 'false', see above) and, if appropriate, must be used as (quoted) strings.

(Structured) Lists

Lists include specifications of points, vectors and line segments. As mentioned above, most authors show quite a variety in how lists are identified with (or without) enclosing brackets and separation marks. Considering that descriptions must be human-readable, foremost, there is no reason not to support such variety. In order to avoid any ambiguities that may arise, we suggest disambiguation rules to mimic as much as possible how we, humans, might interpret such situations. As an example, a minus sign separating two numerical entities, the first one of which might be represented as a variable, should be interpreted as such, specifying a subtraction, even when it might be possible to consider it instead as a unary negation within a list of (at least two) numbers, with separation marks omitted. Unable to go into more detail within the space of this paper, we must acknowledge that such mimicry might imply some subjectivity.

Operators on lists are append, prepend, and addition. Brown et al. (1996) suggest a notation borrowed from logic programming, uniquely for prepend. Instead, we borrow from regular expressions the ability to collect any number of elements from a list in a variable by adding a postfix '*' or '+' to the variable specification, denoting a list of zero, one or more, respectively, one or more elements. In this way, we do not play up prepend over append. The absence of a separation mark identifies an append or prepend operator. In order to distinguish the entity to be appended or prepended from the list, we intend to rely on the structural similarity of the entity to the entities within the list, in a similar way we humans would do so. Lists of equal length and corresponding entity types can also be added ('+'), following Duarte (2001) (see above).

Additionally, we consider *e* and *nil* as entity placeholders, i.e., denoting an empty entity, whether zero, an empty string or an empty list.

Variables

Variables are specified as identifiers and defined within the left-hand-side of the description rule, where, as a parameter, each is matched onto an entity, list of entities or part of a string. Duarte (2001), instead, suggests variables to be prefixed with a question mark, though he only applies this convention during matching, and may reuse the same name within the same rule with different intention. When reused in the right-hand-side of the description rule, the variable refers to the matched item. For example, the description rule $a \rightarrow a - 1$ applies to a numerical description and subtracts one from the numerical value; the rule "central_dome ".*s* $\rightarrow$ "arch central_dome ".*s* prepends "arch" to a string, but fails if the existing string does not start with "central_dome"; the rule *s1*." arch ".*s2* $\rightarrow$ *s1*." arch ".*s2* leaves the existing string unchanged, but ensures that "arch" is already present within this string.

Conditionals

We propose a shorthand notation for conditional specifications incorporated within the left-hand-side of the description rule, adding a question mark immediately following the definition of the variable, trailed either by a set of admissible values or a conditional expression referring to a previously defined variable (see below for an example).

Descriptions and sets

Most authors consider unique, parallel descriptions. Stiny (1981), instead, considers a single description with multiple sections, each separated by the ' # ' symbol. Beirão (2012), on the other hand, considers multiple instances of a description, each linked to a particular shape 'object'. Others consider sets of descriptions.

For parallel descriptions, we intend to rely on the *sortal* grammars framework (Stouffs 2012), which considers a compositional approach to the representational structures underlying (augmented) shape grammars, allowing for a variety of grammar formalisms, including parallel grammars – at least to some extent –, to be defined and explored. Particularly, the *sortal* grammars framework currently supports parallel grammars on condition that the parallel design descriptions, including spatial ones, do not specify rules to apply under the same transformations. In practice, therefore, *sortal* grammars do not (yet) support parallel shape descriptions, as shape rules commonly apply under a similarity transformation. However, omitting transformations on textual descriptions, they would support parallel textual descriptions, even in conjunction with a shape description. Such is sufficient for our investigation. Note that the association of descriptions to shapes, in support of Beirão (2012), can also be achieved within the *sortal* grammars framework (Stouffs 2012).

It is straightforward to support descriptions with multiple sections, as Stiny (1981) suggested. Any special character that will not be confused for any other purpose, such as the '#' symbol, can be used to separate sections within a description. Variables must be defined to match within a single section, e.g., #*a1*#*a2*# will match a description specifying two components.

We support sets through the ability to specify a set of instances of a single description. That is, sets cannot occur within a unique description, but any description can exist as a set. Similar to a shape rule, a description rule applying to a set of descriptions only needs to specify the subset of those descriptions that are affected by the description rule.

References

The *sortal* grammars framework (Stouffs 2012) currently offers a functional description that allows the formulation of functional expressions to perform calculations over data queries on descriptions, including spatial ones. The calculation is continuously updated upon alterations of the descriptions involved. With some modifications, this can support the description of (cost) equations as exemplified by Agarwal (1999). Data queries are specified by the name of the description and of a predefined property of this description, e.g., the vector position of a point or the length of a line segment. As such, it is not specific to any single data entity, e.g., point or line segment, but with the addition of conditionals, such could be achieved. This reference system could be further extended to provide access to the variables defined within the left-hand-side of a conjunctive rule.

EXAMPLES

Here, we present a limited number of examples illustrating the reformulation of rules and descriptions from the schemes presented above into our proposed generalized specification.

Stiny (1981, p.263) includes a description function g5 affecting four description components:

$a_1 \leftarrow (x, y)$; where (x, y) is the last coordinate pair in the sequence of coordinate pairs a_1

$a_2 \leftarrow p$; where p is the number of distinct coordinate pairs in a_1

$a_3 \leftarrow q$; where q is the number of distinct sets of adjacent coordinate pairs in a_1

$a_4 \leftarrow \lceil rooms \rceil$; where *rooms* is a (possibly empty) list of the rooms defined by a_1

Adopting the rule notation $a \rightarrow b$, and specifying the full description function, we write:

g5: #*a1*#*a2*#*a3*#*a4*#*a5*#*a6*#*a7*#*a8*# $\rightarrow$
#last(*a1*)#length(unique(*a1*))
#length(unique(segments(*a1*)))
#loops(*a1*)#*a5*#*a6*#*a7*#*a8*#

where *a1* through *a8* define eight variables corresponding to the eight sections in the description and last(), length(), unique(), segments() and loops() are predefined functions operating on a list and resulting in, respectively, its last element, its length, a list of its unique elements, a list of pairs such that the i^{th} pair is made up of the i^{th} and $(i+1)^{th}$ elements of the operand list, a list of lists identifying the loops in the operand list.

Li (2001, p.34) includes a description rule B46, with conditional specification, affecting two entities from the triple description of the building style:

B46: For $a_1 + a_2 = v$
a_1 in front, a_2 in back $\rightarrow$ a_1 abutting a_2
$b_3 \rightarrow$ with $c - 1$ columns

This is a grave simplification of the actual rule operation. The value for *v* can be taken from the first entity in the triple; the value for *c* from the third entity in the triple; the second entity contains addition text. Taking into account the necessary expansions, we write:

B46(b): *v*."-rafter building" *b*.", ".*a1*."-rafter beam in front,".*a2*?=(*v* – *a1*)."-rafter beam in back" "with ".*c*." columns" $\rightarrow$
v."-rafter building" *b*.", ".*a1*."-rafter beam abutting ".*a2*."-rafter beam" "with ".(*c* – 1). " columns"

where B46(b) refers to rule B46 applying to description 'b', enclosing brackets and separation marks for the triple are omitted, variables *v*, *b*, *a1*, *a2* and *c* match diverse substrings, a conditional specifies *a2* to be constrained as equal to the value resulting from *v* – *a1*, and 1 is subtracted from the value of *c* in the right-hand-side of the rule.

Duarte (2005b, p.358-360) presents a description rule R9 depicting the case in which the outside zone is dissected into yard and sleeping zones:

R9: <F1; fb, fr, ff, li; o; Z> $\rightarrow$
<F1; fb, fr, ff, li; ya, sl; Z – {ya, sl}>,
ya, sl $\in$ Z = {required zones};

where the label 'F1' indicates the 1st floor stage of the derivation; fb, fr, ff, and the labels 'li' – living room – and 'o' – outside zone – identify the functions associated with adjacent rectangles at the back, right, front, and left side and with the rectangle currently under rule application; the labels 'ya' – yard – and 'sl' – sleeping zone – identify the functions of the resulting rectangles; and Z is the set of required zones. Additional control conditions constrain the correspondingly resulting functions for the 2nd floor stage, though the rule itself only applies to the 1st floor stage.

We suggest to split the description corresponding the first and second floor and, necessarily, consider the set Z as a separate description as well. We write:

R9(F1): *a** <*fb*, *fr*, *ff*, "li", "o"> *b** $\rightarrow$
a <"sl", *fr*, *ff*, "li", "ya"> <*fb*, *fr*, "ya", "li", "sl"> *b*

R9(Z1): {"ya", "sl"} $\rightarrow$ {}

where *a** and *b** determine any number of 'rectangles' that may precede or follow the 'rectangle' under rule application within the total list of 'rectangles' yet defined.

A REPRESENTATIONAL STRUCTURE FOR DESCRIPTIONS

While the descriptions here considered are textual in nature, in order to support the matching of descriptions, a different representational structure may be more appropriate. While description rules may be specified in a textual manner by the author of the rule, and stored as such, within the shape/description grammar interpreter, a tree structure representation will be used for both descriptions and the left-hand-side of description rules, in order to support the matching of both. Tree leaves consist of literals and variables, where literals represent numeric and string entities that serve as constraints in the matching process, and variables specify parameters that can be matched to entities and tree substructures. Additionally, conditions can be imposed on variables to further constrain matching. Non-leaf nodes represent lists and string concatenations. In the case of a list, each entity may correspond to a child node or a single 'variable' leaf node may imply a collection of zero, one or more entities from the list. Similarly, in the case of a string concatenation, the non-leaf node should match a single leaf node, expressing the actual string to be matched.

CONCLUSION

We have presented a generalized specification for descriptions and description rules with the aim to establish a formal representation for description rules to support a large variety of applications of description grammars. For this purpose, we have included an extensive overview of description schemes found in literature and developed our specification on the basis of these schemes. The ultimate aim of this endeavor is to facilitate the exploration, development and implementation of description grammars. Active developments are a formal representation for descriptions and description grammars, and an implementation within the *sortal* grammar framework.

REFERENCES

Agarwal, M 1999, *Supporting Automated Design Generation: Function Based Shape Grammars and Insightful Optimization*, Ph.D. Thesis, Carnegie Mellon University

Agarwal, M, Cagan, J and Constantine, KG 1999, 'Influencing generative design through continuous evaluation: associating costs with the coffeemaker shape grammar', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 13(4), p. 253–275

Ahmad, S 2009, *A framework for strategic style change using goal driven grammar transformations*, Ph.D. Thesis, University of Strathclyde, Glasgow, UK

Ahmad, S and Chase, S 2006 'Grammar Representations to Facilitate Style Innovation: With an Example From Mobile Phone Design', *Communicating Space(s) - Proceedings of the 24th eCAADe Conference*, Volos, Greece, p. 320–323

Al-kazzaz, DAA 2011, *Shape grammars for hybrid component-based design*, Ph.D. Thesis, University of Strathclyde, Glasgow, UK

Al-kazzaz, D, Bridges, A and Chase, S 2010 'Shape Grammars for Innovative Hybrid Typological Design', *Future Cities - Proceedings of the 28th eCAADe Conference*, ETH Zurich, p. 187–195

Beirão, JN 2012, *CItyMaker: Designing Grammars for Urban Design*, Ph.D. Thesis, TU Delft

Brown, K 1997, 'Grammatical design', *IEEE Expert*, 12(2), p. 27–33

Brown, KN and Cagan, J 1997, 'Optimized process planning by generative simulated annealing', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 11, p. 219–235

Brown, KN, McMahon, CA and Sims Williams, JH 1996, 'Describing process plans as the formal semantics of a language of shape', *Artificial Intelligence in Engineering*, 10, p. 153–169

Duarte, JP 2001, *Customizing Mass Housing: A Discursive Grammar for Siza's Malagueira Houses*, Ph.D. Thesis, MIT, Cambridge, MA

Duarte, JP 2005a, 'A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira', *Automation in Construction*, 14, p. 265–275

Duarte, JP 2005b, 'Towards the mass customization of housing: the grammar of Siza's houses at Malagueira', *Environment and Planning B: Planning and Design*, 32, p. 347–380

Duarte, JP, Beirão, JN, Montenegro, N and Gil, J 2012, 'City Induction: A Model for Formulating, Generating, and Evaluating Urban Designs', in Müller Arisona, S, Aschwanden, G, Halatsch, J and Wonka, P (eds) 2012, *Digital Urban Modeling and Simulation*, Springer, Berlin, p. 73–98

Duarte, JP and Correia, R 2006, 'Implementing a description grammar: Generating housing briefs online', *Construction Innovation: Information, Process, Management*, 6(4), p. 203–216

Eloy, S 2012, *A transformation grammar-based methodology for housing rehabilitation: meeting contemporary functional and ICT requirements*, Ph.D. Thesis, Instituto Superior Técnico, TU Lisbon, Lisbon, Portugal

Eloy, S and Duarte, JP in press 'A transformation grammar-based methodology for housing rehabilitation', *Proceedings of the 5th International Conference on Design Computing and Cognition*, College Station, Texas

Knight, T 2003, 'Computing with emergence', *Environment and Planning B: Planning and Design*, 30, p. 125–155

Li, AI 2001, *A shape grammar for teaching the architectural style of the Yingzao fashi*, Ph.D. Thesis, MIT, Cambridge, MA

Li, AI 2004 'Styles, grammars, authors, and users', *Design Computing and Cognition '04*

Liang, S 1983, *Yingzaofashi zhushi [The annotated Yingzaofashi]*, Zhongguo jianzhu gongye, Bejing

Stiny, G 1981, 'A note on the description of designs', *Environment and Planning B: Planning and Design*, 8(3), p. 257–267

Stiny, G 1991, 'The algebras of design', *Research in Engineering Design*, 2(3), p. 171–181

Stiny, G 2006, *Shape: Talking about Seeing and Doing*, MIT, Cambridge, MA

Stouffs, R 2012 'On Shape Grammars, Color Grammars and Sortal Grammars: A sortal grammar interpreter for varying shape grammar formalisms', *Digital Physicality - Proceedings of the 30th eCAADe Conference, Vol. 1*, p. 479–487

Stouffs, R and Tunçer, B in press, 'Typological descriptions as generative guides for historical architecture', *Nexus Network Journal*

[1] http://mason.gmu.edu/~jgero/conferences/dcc12/DCC12DigitalProceedings/Digital%20pdf/Eloy.pdf