Evolved Representation and Computational Creativity

Ashraf Fouad Hafez Ismail

Advances in science and technology have influenced designing activity in architecture throughout its history. Observing the fundamental changes to architectural designing due to the substantial influences of the advent of the computing era, we now witness our design environment gradually changing from conventional pencil and paper to digital multi-media. Although designing is considered to be a unique human activity, there has always been a great dependency on design aid tools. One of the greatest aids to architectural design, amongst the many conventional and widely accepted computational tools, is the computer-aided object modeling and rendering tool, commonly known as a CAD package. But even though conventional modeling tools have provided designers with fast and precise object handling capabilities that were not available in the pencil-and-paper age, they normally show weaknesses and limitations in covering the whole design process.

In any kind of design activity, the design worked on has to be represented in some way. For a human designer, designs are for example represented using models, drawings, or verbal descriptions. If a computer is used for design work, designs are usually represented by groups of pixels (paintbrush programs), lines and shapes (general-purpose CAD programs) or higher-level objects like 'walls' and 'rooms' (purpose-specific CAD programs).

A human designer usually has a large number of representations available, and can use the representation most suitable for what he or she is working on. Humans can also introduce new representations and thereby represent objects that are not part of the world they experience with their sensory organs, for example vector representations of four and five dimensional objects. In design computing on the other hand, the representation or representations used have to be explicitly defined. Many different representations have been suggested, often optimized for specific design domains or design methods, but each individual computational design system has only one or very few different representations available.

Whatever the choice of the representation, it is likely to influence the outcome of the design process. In any representation, some designs may be more difficult to represent than others, and some designs may not be representable at all.

The same applies if the design process is implemented in a computer program. If a design cannot be represented with a given representation, it cannot be the outcome of a design process using this representation. As is the case for human designers, it is also possible that the representation influences a computational design process such that it is easier for the program to find some designs than others. Depending on the design process used, this might make those designs a more likely outcome of the design process. This is for example the case with stochastic optimization processes, like evolutionary systems and simulated annealing. In these cases, the representation is likely to introduce a bias into the design process.

The selection of the representation is therefore of high importance in the development of a computational design system. Obviously, while choosing the representation the programmer has to ensure that all or as many as possible potentially 'interesting' designs can be represented. But it is also generally desirable to minimize the bias introduced by the representation. In contrast to the user-provided design criteria, the bias caused by the representation influences the outcome of the design process in an implicit way which is not obvious to the user, and is difficult to predict and control.

The idea developed in this research is that it is possible to turn the bias caused by the representation into a virtue, by deliberately choosing or modifying the representation to influence the design process in a certain desired way. The resulting 'focusing' of the search process is connected to the idea of 'expansion of search spaces', a notion used in some definitions of computational creativity. Both 'focusing' and 'expansion of search space' will be explored in this research.

Keywords: computational creativity, evolved representation, design processes, evolutionary algorithms, search space.

1 Evolved representations

As described in the Abstract, the choice of a representation will have an influence on the result of a design process using this representation. Any particular representation might make some designs impossible to generate, and some designs less likely to be produced than others. The first of these effects is often used to restrict the size of the search space; the second however is generally avoided, because it is implicit and difficult to predict and control.

In this research, it will be shown that it is possible to create representations that bias a search process in a predictable and controllable way. This means that the representation introduces a user-controllable 'focus' into the design process. Designs inside this focus, those showing user-defined features, have a higher probability to be the result of the design process than designs from other areas of the search space.

1.1 Using the representation to focus design processes

To make use of the influence that representations can have on a design process, any implicit bias introduced by the representation has to be replaced by a bias that is both predictable and controllable by the user. The goal is therefore to find a way to create a representation for an evolutionary system that transforms the search space in a way such that designs are more likely to be generated that shows certain preferred attributes.

This goal can be divided into two parts.

- Identify a method to influence a design process in a predictable way by modifying the representation used in this process. Many different representations are used for design, and variations on these representations influence the design process in different ways. The method should be applicable to different representations in different applications.

- Design a mechanism that allows the creation of such a representation for particular design problems. Ideally, this mechanism would require little user-interaction.

The following section presents an intuitive view of the algorithm that has been developed, and the sections following describe how this algorithm achieves the two parts of the goal described here.

1.2 Basic implementation schema

The general schema of the algorithm is shown in Figure 1. The first step is the same as in any design computing application: the definition of a representation. In this case however, this is only an initial representation, referred to as the 'basic representation'. It is designed to allow a very large search space, including as many potentially interesting designs as possible, and is therefore usually very basic and low-level. In the example, the basic representation is based on squares that can be connected to form larger shapes (Figure 1(a)).

In the second step, the system is put into a training situation, where a search algorithm using the initial representation is set to solve a simple design task: to produce phenotypes that are (partial) copies of a set of examples given to the system. During this phase, a meta-level process observes how the basic representation is used. It identifies patterns in the genotypes of the individuals that are particularly successful, and modifies the representation used by the system by adding symbols for these patterns. The result is a new, 'complex' or 'evolved' representation, biased in favor of common features in the designs produced in the training session. In Figure 1, L-shaped shapes appear in the design examples (Figure 1(b)); therefore the representation is expanded by adding a symbol for this shape (Figure 1(c)).

The designs produced in this step that are copies of the examples are discarded. However, the evolved representation provides the required focus, centered on the examples. A regular search algorithm, using this evolved representation, can then be used to produce new designs that are likely to be similar to the examples. The effect of the evolved representation depends on whether the basic representation is replaced by the evolved representation or whether the evolved representation is added to the basic representation. In this example, the L-shape is only added, and the new designs can use both the original square and the L-shape (Figure 1(d)).

This basic algorithm solves the following two parts of the goal.

- The representation is manipulated by adding new symbols to the alphabet used in the representation. These additional symbols represent certain features, and by introducing them into the representation designs containing these features are favored; in other words the additional symbols create a focus in the search space.
- The additional symbols can be identified automatically, using machine learning. No user interaction is required, only the provision of a set of example designs.

Both points will be elaborated in the following sections. A system as described here also allows for the transformation of the focus thus created, simply by modification of the representation. This feature is important in connection with creative design.

1.2.1 Use of evolutionary algorithms

The creation of an evolved representation from a training situation requires two main features: the ability to produce copies or partial copies of the examples without additional knowledge or supervision, and a flexible and easily modifiable representation. At the same time, the evolved representation is intended to be used to produce new designs. Therefore, it has to be compatible with the search method that will be used to produce these new designs.

Evolutionary algorithms seem to fit these requirements very well. They require only a fitness value that can easily be

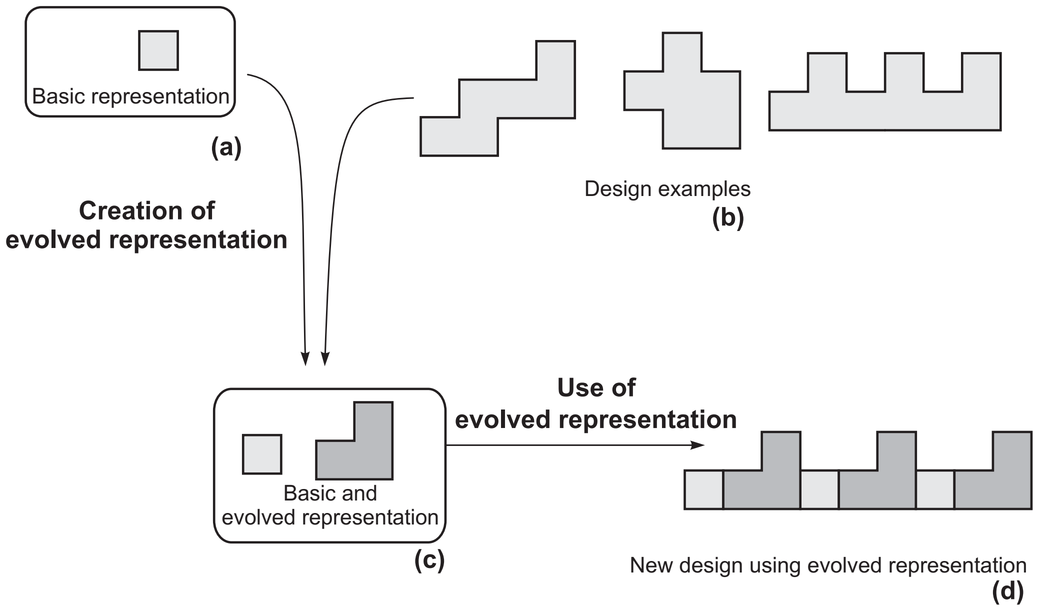
Basic representation
(a)
Creation of evolved representation
Design examples
(b)
Use of evolved representation
Basic and evolved representation
(c)
New design using evolved representation
(d)

Fig. 1: Basic concept of use of evolved representations in creative design

calculated from a comparison between phenotypes and the design. As will be shown in the following sections, they also allow for the manipulation of the representation during the search. Finally, as the large body of existing design systems using evolutionary algorithms shows, they are also very well suited to the generation of new designs. This means that the same type of algorithm can be used for the creation and for the use of the evolved representation, and the compatibility of the representation is therefore ensured.

1.3 Influencing search space using evolved representations

In evolutionary algorithms, a bias towards particular designs can be introduced either in the genotype representation, or in the genotype-phenotype transformation. For example, using evolutionary algorithms with variable-length genotypes, individuals with short genotypes are generally easier to find than individuals with long genotypes. Similarly, if the genotype-phenotype transformation were such that particular phenotypes can be represented by many genotypes, these phenotypes would be expected to be easier to find than phenotypes that are represented by only one genotype.

In the method presented here, the biggest influence on the search process comes from the first effect: designs with certain desired features are represented with shorter genotypes, and are therefore easier to find. However, the method also introduces new ways to represent these designs, which again improves the chances of these designs in the design process.

To illustrate the creation of a focus, an evolutionary system with a string representation is used. In such a system, the genotypes are strings of fixed or variable length, constructed from symbols of a predefined alphabet. To create a focus in the search space of such an evolutionary algorithm, the representation used in this algorithm is modified by the introduction of additional symbols to the original alphabet. To distinguish the introduced symbols, they will be referred to as 'evolved genes', while the original symbols will be called 'basic genes'.

Evolved genes can be used together with the basic genes to produce new genotypes. As a result, two different kinds of genotypes can be distinguished: genotypes that use only basic genes (referred to as 'basic-level genotypes'), and genotypes that also use evolved genes (referred to as 'evolved-level genotypes'). This introduces another representation level, as the genotype representation is now split into a 'basic genotype representation' (or 'basic representation') and an 'evolved genotype representation' (or 'evolved representation'), as shown in Figure 2.

The evolved genes are defined such that each evolved gene represents a certain combination of basic genes. As Figure 2 shows, evolved-level genotypes can therefore be transformed into basic-level genotypes by replacing each evolved gene with the set of basic genes it represents. For example, if gene *A* in the figure appears in an evolved level genotype, this indicates that in the corresponding basic level genotype, this and the next position is filled by basic gene 0; evolved gene *C* indicates a sequence of four basic genes 2. The original genotype-phenotype transformation can then be used to generate a phenotype. In the general case, the evolved-level genotype to basic-level genotype transformation is an n-to-one transformation; there will be many different ways to represent the same basic level genotype using evolved genes.

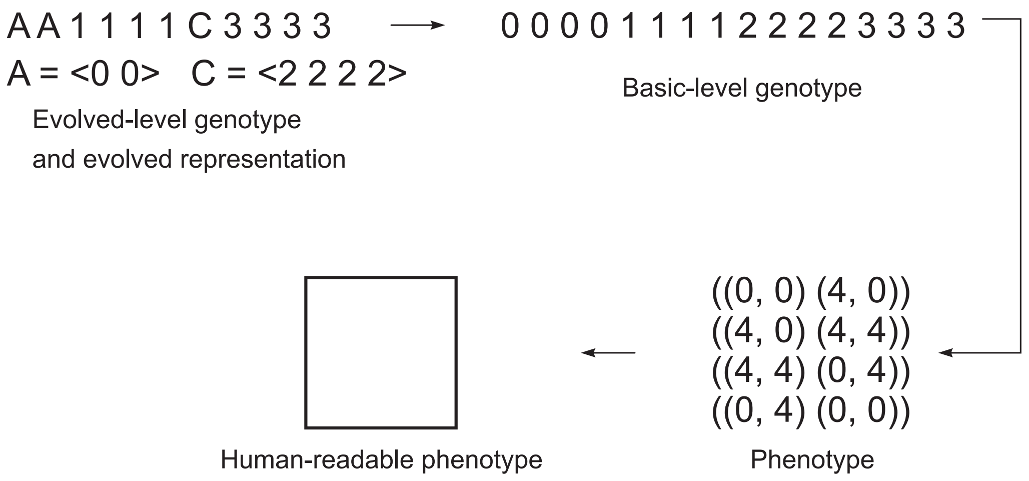

Fig. 2: Additional layer of representation caused by the use of evolved genes

1.3.1 Transformation of the search space

Since evolved genes are represented by single symbols in the genotype, they are 'atoms' for evolutionary operations. At the same time, they represent a combination of basic genes, effectively encapsulating this gene combination. As a result, this gene combination cannot be 'broken up' by any genetic operation. It can still be removed as a whole, but its chances of surviving a genetic operation are much higher than for all other gene combinations that occupy the same positions on the basic-level genotype, but are not encapsulated into an evolved gene. The more basic genes an evolved gene encapsulates, the stronger is this effect. Similarly, if evolved genes are used in the creation of an initial population, then the gene combinations represented in the evolved genes will have a higher chance of being represented in an individual than any other, random combinations of basic genes.

The effect of the introduction of evolved genes is, therefore, that certain combinations of basic genes will be advantaged in the genetic search. It follows that evolved genes can be used to bias the search of the evolutionary system in favor of this feature if combinations of basic genes can be identified such that the probability that a certain feature is present in the phenotype is higher if the gene combination is present in the genotype than if it is not present.

The introduction of evolved genes can be seen as a transformation of the search space, as illustrated in Figure 3. The example assumes a variable-length representation where each basic gene in the genotype is directly translated into a movement of a pen in a certain direction. The original alphabet therefore has four members, shown in the figure as *a, b, c* and *d*. The genotype-phenotype transformation transforms each letter into the movement of a pen, for example each occurrence of the symbol *a* in a genotype results in an upward movement of the pen of one unit length. The genotype *dacb* describes a simple square, constructed by the movement of the pen one unit to the left, one unit upward, one unit to the right, and one unit down. The genotype *dbca* represents the same square, however the pen ends up at a different corner or the square. In the figure, the endpoint of the movement is indicated with an arrow.

The search space can be illustrated by a number of concentric circles, each defining the space of designs that can be defined by a genotype of a certain length. The inner circle contains the designs represented by genotypes of length one, in other words the basic genes translated into phenotypes. The further away a design (or part of a design) is from the center, the larger is the genotype required to represent it, and also the larger the space that has to be searched to arrive at this design.

The original search space is illustrated in Figure 3(a), with the four basic genes in the center. The second circle shows all designs that can be derived from genotypes of length two (i.e., using two vectors). The other circles give some examples of designs using genotypes of length three, four and five.

Every time an evolved gene is created, the structure of the search space is changed. The state of the new gene in the search space is moved into the center, all design states in the next circle that can be derived from that state are moved into the second circle, and so on. For example, if an evolved gene is introduced for each of the combination of four consecutive basic genes that represent the two closed shapes in the fourth circle, the search space changes as shown in Figure 3(b). The squares are now represented directly by an evolved gene, and the shapes on the fifth circle that are derived from the squares can now be found in the second circle. The greater the number of evolved genes a design state involves, the more it is moved towards the center. For example, a shape with the four squares that is now on the fifth circle (that is, can be constructed from genotypes of length five) would have been on the fourteenth circle before (fourteen vectors, because the shape cannot be drawn without drawing two lines twice).

Since the introduction of a new gene increases the size of the alphabet of the representation, more genotypes of a given evolved-level genotype length exist, and the size of the search space for a given length increases. This is illustrated by using

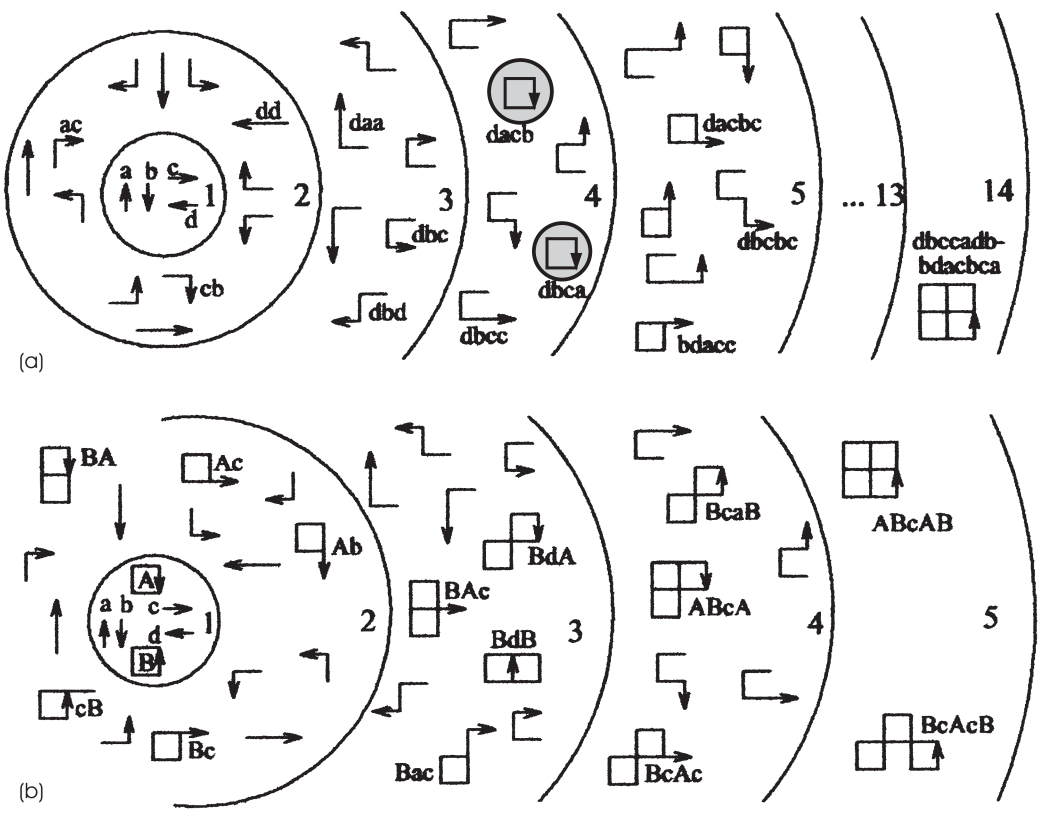

Fig. 3: Example of an evolved representation: (a) original representation; (b) representation with evolved genes. Some of the corresponding genotypes are given, capital letters denote evolved genes. The transformation from phenotype to genotype is not always unique, e.g., the genotypes 'ABc' and 'BAc' produce the same phenotype. Arc segments indicate that only part of the space is shown.

larger circles in Figure 3(b). However, the reduction in genotype length has a much stronger, search space reducing, effect, as can be shown for the foursquare shape. To produce this shape using basic genes only, the search space consists of basic-level genotypes of length fourteen, with four basic genes, containing $4^{14} = 268\,434\,456$ elements. To produce the same designs using basic and evolved genes, the search space would consist of all genotypes of length five with 6 symbols in the alphabet, containing $6^{5} = 7\,776$ elements.

1.4 Creating evolved representations

The previous section showed how evolved genes can be used to influence an evolutionary search in such a way that certain features are favored. The second task is to find a way to identify the appropriate combinations of basic genes, so that the evolved representation can be created.

Creating an appropriate evolved representation is straightforward in the case where the features that are intended to be included are explicitly known, and the genotype-phenotype representation is such that it is possible to directly map those features onto gene combinations. However, neither of those conditions is usually fulfilled. Explicitly enumerating all desired features requires a high amount of user input, and the genotype-phenotype translation can be such that a reverse translation is difficult or impossible. It is therefore necessary to find a different method to create the evolved representation.

Machine learning can provide such a method. Figure 4 shows a schematic outline of a system-employing machine learning to create the evolved genes. The central element is a user provided example. The features present in this example will provide the center of the focus created by the evolved representation. The only user input required is the provision of this example.

The loop on the left of Figure 4 is based on a conventional evolutionary system. Individuals are taken from the population (which is initially generated randomly), offspring are produced, fatnesses calculated and the new individuals are either discarded or introduced into the population. The fitness function in this system is a comparison between the phenotypes produced and the example. This comparison returns a high value for phenotypes that are similar to the example, and lower values for phenotypes that are less similar; the exact implementation depends on the application domain. This fitness will be referred to as ‘similarity fitness’ ρ. At the start, the individuals produced will have hardly any similarities to the example; at the end the system might have found an identical copy of the example. In between, the system produces a high number of individuals that are in some features similar to the example. The goal of this system is not to produce the final individual, but to generate a range of individuals that contain a large variety of features from the example in a variety of combinations. Some additional control may therefore be necessary to prevent convergence of the population; this control usually influences the fitness and the way new individuals are inserted into the population.

The population generated by the evolutionary system is then used as a pool of samples to create the evolved re-

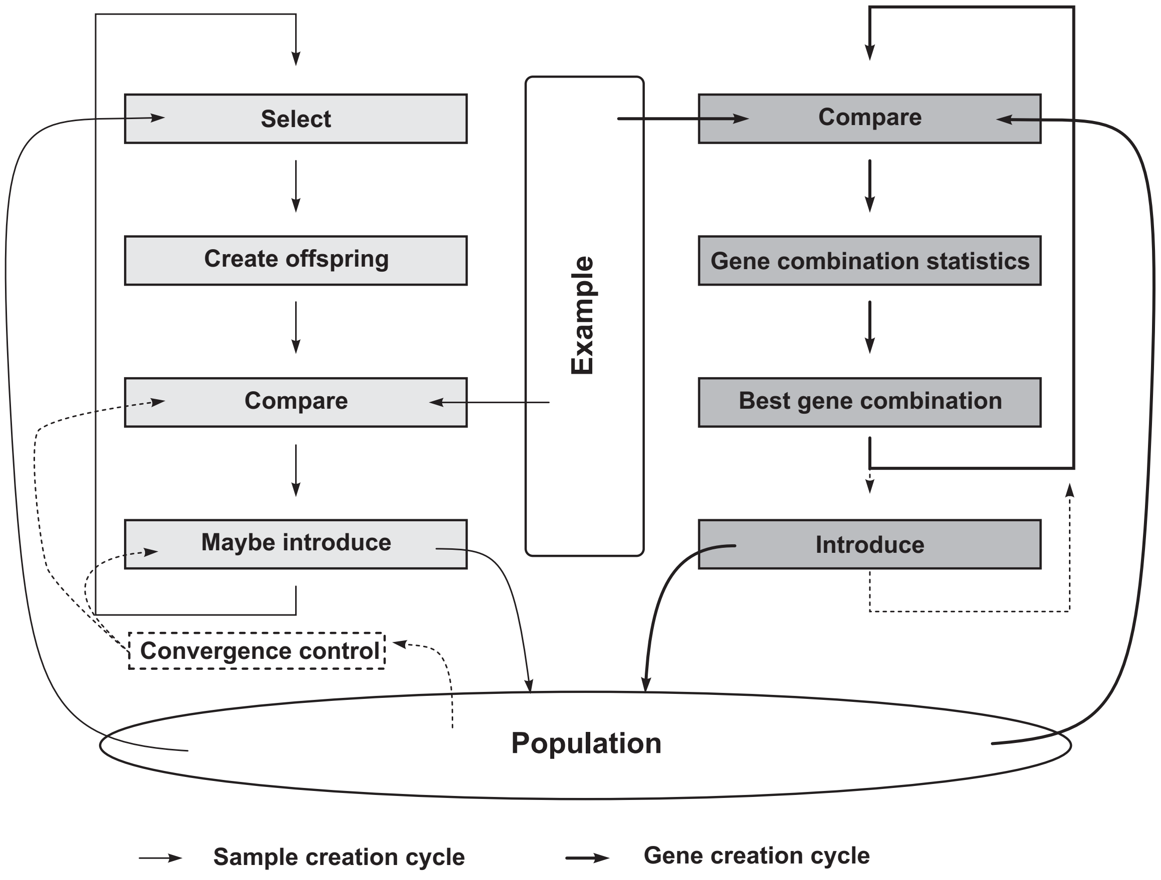

Fig. 4: Schematic representation of the proposed system to create evolved genes

presentation. This is done in the right loop in Figure 4. Gene combinations that appear predominantly in sample individuals that are very similar to the example can be used to create the set of evolved genes. The assumption behind this is that the genotype-phenotype transformation is defined in such a way that features in the phenotype are correlated to subsets of genes on the genotype. This does not have to be a direct mapping, it is sufficient when:

- The probability that the feature exists in the phenotype is higher if the gene combination can be found in the genotype than if not.
- The probability that the gene combination can be found in the genotype is higher if the feature exists in the phenotype than if not.

A result of the probabilistic nature of the evolutionary process is that the evolved representation created for an example is not unique. Instead, different runs will produce different evolved representations, each creating a focus around the example, but each slightly different from the others.

To reduce the computational cost in identifying the best new gene combination, it is possible to take only combinations of two existing genes into account. These existing genes can be either basic genes or evolved genes; any new evolved gene can therefore be composed of two basic genes, two evolved genes, or a basic gene and1 an evolved gene.

In cases where the genotype-phenotype transformation allows gene combinations to be converted directly into features, this construction of high order genotypes can be inter-prated as a creation of large 'building blocks' by combining smaller ones. Figure 5 shows how a building block with seven elements can be assembled in four steps, from both basic blocks and lower-order building blocks. The building block that is added in the third step would in turn have been created from basic blocks in a similar process.

Creating complex evolved genes by combining simpler evolved genes can be described as a 'bottom-up' process, in the sense of artificial life research, where complex behaviors and structures are the result of interaction of a number of simpler behaviors or structures.

1.4.1 Feedback into the evolutionary system

It is possible to run the evolutionary system until a sufficient number of samples are generated, and then run the gene extraction to create all evolved genes. However, given the bottom-up construction of complex evolved genes, a different approach offers itself: phases of sample creation and gene extractions can be interwoven. The evolved genes created in the gene extraction can be added to the representation and introduced into the population. In the early stages, where the individuals produced contain only little knowledge about the examples, simple evolved genes are produced and introduced; in later stages, more complex evolved genes will be introduced. The evolved genes thereby continually improve the representation used in the evolutionary process, helping it to produce larger and better fitting individuals.

If this strategy is adopted, it is especially important to calculate the similarity fitness ρ in a way that ensures that a gene combination is in fact related to a feature, because the first gene extractions will occur in early phases of the run of the evolutionary system, where the individuals still differ strongly from the examples. Any gene combination, even if it occurs in high-fitness individuals, could otherwise simply reflect random influence from the initial population, instead of features in the example.

1.5 Creating new designs using evolved representations

When a set of evolved genes has been created, it can be used to provide a focus for the generation of new designs. Evolved genes are used to create a new representation, and a new random initial population is created using this representation. A conventional evolutionary system can then be run to produce new designs, using a fitness function that represents user-defined design criteria. Depending on how the evolved genes are used, their effect on the search space can be different. If the evolved genes are added to the basic representation, the system can still use basic genes at any place in the genotype if the fitness requires it. Therefore, the

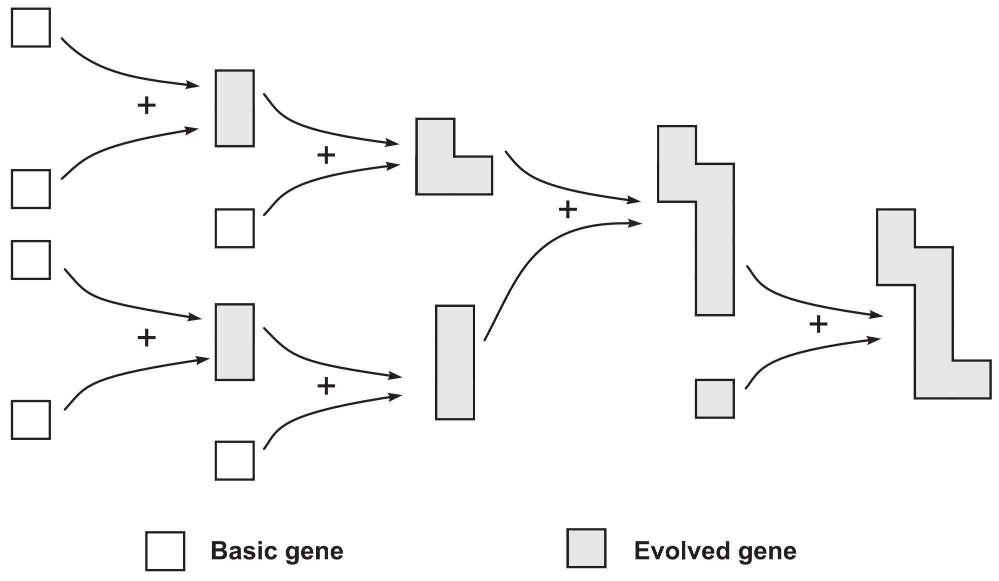

Fig. 5: Interpretation of the creation of high-order evolved genes as the combination of building blocks

set of genotypes in the basic-level genotype search space and the resulting set of designs in the phenotype space are not changed, only the probability that some designs are found. If, on the other hand, the evolved genes *replace* the basic genes, only some basic-level genotypes can be produced, and therefore the set of designs in the phenotype space is restricted to a subset of all phenotypes possible with the basic representation. This will be referred to as 'hard focus'.

In most applications, the 'soft focus' approach is more appropriate, since it still allows the adaptation of the design to any specific design criteria. In these cases, two 'forces' influence the outcome of the design: the influence of evolved genes on the initial population and on the genetic operations, and the selection for or against phenotypes containing certain features. Kauffman [10] shows that on general rugged, multi-peaked fitness landscapes, often-large bands of near constant fitness exist, where selection therefore has no influence on the population. Released from random points in the fitness landscape, the population usually ends up in those bands. Inside a band, other forces, usually weaker than selection, can influence the population. Local optima inside regions favored by such a force are then more likely to be reached than other local optima in other regions. Rugged fitness landscapes usually result when the influence of a gene in the genotype onto the fitness of the phenotype depends on a number of other genes in the genotype, a condition that certainly holds for most situations where evolutionary algorithms are used in design. The evolved representations can then be seen as a force that controls the 'neutral drift' of the population towards the focus in the search space. The evolved genes will only introduce those features that are positive or neutral with respect to the user provided fitness.

If no basic genes are used in the evolved representation, the evolutionary design system has no choice but to use the evolved genes. The choice of evolved genes however is again dependent on how the use of the specific evolved genes interacts with the fitness function.

2 Computational creativity

When can a computational process be called 'creative'? It seems there are two ways to give a claim of creativity some foundation. One is to derive the process directly from observations of particular processes of human creative design, for example the use of analogies and emergence. Both of these processes are assumed to play a role in human creative behavior, and a number of computational processes have subsequently been developed that use analogies to them ([20], [2], [22], [21]) and emergence ([7], [16], [18], [8]) create or facilitate creative design.

The second way is to try to define a general characterization in computational terms of human creative design activity, and to use this to guide the development of a computational process. This 'top-down approach' allows the use of computational techniques and methods that are not related to any specific human cognitive behavior, as long as they correspond to the general characterization. It might, for example, allow the use of evolutionary algorithms and neural networks, which most likely do not play any role in human creativity. The definitions involve concepts related to transformation or expansion of search spaces (for computers) and conceptual spaces (for humans). Neither of the definitions, however, says anything about how such a behavior can be achieved. Where do new rules come from? How can the conceptual space be transformed? Or, in computational terms, where do the new variables come from?

This research will look at search spaces in the context of a finite system, and how a computational process can expand a search space.

2.1 Finite systems and closed worlds

The fact that a process is running as a program inside a computer introduces a set of theoretical limitations. The two most important in the context of design computing are:

1. The size of memory available to the program is limited, which means that the total number of different states that the program can assume is limited (finite system).
2. The computing power available to run the program is limited, which means that the number of different states a program can assume in any specific time span is limited.

The limitations have a strong influence on what a computational design process can do, and what is impossible. The main implications are:

1. Since each different design produced is connected to a different state of the machine, and the number of states is limited (limitation one), the total set of different designs that a program can generate is limited and defined *a priori*. Another way of putting this restriction is that every design has to be represented by design variables, and each variable has to be stored in memory, limiting the number of variables and therefore designs.
2. Due to limitation two, the set of different designs that a program can produce and evaluate in an acceptable time frame is limited. In practice, this number is usually much smaller than the set of possible designs. This means that the space searched usually has to be much smaller than the space of possible designs.
3. The set of designs a program could produce in a limited time frame is not generally known *a priori*. As Langton [12] observes, Turing's halting theorem can be expanded to show that

   > "It is impossible in the general case to determine *any* nontrivial property of the future behavior of a sufficiently powerful computer from a mere inspection of its program and initial state alone."

   Even with a knowledge of the representation and the program, it might therefore not be possible to predict which points in the search space the system can assume, and therefore which designs can possibly be the outcome of the design process.
4. Due to limitations one and two, the evaluation of the design can take into account only a certain limited set of interactions between a design and its environment. For example, individuals in an artificial life application can develop 'vision' only if (a) the individuals have access to some kind of optical sensory organs, and (b) in every time-step of the evolution, a simulation is run to calculate what each individual would 'see' (as is done for example in [23]). However even this would not allow for individuals developing, for example, flight.

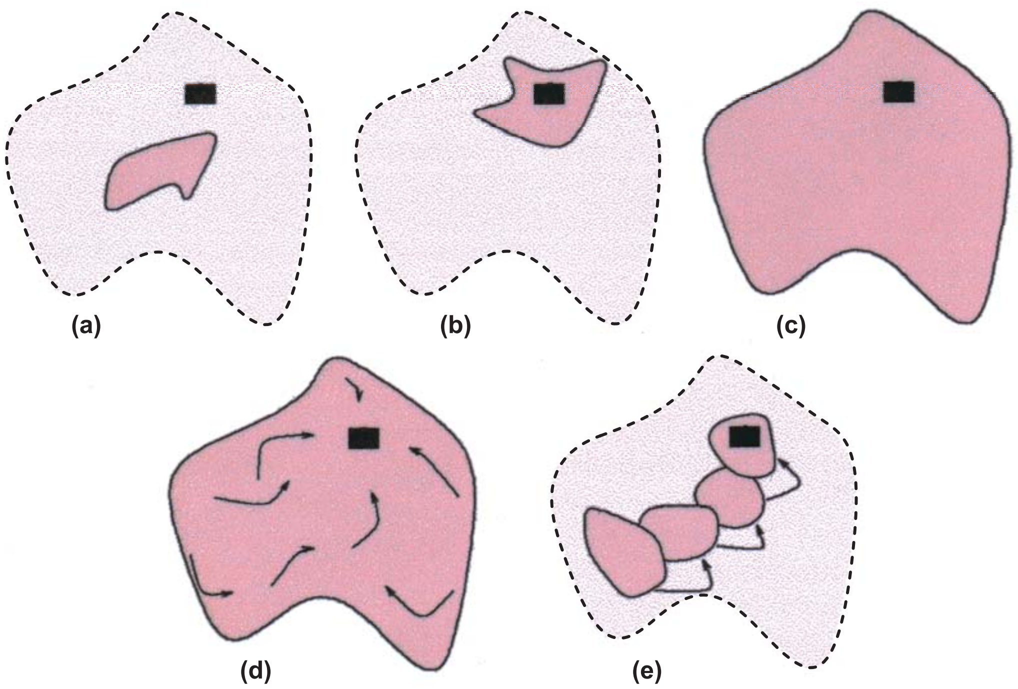

Fig. 6: Methods to search a large search space (black dot: global design optimum; broken line: meta search space, bounded by restriction one; continuous line: search space searched: (a) focus on a subspace, possibly excluding optimum; (b) use domain knowledge to set focus to include optimum; (c) search whole search space; (d) search whole search space using domain knowledge; (e) focus on subspace, but move focus)

The main consequences of the limitations are therefore that the total set of designs that can be produced is fixed, and that usually only a small part of it can be tested by the design process in an acceptable time. If the design process is seen as a search process, it means that the search will always proceed inside a predetermined search space, which can be referred to as the Meta search space; and that of the designs in this Meta search space, the search process can test only a small fraction. However, there are still a number of methods by which this search space can be searched by the design process. The following discussion and the illustration in Figure 6 assume that one design exists in the meta search space that can be considered the optimum in terms of design performance; but the same methods are also applicable if more than one equally acceptable performance exists.

1. A process can search only a small search space, accepting the outcome of this search, even if it represents only a local optimum, and better designs lie outside the search space. For example, the search space can be restricted to designs where methods to optimize them analytically are known. This method is illustrated in Figure 6(a).
2. Using domain knowledge, a search space can be created that is known to contain the desired design. This could be the case in a situation where, say, theoretical analysis shows that all designs outside a certain space give results that violate one or more of the design restrictions. This method is illustrated in Figure 6(b).
3. Searching the global optimum in a large space without heuristics. This could either use a random search or attempt to enumerate all possible designs, Figure 6(c).
4. Using forms of domain knowledge, including 'soft' knowledge and heuristics, to guide the search for the best design in a very large search space. Search strategies such as following the local gradient (hill-climbing), simulated annealing ([11]) and evolutionary algorithms fall under this criterion, as well as guaranteed methods such as logic programming. The arrows in Figure 6(d) illustrate this use of local knowledge.
5. Focusing the search onto a sub-space. The sub-space is searched with any of the other methods. Domain knowledge is used to change the focus inside the total search-space, as shown in Figure 6(e).

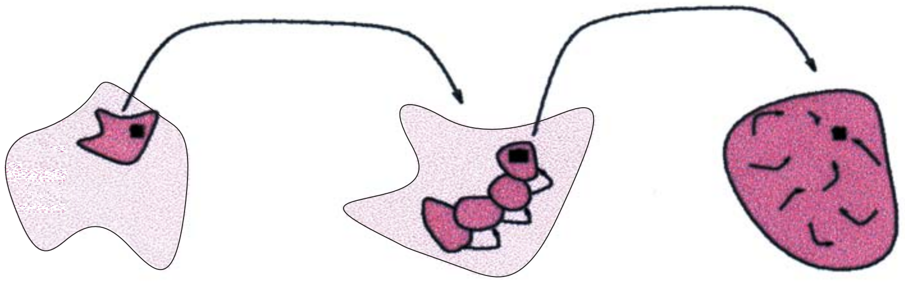

Fig. 7: Hierarchy of search spaces

In practice, the methods will be combined, so that what appears as the meta search space in any of the last three methods is in fact a subset of the set of theoretically possible designs, resulting from the limitation of the search space by either method one or two. With focusing, the search becomes a three-layered process, as shown in Figure 7: the search space is restricted from the Meta search space to a smaller search space, usually using domain knowledge. In this search space, focusing creates smaller sub-spaces, which in turn are searched using local knowledge.

While the limitations of a finite state process are related to the concept of a 'closed world', as used in artificial life (see for example [1]), where it implies total reproducibility, and logic programming (see for example [9]), where the term implies complete knowledge, it is important to note that a closed world is not necessary for the above restrictions to apply. For example, it would be possible to exchange the pseudo random generator used in many programs by a physical device, based on thermal noise or on radioactive decay. This results in a process that is neither reproducible, nor allows complete knowledge; however the rest of the computer is still a finite state machine, and the openness of the world will have no effect on the qualitative outcome of the process.

2.2 Focusing: creativity in a finite system

The focusing in method 5 is very similar to the 'introduction of new variables' and 'modification of search space'. In fact, the difference in definition may be seen as a difference in perspective. Looking at a process as a local observer, who only sees the currently accessible part of the search space, the move of focus in this method appears as a move of the search space. However, a global observer will be able to tell that all successively searched search spaces are in fact part of the larger Meta search space.

An example from the literature can be used to illustrate this point. In [6], an evolutionary system is used to generate beam sections, with perimeter and moment of inertia as two competing design criteria. The sections are represented using a shape grammar; the initial search space S_0 is the set of all designs that can be generated using this grammar. The authors then allow the shape grammar itself to change, and at the end of the evolutionary process a new shape grammar is learned. With this new grammar, a different set of sections can be produced; the system is therefore using a new search space, S_n. The authors observe that $S_0 \not\subset S_n$ and $S_0 \cap S_n \neq \emptyset$, and therefore argue that the change in the shape grammar led to a substitutive change in search space. However, a global observer would be able see that both S_0 and S_n are in fact part of a larger search space S^*, $S_0 \subset S^*$, $S_n \subset S^*$, which may also hold many other designs that are neither part of S_0 or S_n. In terms of focusing, S_0 and S_n both represent a focus inside the space of all designs that can be represented by all possible sets of shape grammars, S^*.

Another example can be seen in the variable addition shown in Figure 2.7(a). If the space of all possible pentagons is considered the original search space S_0, then adding a variable and thereby introducing hexagons creates a new, expanded search space S_n. But it is also possible to argue that both are a subset of the space of all possible polygons, S^*.

2.2.1 *Is focusing creative?*

In terms of [5], most of the methods illustrated in Figure 6 would have to be classified as 'routine design', since the search space that is searched remains constant. Focusing on a sub-space, however, requires that some of the total set of variables are restricted in their range or set to constant values. Moving the focus, then, introduces new variables, and/or uses variables with values outside their current scope. This fulfils the condition for 'creative' or 'innovative' design, not for the entire process, but for the local view onto the focused search.

The position of a search using focusing in Poon and Maher's [18] transformation-exploration matrix (Figure 8) depends on the way the sub-space is searched. As argued above, the process certainly can be classified as 'novel' or 'original' in terms of transformation. If then for example an evolutionary search is used inside the focus area, strong diverging elements are introduced, giving a value of 'novel' or 'original' for 'exploration'. The resulting process is then inside the area of processes with the potential for creativity. A simple hill climbing inside the focus-area, on the other hand, would be entirely convergent, the value for 'explora-

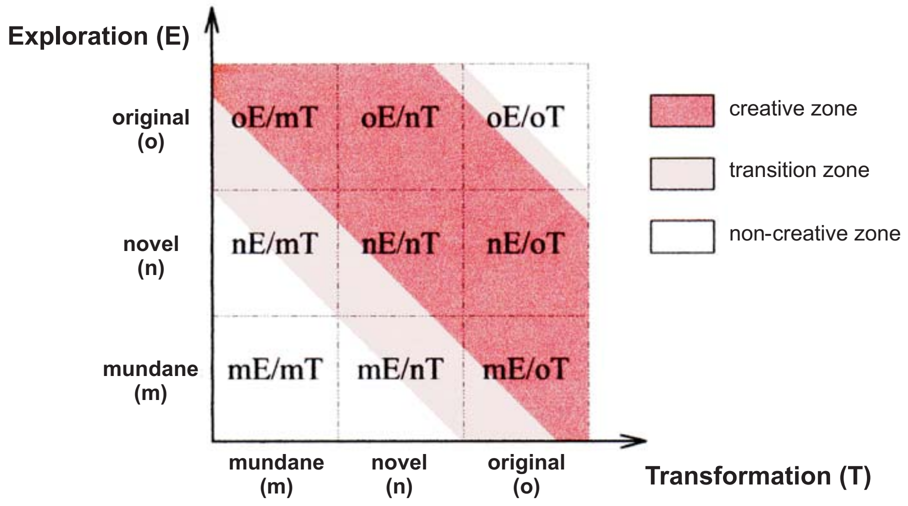

Fig. 8: Framework to classify design processes in terms of exploration and transformation (from [14])

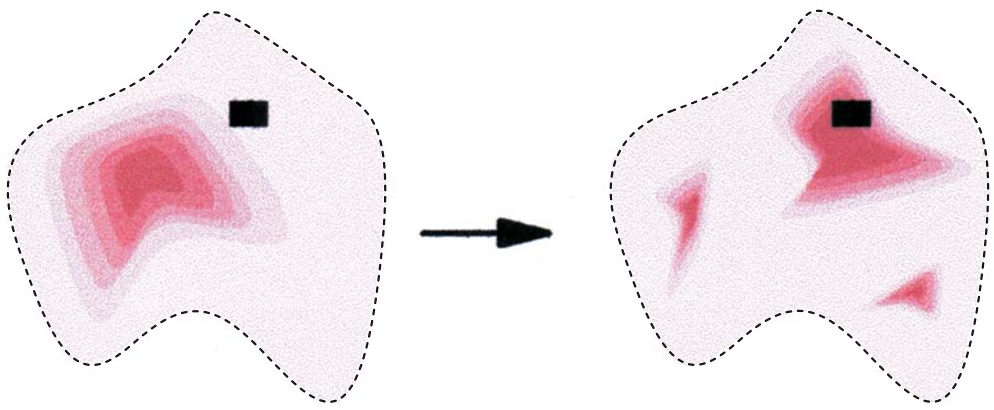

Fig. 9: Transforming a soft focus inside a search space, darker shades represent higher probability for a design to appear as result of the search

tion' would therefore be 'mundane'. Such a process would then be classified as having only a low potential for creativity.

2.2.2 *Soft focus versus hard focus*

The focus does not have to be as clear-cut as in Figure 6. It is also possible to have a 'soft' focus, where all variables can be modified, but some are much more likely to be modified than others, and/or variables are much more likely to assume values in one range than in a different range. In other words, certain points in the search space, those inside the focus, are much more likely to be found than other points. Moving the focus would then correspond to changing the probabilities of the search process. In Figure 9, the soft focus is shown as regions of higher and lower probability inside a search space. As the figure shows, the regions of higher and lower probability do not have to be connected; indeed it is also possible that neighboring designs have very different probabilities, and no distinct regions exist at all.

In the literal sense, moving or expanding a soft focus constitutes neither an addition of variables nor an expansion of the search space, not even from a local perspective, since every design inside the meta search space can always be produced, independent of the positions and shape of the focus. However, this is only a difference in degree, a system where some designs that had previously have been impossible to find are now available (moving a hard focus) will behave very similarly to one where some designs were very unlikely to be found and are now much more likely (moving soft focus). For this reason, it can be argued that moving a soft focus equally well fulfils this requirement for creativity.

2.2.3 *Moving the focus*

Gero [4], Maher, Boulanger, Poon and Gomez [14] define any necessary attributes of the mechanism that drives the transformation or exploration. An entirely random mechanism seems not very useful: if there are only a few acceptable designs in the meta search space, then a randomly positioned focus will have a low probability of containing one of them. This point is also made by Boden when she says that without hunches, a creative robot "would waste a lot of time in following up new ideas that 'anyone could have seen' would lead to a dead end" [3]. Two sources for such 'hunches' have already been discussed: the use of analogy and of emergence. Other sources seem possible, the important aspect is that some connection exists between the current and the transformed focus that improves the probability of finding good designs in the new focus above that of random moves.

2.2.4 *Humans and finite systems*

The previous section has argued that the possibilities for expanding and moving the search space in a computational process are limited by the fact that they have to work inside a finite system. It is an ongoing philosophical debate whether the human mind is essentially nothing more than a complex computational process, or if other, fundamentally different, processes are involved (see for example [17]). However, it is possible to argue that human designers also only focus onto subspaces of a larger Meta search space without requiring any assumptions on the fundamental nature of the human mind. For example, as argued in [13], a very good knowledge both in breadth and in depth about a field is a necessary condition for creativity in humans. Apart from limited knowledge, the number of design alternatives they can consider in a limited time also restricts humans. For complex design tasks this might easily be more restricting for a human than for a computer.

Focusing in human design processes can be directly shown using protocol analysis. In [15], the authors analyze the design behavior of designers during conceptual design. Among other things, they classify the design activities into 'analysis' of the problem, 'synthesis' of solutions, and 'evaluation' of the solutions with regard to the problem. Only during the analysis phase, where the problem is taken into account, does the designer define the search space. The authors report that, as could be expected, the designers observed usually proceeded from analysis to synthesis, and from synthesis to evaluation. However, even in the early stages of the design process, the designers often went from evaluation back to synthesis without a new problem analysis, and therefore without a change in search space. After the initial phase, the designers were about five to six times more likely to proceed from evaluation to synthesis than to analysis. This can be interpreted as a focused search of the search space, interrupted by analysis phases, which allow the focus to be changed.

Evidence of focusing occurring in human design can also be found in the observation that designers tend to reproduce both adequate and inadequate design features of examples if they are given such examples in a design brief, a phenomenon referred to as 'design fixation' [19]. Seeing an example is sufficient to create a focusing effect in the following design activity.

The notion of focusing as an essential component in a creative computational process, as presented in this research, has been derived entirely from general ideas about creativity and computational processes, quite specifically without looking at particular instances of human creative behavior. It is therefore especially encouraging to find this evidence of focusing in human creative design. As in the computational processes, the focus can be 'clear-cut', for example as a result of a decision not to modify some design variables, or 'soft', as a tendency to or as a preference for certain types of design.

2.3 Requirements for a creative design process

From the previous sections, a number of criteria can be identified that a potentially creative computational process should fulfill. The process should:

- Be able to define a non-trivial sub-space, or focus, inside a very large search-space (the meta search space).
- Use divergent and convergent elements in a search process, such that the search is either entirely bounded by the focus, or that designs inside the focus are far more likely to be sampled by the search than designs outside the focus.
- Allow for transformations of the focus inside the meta search space.
- Allow for goal-oriented control of the modifications of the focus.

This characterization differs from those used by [5] [14] in three points:

- It specifically acknowledges that every search space searched will be a sub-space of a larger, predefined search space, the meta search space.
- It specifically allows for 'soft' focus.
- It requires that the control for the modification of the focus is not random.

The requirements form a set of necessary conditions for a creative computational process. Whether they are also sufficient conditions depends on what measure is used to judge the creativity. They are sufficient according to the two definitions used, but as mentioned, Maher, Boulanger, Poon and Gomez [14] carefully limit the definition to the *potential* for creativity. It is very likely that processes exist that fulfill the criteria, but where the designs produced appear not to be creative. Additional requirements, which narrow down the definition, might emerge in future research.

References

[1] Ackley, D. H.: *A network of worlds for research*. In C. G. Langton and K. Shimohara (eds), Proceedings of the 5th International Workshop on Artificial Life: Synthesis and Simulation of Living Systems (ALIFE-96), MIT Press, Cambridge, 1997, pp. 116–123

[2] Bhatta, S., Goel, A., Prabhakar, S.: *Innovation in analogical design*. In J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 57–74

[3] Boden, M. A.: *Could a robot be creative – and would we know?* In K. M. Ford, C. Glymour and P. J. Hayes (eds), Android Epistomology, MIT Press, Cambridge, MA, 1995, pp. 51–72

[4] Gero, J. S.: *Design prototypes: A knowledge representation schema for design*. AI Magazine Vol. 11, No. 4/1990, pp. 26–36

[5] Gero, J. S.: *Towards a model of exploration in computer-aided design*. In J. S. Gero and E. Tyugu (eds), Formal Design Methods for CAD, North-Holland, Amsterdam, 1994, pp. 315–336

[6] Gero, J. S., Louis, S. J., Kundu, S.: *Evolutionary learning of novel grammars for design improvement*. Artificial Intelligence for Engineering Design, Analysis and Manufacturing AIEDAM, Vol. 8, No. 2/1994, pp. 83–94

[7] Gero, J. S., Jun, H. J.: *Visual semantics emergence to support creative designing: A computational view*. In J. S. Gero, M. L. Maher and F. Sudweeks (eds), Preprints Computational Models of Creative Designs, Key Centre of Design Computing, University of Sydney, 1995, pp. 87–116

[8] Grabska, E., Borkowski, A.: *Assisting creativity by composite representation*. In J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '96, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, pp. 743–760

[9] Jäger, G.: *Annotations on the consistency of the closed world assumption*. Journal of Logic Programming, Vol. 8, No. 3/1990, pp. 229–247

[10] Kauffman, S. A.: *The Origins of Orde*r. Oxford University Press, New York, 1993

[11] Kirkpatrick, S., Gelatt, C., Vecchi, M.: *Optimization by simulated annealing*. Science 220, 1983, pp. 671–680

[12] Langton, C. G.: *Artificial life*. In C. G. Langton (ed.), Artificial Life, Vol. VI of SFI Studies in the Sciences of Complexity, Addison-Wesley, Reading, 1988, pp. 1–47

[13] Maher, M. L., Zhao, F., Gero, J. S.: *Creativity in humans and computers*. In J. S. Gero and T. Oksala (eds), Knowledge-Based Systems in Architecture, Acta Scandinavica, Helsinki, 1989, chapter 13, pp. 129–141

[14] Maher, M. L., Boulanger, S., Poon, J., Gomez, A.: *Exploration and transformation in computational models for creative design processes*. In J. S. Gero, M. L. Maher and F. Sudweeks (eds), Preprints Computational Models of Creative Design, Key Center of Design Computing, University of Sydney, 1995, pp. 233–265

[15] Mc. Neill, T., Gero, J. S., Warren, J.: *Understanding conceptual electronic design using protocol analysis*. Research in Engineering Design 10/1998 (to appear)

[16] Nagasaka, I., Yamagishi, A., Taura, T.: *A methodology of emergent shapes for creative design*. In J. S. Gero, F. Sudweeks and M. L. Maher (eds), Preprints Computational Models of Creativity, Key Center of Design Computing, University of Sydney, 1995, pp. 117–130

[17] Penrose, R.: *The Emperor's New Mind*. Oxford University Press, New York, 1989

[18] Poon, J., Maher, M. L.: *Emergent behavior in co-evolutionary design*. In J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '96, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, pp. 703–722

[19] Purcell, A., Gero, J. S., Edwards, H. M., Matka, E.: *Design fixation and intelligent design aids*. In J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '94, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 483–496

[20] Qian, L., Gero, J. S.: *A design support system using analogy*. In J. S. Gero (ed.), Artificial Intelligence in Design '92, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992, pp. 795–816

[21] Qian, L., Gero, J. S.: *An approach to design exploration using analogy*. Preprints Computational Models of Creative Design, Key Centre of Design Computing, University of Sydney, 1995, pp. 3–36

[22] Wolverton, M., Hayes-Roth, B.: *Finding analogues for innovative design*. Preprints Computational Models of Creative Design, Key Centre of Design Computing, University of Sydney, 1995, pp. 59–84

[23] Yaeger, L.: *Computational genetics, physiology, metabolism, neural systems, learning, vision and behaviour or PolyWorld: Life in a new context*. In C. G. Langton (ed.), Artificial Life III, Vol. XVII of SFI Studies in the Sciences of Complexity, Santa Fe Institute, Addison-Wesley, Reading, 1992, pp. 263–298

Ashraf Fouad Hafez Ismail
phone: +420 602882128
fax: +420 2 6517811
e-mail: hafez@fa.cvut.cz

Department of Design Theory
Czech Technical University in Prague
Faculty of Architecture
Thákurova 7, 166 29 Praha 6, Czech Republic