A Computational Model of Creative Design as a Sociocultural Process Involving the Evolution of Language

Anhong Zhang

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

Design Lab
Faculty of Architecture, Design and Planning
University of Sydney

May 2017

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

This thesis contains material published in (Zhang and Saunders, 2012) and (Zhang and Saunders, 2014). The former is section 4.2 Compositional Representation of Rectilinear Relation; and the latter is section 5.1 Ambiguity. I am the corresponding author of the two papers.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Anhong Zhang

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor, Professor Rob Saunders, who has led me into a wonderful new world, Computational Creativity, which is filled with numerous surprising and exciting research topics. His insights and profound knowledge of the field provided me with an invaluable guide as I sought to understand the principles hidden behind phenomena. To make new ideas easy to understand, he drew diagrams and wrote descriptions directing my future study that impressed me every time. Under his guidance, I have published my work in a highly respected forum in our research field. He also helped me to clarify the keystone and structure of my thesis, taught me various invaluable writing techniques and provided numerous important recommendations for revising my thesis. The research study would not have been completed without his kind, insightful and diligent guidance.

My associate supervisor, Dr Oliver Bown, also gave me great support and helped me to publish and present my paper. Sincere thanks to Dr Bown for his kindness, wisdom and patience. Also thanks to Professor Martin Tomitsch for his great and generous support on completing my study.

Thanks to Professor Simon Colton for his precious comments on my submitted paper and many other academics for their excellent presentations and discussions at the conference on Computational Creativity. Thanks to Professor Richard de Dear and Professor William Martens for teaching me Modes of Inquiry: Research and Scholarship. I still remember the story of the "black swan and white swan" that Professor Martens told in the statistics course. Also thanks to Professor Paul Jones for discussing my annual progress. He always encouraged me to study hard and energised me. Thanks to Professor John S. Gero, Professor Mary Lou Maher, Professor Andy Dong, Honorary Associate Paul Murty, Dr Somwrita Sarkar and Dr Lian Loke for their wonderful presentations and insightful discussions. Also thanks to my college mates for their great advice and interesting discussions.

I would also like to thank Dr Cherry Russell for doing such a phenomenal job of editing my thesis, Dr Ricardo Sosa, Dr Kathryn Merrick and Dr Nick Kelly - my examiners, for their invaluable comments and recommendations greatly improving the quality of my thesis, and the Student Administration Centre for their great support with research documentation and facilities as well as the many people I never met for their contributions to numerous open source libraries and research tools, in particular the community of Python and Common Lisp.

Last but not least, I am grateful to my family and friends for their complete support throughout my candidature.

Abstract

The aim of this research is to investigate the mechanisms of creative design within the context of an evolving language through computational modelling. Computational Creativity is a subfield of Artificial Intelligence that focuses on modelling creative behaviours. Typically, research in Computational Creativity has treated language as a medium, e.g., poetry, rather than an active component of the creative process. Previous research studying the role of language in creative design has relied on interviewing human participants, limiting opportunities for computational modelling. This thesis explores the potential for language to play an active role in computational creativity by connecting computational models of the evolution of artificial languages and creative design processes.

Multi-agent simulations based on the Domain-Individual-Field-Interaction framework are employed to evolve artificial languages with features that may support creative designing including ambiguity, incongruity, exaggeration and elaboration. The simulation process consists of three steps: (1) constructing representations associating topics, meanings and utterances; (2) structured communication of utterances and meanings through the playing of "language games"; and (3) evaluation of design briefs and works. The use of individual agents with different evaluation criteria, preferences and roles enriches the scope and diversity of the simulations.

The results of the experiments conducted with artificial creative language systems demonstrate the expansion of design spaces by generating compositional utterances representing novel concepts among design agents using language features and weighted context free grammars. They can be used to computationally explore the roles of language in creative design, and possibly point to computational applications. Understanding the evolution of artificial languages may provide insights into human languages, especially those features that support creativity.

Contents

Abbreviations

Abbr.	Meaning
ACDL	Artificial Creative Design Language
ANN	Artificial Neural Networks
ART	Adaptive Resonance Theory
CAD	Computer Aided/Automated Design
CFG	Context Free Grammar
CSC	Computational Social Creativity
CSG	Context Sensitive Grammar
DARCI	Digital ARtist Communicating Intention
DIFI	Domain-Individual-Field-Interaction framework
FACE	Framing-Aesthetic-Concept-Expression creative act
FBS	Function-Behaviour-Structure process
gs	guessing game
gt	generation game
GA	Genetic Algorithm
GN	Graph Network
GNG	Growing Neuron Gas
HCI	Human-Computer Interaction
HRRs	Holographic Reduced Representations
HSOM	the Habituating Self-Organising Map
IDEA	Iterative-Development-Execution-Appreciation cycle
LTM	Long Term Memory
OpenCV	Open Source Computer Vision
RFBSD	Requirement-Function-Behaviour-Structure-Design
RGBA	Red-Green-Blue-Alpha
SOM	Self-Organising Map
SPECS	Standardised Procedure for Evaluating Creative Systems
STM	Short Term Memory
WCFG	Weighted Context Free Grammar

Chapter 1

Introduction

Language is a critical part of human creativity (Noble and Davidson, 1996), particularly when creativity is considered as a social phenomenon (Csikszentmihalyi, 1999; Gardner, 2011; Vygotsky, 1986). A number of psychological experiments were conducted to discover the role of language in creativity for individuals (Carter and McCarthy, 2004; Glenberg and Kaschak, 2002; Zwaan et al., 2002; Patel et al., 1998). For example, Carter and McCarthy (2004) recorded and analysed informal conversations using creative language related with metaphor, simile and mimicry.

Traditionally, a conceptual design relies on the sketch, which lacks a linguistic level of creativity supported by symbolic connections generated from both words associations exploring new ideas and day-to-day conversations leading to the emergence of creative concepts, i.e., a natural language communication adds significantly to the creative process beyond sketching. As Mitchell (1995) suggested, "language, narrative, and discourse can never—should never—be excluded" from conceptualisation although an image (or idea) may be "abstract" (p. 226). Language can be used to not only emphasise specific design qualities and bring the narrative element in a concept to the fore, but also evolve creative concepts through conversation and communication with association objects (Vyas et al., 2013).

The mechanism involved in the social process of design related to representation and communication remains poorly understood (Stacey et al., 1999). An important part of the design process is expressing and exchanging interesting ideas between participants. In order to brainstorm and evaluate novel concepts successfully, and reach a common creative vision, a shared language needs to be developed. Designers and artists would benefit from the ability to reuse successful metaphors or to learn from unsuccessful ones based on their shared language (Mamykina et al., 2002).

The studies presented in this dissertation address the issues mentioned above by using computational simulations of the evolution of language (Kirby et al., 2008; Steels, 1995;

Vogt, 2005) in artificial creative systems (Saunders, 2011). The studies are developed to support the reasoning about the role that language plays in creative design, and the principle features and mechanisms of language for social creativity. The computational model is developed to be as simple as possible while maintaining the critical elements of using "grounded" languages, which are the combinations of arbitrary utterances and basic referents related with states and events where meanings are generated by the contextual broadening of grounded symbols from sensorimotor projections of objects in particular situations (MacDorman, 2007). In simulations of the evolution of language "grounded" languages are learned as a consequence of use in "language games".

Wittgenstein (1958) developed the concept of "language games". As a thought experiment, Wittgenstein (1958) describes how the meaning of a word/sentence may be generated in a way that is analogous to making a move in a game. Wittgenstein (1958) provided two examples including the description of "five red apples" to a seller for completing a shopping task, and the communication between a master and his assistant for implementing cooperative actions. Through these two examples, Wittgenstein (1958) suggested that a language evolves in a shared context using repeated language games with the cycling of asking, answering and confirmation. In each game, one individual called initiator points to an object; another individual called learner names the object by uttering it; then the initiator confirms the answer (Wittgenstein, 1958). Another type of process can be the learner repeating the words after the teacher (Wittgenstein, 1958). Both of the two processes may lead to the assimilation of language, which is learned by ostensive teaching from representing objects to counting objects, e.g. "a, b, c slabs" representing "one, two, three slabs", and signifying objects, such as "this, that, here, there"(Wittgenstein, 1958). Through language games, the ambiguity of languages occur due to both the multiple mappings between utterances and meanings, and simplified representations. For example, "Slab!" could mean "Bring me the slab", or just identify "It is a slab" (Wittgenstein, 1958).

By repeated playing of language games, Wittgenstein (1958) suggested that different meanings may be produced as a consequence of trying various behaviours in response to the same word/sentence. Consequently, Wittgenstein (1958) suggested that meanings are dynamic instead of predefined in terms of changeable actions and contexts. By varying the rules different types of language games can be devised. Several types of language games were implemented in simulation, including "guessing games" (Steels, 1995) and "imitation games" (de Boer, 2000). Further, Saunders and Grace (2008) developed "generation games" and applied them to the modelling of creative designing. The computational simulations implemented in this dissertation explore the features of languages, such as composition and ambiguity, for creative conceptual design by utilising language games.

1.1 Motivation and Research Questions

The motivation for this study is to enable and augment the exploration of artificial languages for creative design. It is related with language evolution, the creative features of language, and its social dimensions. The following questions form the basis of this research: How to evolve artificial languages to support creative designing? How to use the creative features of language in design? How do artificial languages support social creativity? What can be learned about exploring the role that language plays in creative designing through computational simulations?

1.1.1 The Evolution of Artificial Languages for Design

How does the evolution of artificial languages support designing? The process of evolving an artificial language for design may be initialised by having agents connect randomly generated utterances with the features of designs, then exchange and share their representations. Over time, the shared representations become a new part of the language, grounded in use, as more complex compositional symbols develop to describe designs. Through the computational simulation of these phases, the artificial languages suitable for supporting creative design can evolve to provide a platform for addressing the questions about the relationship between language and design. For example, in the evolution of artificial languages, what factors might determine the generation of different language structures suitable for supporting creative design?

How does creative designing affect the evolution of language? The evolution of languages can be used in the creative design process, e.g. in the articulation of a new design brief to drive a generation of novel design works. Such creative production, by definition, produces novel artefacts and these may require a new language to be used to describe them, expanding the repertoire of language that may be used in the future. Thus, language evolution and creative design can be seen as connected creative processes of exploration in different domains.

1.1.2 The Application of Language Features in Design

How to improve design creativity by using the language features such as ambiguity, incongruity, scalability and extensibility? Ambiguity is related with polysemy and synonymy. It enables the development of new concepts by connecting different utterances and meanings. Incongruity occurs when generating unusual combinations of objects and contexts. Scalability is the capability of using adverbs to exaggerate meanings. And extensibility can be used to elaborate original representations to more complicated and interesting descriptions with abundant meanings.

How to realise the creative features of language? The mechanisms of language are composition, decomposition and recursion, which generate the dynamic and hierarchical structure of language. Regarding the hierarchical structure, novel concepts can be produced via the combination of different symbols, the replacement of one symbol with another one, the variation of symbols, and the decomposition of symbols as well as the recursion and iteration of symbols. Through these operations, the creative features of language can be realised in the production of new interesting design representations. For example, an incongruent representation, "fish in sky", can be generated by replacing "bird" with "fish" in the phrase "bird in sky".

1.1.3 Social Creativity Using Artificial Languages

How do artificial languages support social creativity? Language plays an important role in the social creative process, cultural inheritance and innovation beyond being a medium for creative expression, e.g., poetry, and personal creative thinking, e.g., ideation. Language is dynamic: utterances, meanings and their associations change over time according to the specific context, the experiences and interactions of individuals. The dynamic nature of language makes social communication diverse, which can be studied in computational simulations as a means of generating novel concepts.

There are various forms of creative social communication, including imitation, guessing and generation among individuals with different roles, such as clients and designers. Individuals may have different cognitive styles and experience various environmental conditions. This dissertation explores how the simulation of social creativity may be enhanced via communication between different types of individuals with different strategies in the evolution of artificial languages.

1.2 Aims and Objectives

The broad aim of the research presented here has been to explore the evolution of artificial languages in social creative systems. The approach taken has been to develop simulations of artificial creative systems using curious design agents, which have previously been developed by Saunders (2002) to simulate creative design behaviours and the emergence of social creativity (see Section 2.3.2.2). The curious design agents developed in this research support both the evolution of artificial languages and the generation and transmission of designs as a form of social creativity, similar to those initially presented by Saunders (2011).

The curious design agents developed in this research go beyond those of Saunders (2011) by supporting language features not previously supported, e.g. ambiguity and compositionality. To achieve this the agents in the simulations use much more sophisticated

methods to map between utterances and concepts. To this end, the features of artificial languages for creative design are analysed, and the mechanisms of the evolution of artificial languages for social creativity are investigated.

Curious design agents play two roles in the co-evolution of artificial languages and designs. In relation to the evolution of artificial languages, they observe the environment and communicate with each other. In relation to the use of language features and continuing to evolve artificial languages, they interact, cooperate with each other and generate new design concepts. Hence the research sets out to apply language games to the computational modelling of social creativity in design. Its main contribution lies in the extent to which it explores the features of language that can be important in creative design and can be modelled computationally.

To achieve this aim the following objectives were set:

- To develop a computational process of the evolution of grounded language that supports the application of creative features of language in design.
- To investigate the effects of different features of language in computational simulations for creative design using curious agents.
- To develop a computational model of social creative systems that incorporates the evolution of artificial languages.

1.3 Research Contributions

This dissertation contributes to three aspects including the development of a computational model of evolving artificial languages for creative design, the development of metrics for the evolution of artificial languages at the sociocultural level, and new Knowledge generated through experiments.

1.3.1 Computational Model of Evolving Language for Creative Design

A computational model of evolving artificial languages based on the Domain-Individual-Field-Interaction (DIFI) framework has been developed to evolve compositional languages by combining and transforming diverse associations of utterances and design concepts among multi-agents using language games. In other words, a model of an artificial creative system has been developed via the evolution of compositional languages for creative design at the sociocultural level.

1.3.2 Metrics of Evolving Language for Social Creativity

A computational process, representation-communication-evaluation (see Sec. 3.2, 3.3, 3.4 and Fig. 4.20 as an example), has been developed to evolve grounded language for creative design via guessing games and generation games;

The structure and functions of an agent for playing language games are developed to explore language features and generate interesting designs (see Sec. 3.5.3 and Appendix A).

Graph networks are integrated into multi-agent simulations for storing association rules (see Sec. 5.2, 5.3 and 5.4), evaluating association rules (see Sec. 5.3), and are utilised to analyse the results of simulations (see Sec. 5.1).

1.3.3 Knowledge Generated Through Experiments

Section 4.1 Compositional and Holistic Language: Compositional language is more efficient (see Fig. 4.4) and creative (see Fig. 4.6) than holistic language. And several methods of selecting topics are explored. Results show that "random" selection is good at distinguishing topics while "common" selection can improve consistency of an evolved language.

Section 4.2 Compositional Representation of Rectilinear Relation: Holographic Reduced Representations can be used to represent geometric relations (see Table 4.10), which can be mapped to compositoinal utterances via Self-Organising Map (see Fig. 4.14).

Section 4.3 Compositional Language for Shape Combination: Interesting designs can be generated in the evolution of compositional languages (see Fig. 4.21). And the number of designs collected by a client-agent is affected by the agent's preference of novelty. It could collect more designs when its preference is not too low or too high.

Section 5.1 Ambiguity: Design briefs may be more influential than design works due to the ambiguity of language (see Fig. 5.8) and the role of direction played by client agents.

Section 5.2 Scalability: Exaggeration can be used to expand a design space, differentiate and overlap design categories (see Fig. 5.18).

Section 5.3 Incongruity: Incongruent combinations can be obtained by inverse weighted random choice of possible combinations (see Section 5.3.4).

Section 5.4 Extensibility and Other Features: Elaboration can be used to generate complicated concepts via the recursion of context free grammar (see Table 5.19).

Section 6.1 Growing Population: Growing population mixed with mature and naive individuals can be more efficient than fixed population in the evolution of language (see Fig. 6.3).

Section 6.2 Education in Guessing Game: Appropriate two-way education is more creative than unidirectional learning (see Fig. 6.5, 6.6) .

Section 6.3 Clique Formation: The evolution of cliques generating specific designs is affected by individuals' tolerances, communication preferences and the success thresholds of language games (see Fig. 6.15).

1.4 Significance of the Study

The study of artificial creative design languages evolving in computational simulations is important for two main reasons. First, it helps to clarify possible mechanisms of the relationship between language and creativity that are not easily investigated through traditional experiments with humans, e.g. in cognitive science. Secondly, by simulating the evolution of language in a multi-agent environment, it is possible to investigate and clarify the principles of developing social creativity via interaction and communication.

1.5 Thesis Overview

Chapter 2 provides the background information about research relevant to this study from different fields. Previous work on models of creativity, computational simulation, computational creativity, the relationship between language and creativity, computational models of language and social creativity are reviewed in this chapter. It presents cognitive models, social models and systems models of creativity, in particular the Requirement-Function-Behaviour-Structure (RFBS) process and the Domain-Individual-Field-Interaction (DIFI) framework. Both computational individual creativity and computational social creativity are discussed; and the evaluation of computational creativity and relevant models are reviewed. The chapter then provides the description of the evolution of language, language ecology and the creative features of language such as composition and ambiguity in relation to connecting and generating design concepts. Finally, the models based on social creative systems engaged in language games are reviewed.

Chapter 3 describes a computational model of social creativity based on the Domain-Individual-Field-Interaction (DIFI) framework that incorporates the evolution of language as a multi-agent system. The main processes involved in this computational model are representation, communication and evaluation. Three types of representations are described: (1) simple associations between meanings and utterances, (2) Holographic

Reduced Representations and (3) the representations based on Weighted Context Free Grammar (WCFG). Besides the three types of representations, the communication is developed by learning/unlearning the results of guessing games and generation games. The evaluation of the creative system is related to both individual and social levels. The evaluation at each level relies on similar attributes, including novelty (originality), appropriateness (utility), influence, diversity and efficiency. This chapter also describes the implementation of the computational framework using multi-agent simulations, artificial neural networks, and hedonic functions. A multi-agent system is developed to simulate the communications among individuals, monitor the progress of simulations and collect data for analysis. The evaluation of creativity in this artificial creative system is implemented using both the Wundt curve and its variation based on the measurement of the Euclidean distance or cosine similarity between prototype and design/art works.

Chapters 4, 5 and 6 describe the experiments, which address, respectively, the evolution of compositional languages, the creative features of language, and sociocultural creativity. In Chapter 4, the efficiency, discrimination, consistency and density of compositional and holistic languages are compared; geometric meanings are mapped to compositional utterances using different representation mechanisms; and language games are simulated to evolve domain compositional languages in guessing games and recombine or expand the languages to produce interesting works in generation games.

Chapter 5 explores several creative features of language including ambiguity, scalability, incongruity and extensibility. Ambiguity is applied to find new concepts by connecting multiple utterances and meanings and produce interesting results. Scalability is realised by adding modifiers and combining basic sizes with dynamic categorisation. Incongruity is generated using inversely weighted random selection of the possible relations between objects and context. The last creative feature, extensibility, is implemented with recursive combinations from WCFG.

Three further experiments, described in Chapter 6, explore the creativity of language at the sociocultural level. The first experiment compares the efficiency of a gradually increasing population and a large fixed population in the evolution of artificial languages. The second experiment explores a computational model of peer learning and, within the bounds of the simulation, demonstrates that double-direction education is superior to single-direction teaching. The last experiment explores the mechanism of clique formations in the evolution of multiple languages and the influence of individual tolerances, communication preferences, the number of games being played and the diversity of design/art works.

In Chapter 7, the significance of experiments and results is described. The procedures and results of the experiments are compared and evaluated. The strengths and limitations of the experiments are discussed. The reflections on the research are presented.

Four aspects related with the results of the experiments including linguistic conceptualisation, language as social process, social creativity, and computational simulation are evaluated.

In Chapter 8, the conclusions of the thesis of developing a computational model of creative design as a sociocultural process involving the evolution of language are provided, and possible directions for future study, including meta-creativity, positive learning, complex simulations and autonomous creative system, are presented.

Chapter 2

Background

The previous work on models of creativity, computational simulation, computational creativity, the evaluation of creativity, the relationship between language and creativity, and social creative systems involving language games are reviewed in this chapter. It describes three types of models of creativity consisting of cognitive models, social models and systems models. Both computational individual creativity and computational social creativity are discussed. This is followed by a discussion of the evaluation of computational creativity and a review of several evaluation models. Then it describes language ecology and some creative features of language in relation to connecting and generating concepts, and discusses the study of the "evolution of language" based on Wittgenstein's "language games" (Wittgenstein, 1958). Finally, a discussion of computational models of language and social creativity is provided and three types of language games—imitation, guessing and generation games—are presented.

2.1 Models of Creativity

Boden (2004) proposed that creativity can be divided into P-Creativity and H-Creativity; P-Creativity (Psychological or Personal Creativity) is the judgement by an isolated individual that an idea is novel or surprising and valuable, without reference to other individuals. H-Creativity (Historical Creativity) is a judgement that an idea is unique to not only the innovator but also to other people in the world, at least when compared with the historic materials (Boden, 2004). An individual may be either P-Creative, when his/her work is judged by himself/herself, or H-Creative, when his/her work is judged by society to have made an historic contribution (Boden, 2004). Genius is someone who is able to innovate or generate great unpredicted ideas compared with the average level by using personal capabilities including human intelligence, characters and skills as well as the appropriate context or environment for incubating and stimulating the individual's creative capabilities (Simonton, 1999).

Perceptions of creativity do not necessarily remain constant over time but change according to background context and the influence of others. An example of this is Johann Sebastian Bach's music, which was considered outmoded and was largely ignored during and in the years after Bach's death (31 March 1685 – 28 July 1750), with popular interest only revived several decades later (from 1750 to around 1830) when musical compositional styles had moved on (Temperley and Temperley, 2011).

The following subsections discuss in more detail three approaches to modelling creativity: cognitive models, social models, and systems models.

2.1.1 Cognitive Models of Creativity

One of the most influential cognitive models of creativity is Guilford (1967)'s "Structure of Intellect" model, which provided a fundamental structure of creative cognition theory, and impacted on later models of creativity such as Torrance (1968)'s model of divergent thinking that builds on it. The "Structure of Intellect" model organises various cognitive abilities along three dimensions: content, product, and process. Content is related with visual, auditory, symbolic, semantic and behavioural aspects; product includes units, classes, relations, systems, transformations and implications; process consists of cognition, memory, divergent production, convergent production, and evaluation (Guilford, 1967). Cognitive creative behaviours occur during the interactions between different contents such as the associations between symbolic and semantic aspects, and the transformations between divergent production and convergent production.

Guilford's theories of creativity invoke divergent thinking rather than convergent thinking (Guilford, 1967) partially because ill-defined problems offer a great deal of latitude in the way they can be represented or defined diversely (Hayes, 1989). The features of divergent thinking consist of fluency, flexibility, originality, and elaboration which were widely used in the measurement of creative thinking tests from kindergarten to graduate students (Torrance, 1968). Fluency is measured with the number of relevant ideas such as figural images; flexibility is scored by the variety of categories of relevant responses; originality is evaluated with the number of statistically infrequent ideas related with uncommon or unique responses; and elaboration is tested by the number of added ideas demonstrating the ability of concept development (Kim, 2006).

Other cognitive models emphasising the creative process were developed by Poincaré (1913), Wallas (1926) and Hayes (1989). Poincaré's four-stage model of creativity consists of conscious thought, unconscious thought (or incubation), illumination and verification (Poincaré, 1913). Wallas (1926) described the creative process as preparation,

incubation, intimation, illumination, and verification. Hayes (1989) described the creative process as preparation, finding problems, searching solutions and revision[1]. Hayes' four step model of creativity is described below.

2.1.1.1 Hayes' Model of the Cognitive Processes of Creativity

Hayes' model of the cognitive processes of creativity is composed of four steps including preparation, finding problems, searching solutions, and revision. The first step, preparation, is a necessary stage for accumulating domain knowledge besides general knowledge and become familiar with certain conditions such as the benchmarks used in the standards of great artworks which could be developed or broken through in later steps (Hayes, 1989). The capability of problem recognition is obtained by accumulating enough domain knowledge that makes a novice become potential creative expert (Csikszentmihalyi, 1996).

The second step, finding problems, is to research the knowledge learned from the first stage and analyse relevant situations and personal experience to find confusions. Einstein and Infeld (1971) claimed that the "formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill. To raise new questions, new possibilities, to regard old problems from a new perspective requires creative imagination and marks real advances in science". Good representations of problems can improve the efficiency of finding solutions and be more creative (Lesgold, 1988). Therefore, defining and framing the design problem is a key step of creativity. The more time a subject spent in understanding the problem to form conceptual structures, the better may creative results could be achieved (Dorst and Cross, 2001).

The third step, searching solutions, is composed of two stages, divergent thinking and convergent thinking. As Gero (1994) proposed, a design process can be divided into two stages: conceptual designing and detail designing. The former is likely to explore possibilities while the latter focuses on matching criteria. In the early stage, an initial concept may not be fully formed. Therefore, new areas could be explored to enrich the concept, create novel and useful structures and shapes, and even generate new insights. Three features of designing are finding the possibility, suspending judgment and inspiring imagination. If a judgment is performed too soon, the potential creative "seed" has no chance to reveal its creativity. Therefore, delaying evaluation could be an appropriate strategy. Hence, divergent thinking plays the main role in digging up more potentials. Defocused attention related to diversity refers to the ability to consider

[1]Hayes' model not only matches the process aspect of fundamental cognition theory provided by Guilford (1967) very well, but also clarifies the process to be more logical and complete compared with other cognitive models such as Poincare's model, which lacks an important creative factor, questioning. More recent cognitive models mainly follow Hayes' model such as Plattner et al. (2010)'s design thinking model including five phases: (re)defining the problem, needfinding and benchmarking, ideating, building, and testing.

numerous elements simultaneously, rather than limiting attention to only a few elements (Kaufman and Sternberg, 2010).

Convergent thinking includes grouping, prioritizing and filtering that are used to find appropriate solutions. Similar ideas and concepts become grouped together and the importance of each group of concepts may be ranked to improve the efficiency of collection and analysis. Further, filtering may be useful to eliminate several possibilities for next evaluation (Wheaton, 2014).

The fourth step, revision, is to make cognition more efficient and prepare for future creative events. Effective action can be taken to revise the shortcomings that are evaluated based on the result of previous cognition behaviours (Hayes, 1989).

2.1.1.2 Gero's Model of Cognitive Processes in Creative Design

Gero (1990) put forward that the cognitive processes in creative designing consist of goal-oriented, constrained, decision-making, exploration, and learning activities that operate within a context. The context is changeable due to both the dynamic nature of an environment and the generated design becoming part of a new context, which is affected by the designers' perceptions. Creative design is capable of producing paradigm shifts by introducing new variables into a design prototype that consequently generate novel design prototypes (Gero, 1990).

To identify the mechanisms of creative design, Gero (1994) developed the process of Function-Behaviour-Structure (FBS); then, Christophe et al. (2010) extended FBS to RFBS (see Fig. 2.1). Designers begin with the required functions or structures provided by clients. The transformation between functions and structures are realised by comparing and evaluating the expected behaviours resulting from functions and the behaviours derived from structures. A function can be realised using different structures while a structure can be used to realise different functions due to both various expected behaviours and derived behaviours.

As can be seen from Fig. 2.1, function is transformed from requirement besides being retrieved from structure while structure is embodied into new design. The relationship between Function, Behaviour and Structure are developed by several methods including formulation, synthesis, analysis, evaluation, documentation and reformulation (Gero, 1990). Reformulation occur when behaviours are adjusted and changed repeatedly that leads to finding novel and appropriate structures. Creative behaviours contain several phases including concept formation, feasibility, definition, design and production. They may overlap, and in each phase, divergent thinking for exploring possibilities and convergent thinking for finding solutions could occur.

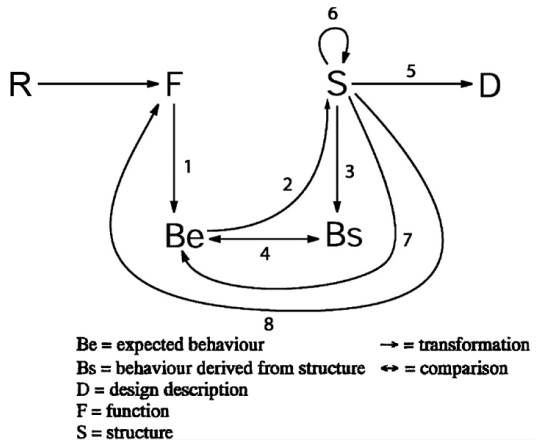


FIGURE 2.1: The Function-Behaviour-Structure (FBS) process of creative design

Gero (1994) proposed that creative design methods based on the FBS framework include first principle, combination (crossover), mutation, analogy (association and transformation) and emergence. First principles are fundamental underlying principles such as postulates in philosophy, axioms in mathematics and established laws in physics (Cushing, 1998). Combination and mutation can be used to generate new designs by mixing and integrating existing components, or mutating them with homogeneous operators. Analogy, which is related to association, leads to the transformation of structure (i.e., transformational analogy) and process (i.e., derivational analogy) from source to target (Gero, 1994). Emergence is the result of self-organisation on a large scale. Emergence includes direct emergence and indirect emergence. The former is generated by direct interactions, e.g. boids (Reynolds, 1986) or a traffic jam, while the latter occurs in the process known as stigmergy (Narzt et al., 2010) by using completed work as the signal for more work, e.g. pathways generated by walking on a grassland. Emergence occurs more easily in non-routine design than in routine design mainly due to the introduction of new variables into the former (Gero, 1994).

It is possible to reflect the FBS process with language games as the transformation between utterances and meanings, which contain both expected extension and observed intension. For example, a client-agent's expected design could be initialised as a prototype particularly represented as an utterance-like requirement; then designer-agents provide their designs to client-agent according to its requirement. The client-agent will compare these observed designs with its expected prototype to select a winning design. Such transformation from requirement to function might be realised using generation games developed by Saunders and Grace (2008). The cycle of problem finding and problem solving might evolve via the communication between client-agents and designer-agents in generation games. Understanding the features of grounded language for design may improve the efficiency of design process, in particular the transformation from utterance-like requirement to function and structure that are related to utterance-extension and utterance-intension respectively. Besides generation games, some other methods such as

AS-IS and To-Be diagrams, and Fit-GAP analysis can also be used to convert requirements into functional specs (Techtarget, 2003). However, generation games are capable of simulating the transformation between requirements and designs interactively and dynamically in the evolution of artificial languages that provides more possibilities of generating creative designs.

The transformation from structure to design is implemented by making components and their relationships to produce a new artifact. The expected behaviours, which are probably associated with the extension of design brief, are used to select and combine the components based on a knowledge of the behaviours produced by structure (Gero, 1994) that may be associated with the intension of design description.

2.1.2 Models of Social Creativity

> *Creative activity grows out of the relationship between an individual and the world of his or her work, as well as from the ties between an individual and other human beings. Much human creativity arises from activities that take place in a social context in which interaction with other people and the artifacts that embody group knowledge are important contributors to the process. Creativity does not happen inside a person's head, but in the interaction between a person's thoughts and a socio-cultural context.* (Fischer, 2005b)

The mechanism of interaction between creative individuals and their sociocultural environment is identified by Vygotsky (1971) in his systems theory. As Vygotsky (1971) stated, creative individuals not just are affected by their experiences of the sociocultural context but also cause changes in their environment by taking actions (Lindqvist, 2003; Saunders and Bown, 2015). A living society continually renews itself via the lifelong learning undertaken by individuals who face continuous emerging challenges from both themselves and the environment (Dewey, 2004).

Sociocultural systems were studied primarily in two aspects including synchronic view and diachronic way. Synchronic view was emphasised on the study of particular points in time and space while diachronic way focused on the processes involved in the development of particular cultures. The former is related with functional systems and structural-function systems. Functional systems see culture as a tool which could be used to satisfy human needs; and structural-function systems regard culture as an adaptive mechanism enabling human beings to live as a well-ordered community in a given environment (Allaire and Firsirotu, 1984).

The second aspect, diachronic research, includes the study of ecological-adaptation systems and that of historical-diffusion systems. In ecological-adaptation systems, the environment not only constrains the cultural development but also channels the cultural

evolution which, in turn, affects the environmental features. In historical-diffusion systems, temporal, interactive, superorganic and autonomous formations are generated by historical conditions and procedures; thus sociocultural systems result from acculturation and assimilation processes (Allaire and Firsirotu, 1984). To explore the evolution of artificial languages for social creativity, this thesis mainly utilises diachronic mechanisms including the dialectical interplay between environment and culture, and social interactions to acculturation and assimilation.

Torrance (1968) argued that creativity is a combination of person, process and product, at least one of which is evaluated as creative. Here, the processes include retrieval, association, transformation, analogical transfer and categorical reduction. These are also described in Finke et al.'s Geneplore model (Finke et al., 1992). Boden (1996) suggested that transformational creativity expands the space by breaking one or more of the defining characteristics or even transforms concepts from the existing space to a new space, while exploratory creativity searches within the space.

Rhodes (1961) introduced the 4Ps model of creativity consisting of Person, Process, Press and Products. The four strands can only operate functionally in unity although each of them has an unique identity (Rhodes, 1961). The first "P" is Person. The most creative people are those who can be very original and yet work within the constraints of the construct (Kaufman and Sternberg, 2010).

The second "P" is Process. According to Kaufman and Sternberg (2010), the creative process consists of three steps: The first step is selective encoding, which involves sifting out relevant information from irrelevant information. The second step is selective combination, which involves combining what might originally seem to be isolated pieces of information into a unified whole that may or may not resemble its parts. The last step is selective comparison, where newly acquired information is related to old information.

The third "P" is Press of the Environment. A person who is creative in one cultural context will not necessarily be so in another. The press of the environment affects how creativity is evaluated by the individual while evaluation is one of the most direct ways for the environment to affect an individual.

The fourth "P" is Products. Kaufman (2009) identified eight types of creative products: replication, redefinition, forward incrementation, advance forward incrementation, redirection, reconstruction, re-initiation, and synthesis.

Besides the 4Ps, persuasion and potential may also be important ways of approaching creativity (Kaufman, 2009). Simonton (1990) proposed "persuasion" with the logic that creative people impact the way others think. Persuasion could also be associated with using ancillary information to help people understand the value of novel artifacts. Runco (2003a) argued that "potential" should be one of the P's. Creative potential is the primary concern rather than unambiguous creative performance. Creativity is widely

distributed and could be found in almost every individual. The key is to transform creative potential to creative behaviour by targeting personal interpretations involving the construction of a new meaning of problem finding and problem solving (Runco, 2003b).

Relationships between different Ps are (1) the relationship between novices and experts (e.g. as novices, children learn from their families and peers), (2) the relationship between individuals and the works in which they are engaged in (i.e., individuals master the knowledge of products, and labour in producing new products, then ultimately revise the nature of their domains), and (3) the relationship between individuals in their world, i.e., people communicate with their rivals, judges, supporters in the field (Gardner, 2011).

2.1.3 Systems Models of Social Creativity

According to Feldman et al. (1994), creativity has three main forms, four levels and four attributes which can be studied in 48 ($3 \times 4 \times 4$) directions. The three forms are trait, process and product adopted to manifest creative process; the levels consist of culture, institution, working group and person; and the attributes are quantitative, qualitative, empirical and normative.

As Piaget (1977) claimed, intelligence can be defined as the state of equilibrium towards which tend all successive adaptations of cognitive development from the sensory-motor stage to mature cognition, as well as all assimilatory (organism $\rightarrow$ environment) and accommodatory (environment $\rightarrow$ organism) interactions between the organism and the environment, i.e., the balance between assimilation and accommodation. The growth of intelligence is thus the growth of the ability to achieve equilibrium at an increasingly high level of complexity (Helmore, 1969).

Vygotsky (1971) argued that social environment bridges individuals and the outside world. It refracts and directs the stimuli acting upon the individual and guides all the reactions that emanate from the individual. Individuals never act alone but instead are always working within cultural and historical channels of practice that mediate their perception of reality and the worldview which is developed through one's interrelated activity in social practices. The works within a genre is taken as a form of social action. "Imagination is also a necessary, integral aspect of realistic thinking" (Vygotsky, 1971). It links art conception and emotion to the spectacle of daily life and its role in socially situated personal growth (Smagorinsky, 2011).

Dewey and Small (1897) claimed that education and schooling are instrumental in creating social change and reform; and "education is a regulation of the process of coming to share in the social consciousness; and that the adjustment of individual activity on the basis of this social consciousness is the only sure method of social reconstruction".

2.1.3.1 Autopoietic Creative System

"Autopoiesis" was introduced by Maturana and Varela (1991) to identify the concept of self-maintaining living cells. Autopoiesis was also utilised in systems theory and sociology.

> *An autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components which: (i) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (ii) constitute it (the machine) as a concrete unity in space in which they (the components) exist by specifying the topological domain of its realization as such a network* (Maturana and Varela, 1991).
>
> *The space defined by an autopoietic system is self-contained and cannot be described by using dimensions that define another space. When we refer to our interactions with a concrete autopoietic system, however, we project this system on the space of our manipulations and make a description of this projection* (Maturana and Varela, 1991).

As Luhmann (1995) suggested, "an autopoietic system reproduces both its reproduction and the conditions for its reproduction". In an autopoietic system, processes and structures are interdependent. Autopoietic systems may be regarded as historical systems (Iba, 2010). The historical structures occur during the process of self-renewal and within the interactions among elements. Then the structure, which is the whole pattern of the relationships among the elements, could affect process. Therefore, processes breed structures while structures guide processes.

Luhmann (1995) applied ideas of autopoiesis to society across many domains, in particular creative domains. "Society is an autopoietic system whose element is communication; and mind is an autopoietic system whose element is consciousness" (Luhmann, 1995). Through social communication and individual consciousness, two-levels of autopoiesis are generated in Luhmann's model. "Elements emerge only when a synthesis of three selections occurs: the selections of hetero-reference and self-reference, and the combination of both". Hence, self-development can occur through inner "mutation" (self-reference) and outer trigger (hetero-reference)[2] (Luhmann, 1995; Iba, 2010).

Luhmann (1995) claimed that communication should be considered as the social phenomenon related to meaning. The same information can be associated with different meanings in different autopoietic systems or in different generations of the same autopoietic system. For example, the colour green looks cold against an orange background,

[2]Hetero-reference means any reference which references entities other than the system itself (such as the environment)

whereas the same colour looks warm against a blue background. In other words, different meanings are obtained in association with different contexts. People coming from different environments share various experiences about the same concept, leading to new meanings of the concept and even new concepts. However, they need to share the same language as a bridge to clarify and transform their thoughts. But the synthesis of information, utterances and understandings cannot be preprogrammed by language. It has to be recreated from the situation by referring to the previous communications and to the possibilities of further communications, which are to be restricted by the actual event (Luhmann, 1995). This operation requires self-reference (Luhmann, 1986), which is emphasised on the function of language within the self-reproductive economy of social communication systems (Luhmann, 1995).

Luhmann (1995) put froward three functions of a language: presentation, expression and appeal. Presentation is the selection of information while expression is the selection of utterances. Appeal is the acception of these selections. The difference between information and utterances in communication impels the evolution of language, which provides the reflexivity of the communication process leading to self-steering.

Iba (2010) claimed that design concepts can be generated and updated iteratively via the loop, "ask-explore-answer-ask-explore...". This consequence is the combination of association (self-reference) and idea (alter-reference) that contributes to the emergence of discoveries which are the elements of a creative autopoietic system. The three selections consisting of idea, association and consequence exist only inside the system. Idea is hetero-reference to the environment while association is self-reference to the system itself. Therefore, "creative systems are recursively-closed systems with respect to discoveries", and the selections result in the reduction of complexity in contingent situations (Iba, 2010).

2.1.3.2 Domain-Individual-Field-Interaction Framework

To identify the structure and process of social creativity, Csikszentmihalyi (1999) developed the Domain-Individual-Field-Interaction (DIFI) framework (see Fig. 2.2). According to this model, individuals create a great number of variations; the field selects some of them and puts them into the domain. Then the domain transmits the selected variants to a new generation of individuals, mainly by training and learning.

In the DIFI framework, Domain, Individual and Field play different roles (Csikszentmihalyi, 1999). Domain takes the responsibility for storing certified knowledge such as structures, processes, rules and cultures, as well as classifying and categorising them. A domain can import resources from other domains (Feldman et al., 1994), e.g. mapping natural ecosystems to human economy.

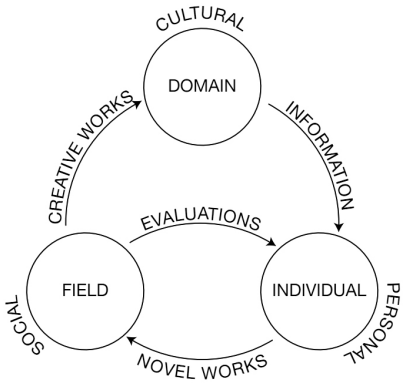


FIGURE 2.2: The Domain-Individual-Field-Interaction (DIFI) framework (Saunders after Csikszentmihalyi)

Individuals play the role of generating various works by learning from the domain, the external environment and themselves. Individuals not only master the knowledge contained in a domain, but also practise within the social constraints of a field, which requires them to play the role of boundary pushing to expand the domain knowledge by taking advantage of the constraints (Feldman et al., 1994). Due to differences in personality, individuals can be categorised as "keepers" repeating and strengthening existing design patterns, "makers" creating and developing new design models, and "breakers" breaking existing design rules and creating novel designs. In addition, they can communicate with each other, modify and improve their works. Creativity can increase with expertise or naivety, that is, when people know "a lot" or "very little" about the activity they are engaged in (Candy and Bilda, 2009). Creativity could also increase during the interactions between naive individuals and experts.

Fields provide a platform for assessing and selecting design works via the interactions and communications between creators, clients, customers, experts, professionals, authorities and institutions. Within this framework, assimilation, accommodation and transformation (Feldman et al., 1994) could occur. Assimilation refers to the tendency to keep reality just as it is, while accommodation represents the tendency to make adjustments in how reality is interpreted when it becomes impossible to live with the distortions that assimilation demands. Transformation refers to the change from one level/space to another level/space related with new processing, status, rules and knowledge based on learning and interaction (Feldman et al., 1994).

2.2 Computational Simulation

Computational simulation is useful when the phenomenon to be studied is not directly accessible or is difficult to observe (Davidsson, 2002). Simulation helps to study the emergence of macro-level phenomena driven by individual actions, the influence of macro structure on individual behaviours, the consequences of social networks and interactions,

and the implications of learning and adaptation. It allows the development of tools that can inform practical actions and contribute to scientific theory (Gilbert, 1999).

2.2.1 Simulation with Theory and Reality

Computational simulation is a mediator between theory and data to help to explain social phenomena (Sawyer, 2004). Unlike mathematical modeling, computational modeling is not limited by traditional mathematical tools. It enjoys expressive power while remaining precise (Sun and Naveh, 2007). Agent-based modeling is appropriate for the domains characterized by a high degree of localisation and distribution and dominated by discrete decisions. Equation-based modeling is naturally applied to systems that can be modeled centrally, and in which the dynamics are dominated by physical laws rather than information processing (Davidsson, 2002). Both equation-based and agent-based modelling have empirical connections and have value in understanding complex societies (Byrne and Callaghan, 2013). Computational models score well on internal validity, statistical conclusion validity, external validity and construct validity when compared with field experiments and laboratory experiments, except for a subcategory of correspondence of the model where field experiments are better. This is the concern for realism and why realism must be addressed in computational modeling (Burton and Obel, 1995).

Computational simulation is connected with the real world by processing data collected from society and the environment. A data-driven simulation is the interaction between multi-agent simulations and real world. The frequency of interaction increases as new sources of data become available. The simulation models such as agent-based models can be constructed to support persistent run-time interactions between computer agents and real-world entities via general types of input-output data streams. In this way, agent-based models become data-driven dynamic application systems entailing the capability of incorporating additional data into an executable application and, vice versa, the ability of applications to dynamically steer the measurement process (Conte et al., 2012).

2.2.2 Process of Computational Simulation

The process of developments unfolding through time is an essential basis for understanding complex systems. Process-tracing is a methodology well-suited for testing theories in a world marked by multiple interaction effects, where it is difficult to explain outcomes in terms of a small number of independent variables (Byrne and Callaghan, 2013). The model statement of variables, parameters, relations and the computational process need to be matched. A experimental design is to try different initial settings of simulations to find the best strategy to satisfy purpose while the data analysis is the analysis of comparisons for different parameter values for testing a hypothesis (Burton and Obel,

1995). The process of computational simulation started from initial settings and ended by data analysis mainly follows bottom-up and top-down directions.

A complex system may be simulated to emerge from a collection of interacting objects. The process of simulating complex systems is describing, exploring, modelling, and establishing causality. In a complex system, a structure is the result of actions by objects within the simulation rather than something pre-existing. But a structure may have a causal potential in relation to the possibility of future actions (Byrne and Callaghan, 2013). Complex social systems are characterised by multiple ontological levels with multi-directional connections. The processes are not only from the micro to the macroscopic levels but also back from the macro to the micro-levels (Conte et al., 2012). The Micro-Macro link is the loop process by which behaviour at the individual level generates higher-level structures, i.e., bottom-up process, which feedback to the lower level, i.e., top-down process (Conte et al., 2012).

A probability distribution exists on the upper level while the realisation of this probability distribution is determined on the lower level. The attributes of the population and of the individuals are interdependent (Gilbert and Troitzsch, 2005). In a macro simulation, the set of individuals is viewed as a structure that can be characterized by a number of variables, whereas in a micro simulation, the structure is viewed as emergence from the interactions between the individuals (Davidsson, 2002). These simulations need to be accompanied by the micro-macro-loop theories, i.e., the theories of mechanisms at the individual level that affect the global behaviour, and the theories of loop-closing downward effects or second-order emergence (Conte et al., 2012).

The study of emergent social behaviour has benefited from computational simulation. Hierarchical and multilevel cultural models are needed to take into account the interdependence of cultural features and the interconnection of cultural dynamics with other social processes (Conte et al., 2012).

2.2.3 Types of Computational Simulation

Computational Simulations may not only be used to simulate the reality to predict the future trend or explain existing phenomena, but also be utilised to build up abstract models to explore the principles of social emergence. The former can be named “thick” simulations while the latter can be called “thin” simulations (Kliemt, 1996). Regarding to different causes, the computational simulations can be used to explore the sequence of causes and results to explain existing phenomena. In terms of different purposes, computational simulations may be used to test hypotheses, train managers, understand decision making design organisations etc.

2.2.3.1 "Thick" and "Thin" Simulations

Kliemt (1996) put forward that computational simulations could be "thick" or "thin", where "thick" simulations are detailed, draw on abundant empirical data, and tell the investigator a lot about a specific question. Such simulations are useful in domains which employed case studies traditionally. "Thin" simulations are the tools for "controlled speculation", useful in disciplining theory formation, simplifying and distorting assumptions (Gotts et al., 2003).

In order to clarify which aspects of a specific situation to be modelled are likely to be most important in understanding its dynamics, the use of "thin" simulations may help to prevent the over-interpretation of results from "thick" ones. Because understanding how such systems are distinctive requires understanding of how complex effects can result from interactions between much simpler elements via "thin" simulations (Gotts et al., 2003).

Burton and Obel (1995) discussed the balance of reality and simplicity between "thick" simulations and "thin" simulations. The reality of the model is a central issue. But how close must a model be to "reality" is relative. The balance between "thick" simulations and "thin" simulations should be considered according the purposes and issues which need to be addressed (Burton and Obel, 1995).

2.2.3.2 Cause-Driven Simulations

The causes of phenomena should be identified to clarify the questions: what might cause changes in the future state of the system and what changes would result from such causes. And the causes should be used by agents to take action, which will result in some particular results. The adjustment of parameters has profound causal power in relation to the outcome states, which emerge from interactive agents with the capability of organisation and decision making (Byrne and Callaghan, 2013).

The evaluation of causes is about what worked as a basis of saying what would work, and realist evaluation is always framed contextually and makes claims constructed in applicable conditions (Byrne and Callaghan, 2013). A great number of quantitative variations may result in qualitative changes, i.e., the changes of kind which are more significant than the changes of degree (Byrne and Callaghan, 2013).

Selection pressures cause evolutionary social change, which is a process of increasing structural differentiation to upgrade its longer-term flexibility and adaptability. Social systems are open-ended dynamic wholes interacting with their environments, including other systems (Byrne and Callaghan, 2013). A new adoption is a response to the current level of adoption, and the evolutionary model, in which competing innovations owe their

relative adoption success to extrinsic, environmental factors applying selection pressures (Watts and Gilbert, 2014).

2.2.3.3 Purpose-Driven Simulations

A purpose could be to formulate theories which explain why existing oragnisations behaved in particular ways, to test these theories by comparing the observed past behaviours with the simulated behaviours generated by the models, and to predict how these organisations would behave in the future (Burton and Obel, 1995). The models used for predictive purposes are related with certain mechanisms. They are output-oriented since, for a given set of initial conditions, to show the state of the system evolving in time (Conte et al., 2012). In addition, abstract approaches could be improved using real data as the basis for specifying an agent-based model (Byrne and Callaghan, 2013).

Another purpose might be to explore the implications of reasonable assumptions about organisational behaviours, in order to determine what the world is like when these assumptions are true (Burton and Obel, 1995). For example, if the prevalence of two sex organisms needs to be explained, what the biological world would be like if there were three sexes can be studied (Conte et al., 2012).

Some other purposes could be to determine which organisation forms are suited to particular goals, or to train people to function better in organisational settings and operations (Burton and Obel, 1995).

2.2.4 The Application of Computational Simulation

The identification of a plausible generative mechanism through simulation is not the end of research, but rather a step to further researches and developments (Watts and Gilbert, 2014). The application of simulation in education and engineering is an open-ended process of modification, updating and improvement.

In the 1980s, computational simulation was applied in the development of education tools to help students complete their courses such as physics, chemistry and architecture. In the early stages, the simulation tools used in physics and chemistry reduced the opportunities of understanding the real process of physics and chemistry phenomena for students because early simulation software packages were developed as “black boxes”. Students were only allowed to input data, and the final result would be output directly without any explanation and illustration. Thus they did not know how the result was generated. Over time the simulation software improved. Students could experience each step of virtual experiment and be asked to find errors in the simulations. Similar improvement occurred in the development of simulations for computer aided design tools.

The early architecture software made students feel like engineers rather than designers when using these tools. But the tools were continually updated and new functions embedded into them to enhance the user experience. Consequently, the simulation used in architecture and civil engineering improved the efficiency and possibilities of generating more interesting concepts and structures (Turkle et al., 2009).

Computational simulations have also been applied in the evolution of artificial languages to explore the origin of language and related mechanisms. Computational linguistics, which develops concrete operational models of the processing, can sustain language as a complex adaptive system (Steels, 2012). The language games as a type of computational simulation for social science are mainly used to explore the relationship between the evolution of language and social creativity (Saunders and Bown, 2015). To improve social creativity, the analysis of conversations are important for understanding the functions well described on a specific body of knowledge accumulated through communications. Conversations produce ordered accounts of particular phenomena instead of rules. Thus conversation analysis should be treated as useful data rather than as a prescription for designers (Luff et al., 2014).

Although computational simulation is widely used in a number of fields, it has some limits. The iterative calculations involved in numerical simulations often produce accumulation errors. Such errors have the potential of misleading the evaluation of results and the validity of the model. In addition, simulation is limited with the memory space and computational performance that cause optimising initial settings and processes (Cangelosi and Parisi, 2012). Thus only partial realities are likely to be reflected.

2.3 Computational Creativity

Computational creativity is a subfield of Artificial Intelligence. It is emphasised on studying creative behaviours and events resulting from the simulation and generation of curious agents in computational systems (Saunders, 2002), and developing creative software using artificial tools and networks to improve human creativity and expand the possibilities of creativity (Colton et al., 2009).

The study of computational models of creative systems that exhibit properties associated with creativity is not necessary requiring that the systems "produce" artefacts for human consumption. In other words, it can become self-evaluation of an autonomous system, which can model creativity-as-it-could-be rather than creativity-as-it-is (Saunders, 2011).

The development of Painting Fool generates structural elements using constraint solving, predicts and controls similarity by machine learning, and evolves art pieces and concepts for the invention of fitness functions for generating works via evolutionary methods. In

the process of invention, less fit individuals are often more interesting in unpredictable ways than the fitter ones (Colton et al., 2012), which leads to more creative behaviours.

Therefore, Computational creativity is the research and development of artificial creative behaviours and events to explore the possibilities of creativity even beyond human creativity. Some definitions of computational creativity are listed below.

- *Computational creativity is the study and support, through computational means and methods, of behaviour exhibited by natural and artificial systems, which would be deemed creative if exhibited by humans* (Wiggins, 2006b).
- *Computational creativity is the study of building software that exhibits behavior that would be deemed creative in humans. Such creative software can be used for autonomous creative tasks, such as inventing mathematical theories, writing poems, painting pictures, and composing music. However, computational creativity studies also enable us to understand human creativity and to produce programs for creative people to use, where the software acts as a creative collaborator rather than a mere tool* (Colton et al., 2009).
- *Computational creativity is a subfield of AI, in which researchers aim to model creative thought by building programs which can produce ideas and artefacts which are novel, surprising and valuable, either autonomously or in conjunction with humans* (Pease and Colton, 2011).
- *Computational creativity is defined as the philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative* (Colton et al., 2012).

Some common features of computational creativity are mentioned in these definitions. They are the study and support of artificial creative behaviours, the evaluation of computational creativity by human beings or autonomous systems. Artificial creative behaviours could reflect individual creativity or social creativity, and they need to be evaluated based on some criteria. Three aspects of computational creativity including individual creativity, social creativity and evaluation of creativity are investigated below.

2.3.1 Computational Models of Individual Creativity

Computational individual creativity was implemented in a number of experiments and applications. For example, the Painting Fool developed by Colton (2012) focused on the creative tripod (see Section 2.3.3.3) composed of skill, imagination and appreciation

(Colton, 2008). To improve its creative capability, a number of existing applications were integrated into Painting Fool (Colton, 2012) such as Context Free Design Grammar (CFDG)[3] generative art software (Machado et al., 2010). And the novelty principle, i.e., the production of similar-yet-different artefacts (Saunders, 2002), was utilised to evolve artworks. For instance, only fitness functions with an average between a specific range of novelty values were accepted (Colton, 2012).

Individual computational creativity, however, might need assistance of external networks to grow up. The Painting Fool has to be connected to the Internet for exploring abundant resources and trained via the judgment of human beings to remedy its own limitation (Colton, 2012). Individual creativity does not only come from the unique perspective that the individual brings to bear in the current problem or situation and the individual's personal interest associated with a particular situation, but also result from its life experience, culture, education, background knowledge and social connections (Fischer, 2005b).

An example of improving individual computational creativity to support human creativity by using domain knowledge is IBM Food Truck, from which customers picked up machine-generated surprising and tasty dishes, which were finally prepared by Institute of Culinary Education (ICE) chef. This work shows how bringing data sources from the creative domain and from hedonic psychophysics together with big data analytics techniques can overcome an individual's self-limited selections to yield a creative system. In such system, numerous and continually updated novel and high-quality dishes can be produced based on cognitive cloud platform (Varshney et al., 2013). Initially, the system learns about ingredient and cuisine pairing, and dish composition by combing huge numbers of existing recipes through the Internet. Then it generates quintillions[4] of possible recipes according to the cross-referencing data on the chemistry of the ingredients and hedonic perception theory, i.e., the psychology of people's preference. Finally, the recipes are narrowed down to around 5,000 options by employing sampling techniques (Murali, 2014).

2.3.2 Computational Models of Social Creativity

To explore the emergence of social creativity in computational systems, Saunders and Bown (2015) developed the framework of Computational Social Creativity (CSC), which is based on the Domain-Individual-Field-Interaction (DIFI) model of creative system (Csikszentmihalyi, 1999). The DIFI model builds up a creative system involving the interactions of components consist of the domain, which is a library storing particular

[3]Context Free Design Grammar (CFDG) is a micro-language designed by Chris Coyne to define 2-dimensional shape grammars based on L-systems.

[4]The figure is provided by Pavankumar Murali, an IBM Research staff member in Business Solutions and Mathematical Sciences (IBM Research Blog, 6 March, 2014).

knowledge related with certain culture, the individuals who learn the domain knowledge and transform them to new ideas and concepts, and the field with numbers of social institutions filtering out non-creative outputs and refresh the domain by adding new knowledge and updating valuable knowledge related with big-C creativity which is required by the whole system to occur (see Fig. 2.2) (Saunders and Bown, 2015). In brief, the social creative system includes individual innovators, their knowledge about a domain, and a field or community of experts who decide which individuals and products are valued as creative, for example by using the Wundt curve to illustrate Hedonic Value (Saunders and Grace, 2008). As Csikszentmihalyi (1999) suggested, creativity is an interaction between (1) the person, (2) the domain in which the individual is working, and (3) the field of knowledge experts who evaluate works in the domain. All possible interactions between the three components could enhance social creativity. For example, an individual may be judged creative to the extent that it exhibits several asynchronies and yet can withstand the concomitant strain (Gardner, 1993).

According to the DIFI model, creativity in design can be evaluated by a target population, selected by opinion leaders, and recognised by colleagues. Innovation could be the diffusion of a design solution across a social group. Variation of preferences enables different decisions based on the shared interpret actions (Sosa and Gero, 2005). Collective conditions may affect individual actions, which in turn stimulate structural social changes. A situational fact or that regulates the way in which individuals interact may heavily impact on the operations of designers and social groups (Sosa and Gero, 2005).

Social creativity might be only a mirage without considering individual creativity. However, the emergence of multi-individual creativity not only is beyond single individual creativity but also can evolve by the interactions of individuals without fitness function (Kirby, 2001). Strong CSC, put forward by Saunders and Bown (2015), requires an artificial creative social system to remain creative even when its individual agents are minimally cognitive. Therefore Computational Social Creativity is similar to Artificial Life occurring from the interactions of simple individual behaviours to complex emergence (Saunders and Bown, 2015) without considering individual creativity.

As Saunders (2002) stated, computational models of social creativity should identify how individuals interact. The simplest computational model of individual creativity is the generate-and-test model with the cycling of self-generation and self-evaluation. Liu (2000) extended this model to social creativity with the dual generate-and-test model including individual self-test and external authorised-test of generated works. When the works are finally judged as creative by monolithic field with experts, i.e., gatekeepers, the new knowledge of the accepted works will be embedded into domain.

Kahl and Hansen (2015) claimed that social creativity is influenced by contextual factors such as evaluation strategies and informational diversity. The novelty of an artifact is evaluated and socially validated by the field in order to be included in the domain, which

provides reference to creators and evaluators. The individuals can change the domain's contents by creating and attributing meanings. Consequently, practice in the target domain would be stimulated and refined (Kahl and Hansen, 2015).

Based on the DIFI framework (Csikszentmihalyi, 1999) and the mechanism of Artificial Life (Langton et al., 1989), Saunders (2002) developed the model of computational social creative system and applied it into artificial creative design via the interactions of curious agents without considering a authorised group. The local interactive behaviours of creative agents without centralised judgment may lead to the emergence of computational creativity on a large scale. Artificial creative societies provide the opportunity for exploring and creating novel designs that might not exist in the real world (Saunders, 2011).

2.3.2.1 Rules for Artificial Creative Systems

Saunders (2011) put forward the concept of distributed domain knowledge and its impact on the evolution of language. Normally, the formation of a domain occurs when the members of a field agree upon a stable lexicon for describing a group of works. In this case, a central repository of all knowledge is not required; rather, the domain is distributed among the members of the field, such that no individual has a complete record of the domain. There are different models of the interaction between domains as individuals migrate between their associated fields. The movement allows individuals to both adapt to the lexicons used in different domains and also affects the development of language as agents transport meanings and words from one domain to another (Saunders, 2011).

The rules for an artificial creative system are based on the mechanism of self-organisation and the DIFI framework (Saunders, 2002) adapting the rules proposed by Langton et al. (1989) for Artificial Life. They are described as follows (Saunders, 2002).

- *The model contains a society of agents situated in a cultural environment*
- *There is no agent that can direct the behaviour of all of the other agents*
- *There are no rules in the agents or the environment that dictate global behaviour*
- *Agents interact with other agents to exchange artefacts and evaluations*
- *Agents interact with the environment to access cultural symbols*
- *Agents evaluate the creativity of artefacts and other agents*

These rules are important for the study of design creativity emerging from simple to complex that is essential for realising the objectives of this thesis including exploring a

minimal set of grounded features of language for creative design and the process of evolving language for creative designing. In terms of the strong social attributes of language, this thesis is emphasised on sociocultural events evolved via the interactions between individuals who are self-controlled and implement simple local behaviours rather than the result of personal genius or authorised groups with predefined complex rules. The former is more natural and radical than the latter because the former can be used to discover the mechanism of creative emergence from simple to complex both for the evolution of language reaching abstract level and the generation of creative works via sociocultural interactions of agents based on the primary generate-and-test model without predefined global rules and authority-controlled behaviours.

2.3.2.2 Curious Agents

An active part of the artificial creative design system, curious agents were researched and developed by Saunders (2002). A single curious design agent is developed from a learning agent with long-term memory to recognise novelty by learning an initial set of pattern categories (Saunders and Gero, 2001b). Multi-agents are developed to not only simulate the emergence of design behaviours but also to evaluate the creativity of artifacts and other agents via communication and interaction. Curious design agents were applied to different models of designing such as direct manipulation and parametric design. Effective models of curious design agents were developed to distinguish novel artifacts from normal products (Saunders, 2002).

Curiosity can be either diverse or specific (see Fig. 2.3). The former addresses boredom and ambiguity to explore diverse creativity while the latter focuses on specific goals to enhance the efficiency of the process of finding creative design sub-spaces.

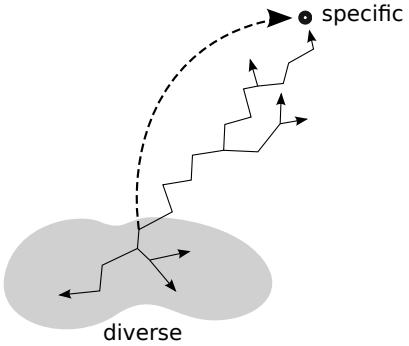


FIGURE 2.3: Diverse curiosity and specific curiosity (after Saunders)

Diverse curiosity is driven by boredom, which is the result of many expected experiences, i.e., an agent is able to predict the results of its actions accurately. When a curious agent feels bored, it will try to do something different to explore new areas to find interesting, novel and surprising designs evaluated using the Wundt curve.

Specific curiosity is driven by goals. Generating specific designs such as incongruent artworks may be a goal for curious agents to explore, e.g., by combining different objects and backgrounds. Some sophisticated design concepts may also be generated using a goal driven model to address embodied design.

Hence, both diverse curiosity and specific curiosity can be utilised by curious agents to explore new design fields. And the development of artificial creativity can benefit from research into new features of diverse curiosity and specific curiosity (Saunders, 2011).

2.3.3 Evaluation of Computational Creativity

Creativity is a process of discovering and creating novel artifacts as well as creating and discovering new values of these artifacts. In terms of both quality and quantity, creativity should be evaluated in several aspects related to novelty, diversity and utility. Among them, higher novelty may be related with deeper and more surprising associations. For instance, Einstein went back to first principles in setting for himself the fundamental problems and in looking for the comprehensive yet simplifying explanatory axioms (Gardner, 2011). Diversity is not only related with the differentiation of results but also relating to divergent processing. Guilford's creative model identifies four components of divergent processing: fluency (i.e., performing intellectual operations smoothly and automatically to produce many ideas) (Sternberg, 1985); flexibility (i.e., producing many different ideas); originality (i.e., producing unusual ideas); and elaboration (i.e., developing these ideas) (Kaufman, 2009).

Within computational creativity Wiggins (2006b) proposed that creativity needs at least two dimensions—novelty and value. Both of them are context dependent (Wiggins, 2006a). Ritchie (2007) added a third dimension, typicality. Novelty addresses the question, "To what extent is the produced item dissimilar to existing examples of that genre?" Quality is related to the question, "To what extent is the produced item a high-quality example of that genre?" However, quality is only a basic essential factor of measuring creativity. Value might be more important than quality for determining the utility of a creative product because the function of comparative performance may depend on value more than the quality of the product. Typicality refers to the extent, "to which the produced item is an example of the artifact class in question?". For example, "Is a generated item a joke or some other creative genre, and how creative is it?" (Ritchie, 2007). Typicality is not only the benchmark of its represented genre for comparing and measuring novelty of new products belonging to the same genre, but also the criteria of creative level compared with different genres or categories, e.g. artworks may have more potential to be creative than standardised daily necessities.

A four-stage model of creativity consists of (i) preparation, (ii) incubation, (iii) illumination and (iv) verification. The final stage, where an idea is consciously verified,

elaborated and applied, may be carried out by either the creator or by the community. Creativity is a function of knowledge, imagination and evaluation (Pease et al., 2001). The criteria of evaluating computational creativity were studied and developed by a number of researchers including Ritchie (2001), Pease et al. (2001), Pearce and Wiggins (2007), Ventura (2008), Colton et al. (2011), Maher (2012) and Jordanous (2013) etc. Among them, Ritchie (2001) identified the evaluation of both products and process; Jordanous (2013) developed the Standardised Procedure for Evaluating Creative Systems (SPECS) while Colton et al. (2011) developed the framework of Creativity Tripod.

2.3.3.1 Ritchie's Formal Evaluation of Creative Systems

According to Ritchie (2001), creative systems can be evaluated in relation to product, process or both. The former is evaluated on the basis of novelty (untypical) and appropriateness (valuable) while the latter is rated according to general analyses and specific implementations such as efficiency and simplicity. Novelty may be evaluated using the concept of a prototype. Prototype theory is based on the fact that humans categorise everyday objects. For example, a robin is a highly typical example of the category bird, but an ostrich or a penguin would be untypical and novel (Ritchie, 2006).

Ritchie (2007) developed a rating scheme to mapping features into the interval [a,b] which could be simplified to [0,1]. Firstly, the typicality of a product is rated as certain genre (poem, story, picture, joke or melody etc) to some extent, e.g. "Is this item a joke or a melody?". Then, novelty is related with untypicality or innovation. Lastly, quality is rated with value. High quality is a good instance of a type of artefact, e.g. "How funny is this joke?" Ritchie (2007)'s evaluation of creativity is primarily based on the comparison of new items with instances (examples), namely an inspiring set, according to two variables which are typicality and value. An item could be novel if it does not replicate the instance. Criteria (Table 2.1) were developed based on the comparison of inspiring set and new items. Among them, criteria 1, 2, 5, 9, 10a, 11, 13, 15, 17 reward typicality while criteria 6, 7, 8, 18 give high scores to atypicality. In addition, criteria 1 to 7 take no account of past artefacts, i.e., the inspiring set, whereas criteria 11 to 18 consider novel results outside the inspiring set (Ritchie, 2007).

Although a number of criteria of evaluation of creativity related with typicality and quality were developed by Ritchie (2007), he also put forward several issues to be addressed including the lack of measuring similarity, the interventions of designer, self-rating, multiple runs, and random generation (Ritchie, 2007).

The measurement of similarity: The evaluation of novelty should involve a similarity measure for events, a probability distribution over events, and the causes of creative events rather than single difference from a inspiring set (Pease et al., 2001). It should

TABLE 2.1: The criteria of evaluating creativity by Ritchie

Criterion(s)	Evaluation(s)	Program behaviour(s)
1, 2	Typical?	Basic success
3, 4	Quality?	Unrestrained quality
5	Typical and quality?	Conventional skill (exploratory creativity)
6, 7, 8	Untypical and quality?	Unconventional skill (transformational creativity)
9	Typical?	Basic success
10a	Typical and interesting?	Avoiding replication
11, 13, 15	Typical?	Basic success, avoiding replication
12, 14, 16	Quality?	Unrestrained quality, avoiding replication
17	Typical and quality?	Conventional skill, avoiding replication
18	Untypical and quality?	Unconventional skill, avoiding replication

be possible to devise a suitable definition of the distance of an item from a set, or even the overall distance of one set of items from another set (Ritchie, 2007).

The contribution of the designer: How to measure the accurate creativity of program taking into account the contributions of the designer of the system? The creativity of program may be measured by comparing the running results and designer's expected results. If the running results surpass the designer's expectation, in particular through evolution, development and multi-agent simulation leading to emergence, the designer's contribution may be not important. However, the program structures and initial settings made by the designer could affect the results fundamentally.

Self-rating of output: The program's rating of its own output ought to consider not only its own behaviours but also the correlation with corresponding external ratings particularly from human judgements.

Multiple Runs: The multiple runs may not be creative if the latter run just repeats the previous one or the runs obey certain expected sequences of change.

Random generation: Statistical tests could be applied to evaluate the distinctions of these sets of ratings. Random generation may be better than fixed settings due to its potentials of exploring a great number of design spaces without bias.

In addition, Ritchie's model doesn't explicitly address the social aspects of creativity evaluation such as diversity, emergence, systemical self-renew, the speed and scale of sociocultural development etc (Fischer, 2005a).

2.3.3.2 SPECS Model

Jordanous (2013) conducted a comprehensive and rigorous review to date of the evaluation of computational creativity, and developed the Standardised Procedure for Evaluating Creative Systems (SPECS), which focuses on assessing computational creativity

through expert evaluation. Jordanous (2013) identified the following 14 key creative components (see Table 2.2):

TABLE 2.2: The creative components of the standardised procedure for evaluating creative systems (SPECS)

Index	Component
1	Active involvement and persistence
2	Generation of results which are novel and appropriate as well as valuable
3	Dealing with uncertainty
4	Domain competence (skills on expert level to generating new ideas)
5	General intellect
6	Independence and freedom (break the existing bondage of context and culture)
7	Intention and emotional involvement
8	Originality
9	Progression and development
10	Social interaction and communication (social influence)
11	Spontaneity/subconscious processing (fluency, acting without thinking)
12	Thinking and evaluation
13	Value (influential and useful)
14	Variety, divergence and experimentation (multi-tasking, divergence without bias)

These components may be grouped into three categories that are concise and clear for efficient expert evaluation: individual process, social process and results (see Table 2.3). Among them, efficiency means that the program should be regarded as more creative than a comparable program which takes longer to produce the same results (Ritchie, 2007) and which requires more effort from the user to justify its results and methods (Pease et al., 2001). Uncertainty can be taken as a creative source for more uncertain and interesting behaviours.

As can be seen from Table 2.3, General Intellect is not included in any one category because some other aspects also require intelligence and so it may not be an independent attribute. In addition, some individual and social processes are interchangeable and affect each other, i.e., the concepts of individual and social process are relative.

TABLE 2.3: The Categorised Components of Creativity

Categorisation	Attribute	Indexes	Explain
Individual	uncertainty	3, 6	
	efficiency	4, 7, 11	time and effort
	evaluation	5, 12	
Social	autopoiesis	1, 7, 9	independence, self-production
	divergence	14	divergence & convergence
Result	novelty	2, 8	
	utility	2, 13	useful and appropriate
	influence	10	benchmark for next generation

Similar research was conducted by Irwing (2010) who suggested that the creative profile can be explained by four primary creativity traits consisting of Idea Generation (originality, fluency, incubation and illumination), Personality (curiosity and tolerance for ambiguity), Motivation (intrinsic, extrinsic and achievement), and Confidence (producing, sharing and implementing). They can help to achieve four goals including "Incubate" (long-term development), "Imagine" (breakthrough ideas), "Improve" (incremental adjustments) and "Invest" (short-term Goals) (DeGraff and Lawrence, 2002).

The process of evaluation is standardised into three steps: defining creativity, standardising the evaluation of domain creativity, and evaluating creativity. In relation to defining a domain creativity, e.g. design creativity, some components should be strengthened, whereas some others can be ignored due to the specific domain context. The style of personal creativity, however, may remain the same in different domains. Forms of representations and topics, as well as process, can serve as the sources of individual difference. Certain components tend more often to operate on certain kinds of representations and topics (Sternberg, 1985).

The results of evaluation from different components (see Table 2.3) made by various experts need to be considered at different levels. More weight should be assigned to important components and experts. The evaluation should include not only the final score but also all sub-test results.

Pease et al. (2013) applied SPECS to the notion of serendipity. The evaluation standards themselves were subject to evaluation, to make sure that they both reflect the intuitive notion of serendipity and are practical to apply to computational creative systems (Pease et al., 2013). However, the process of SPECS (defining, standardising and evaluating creativity) could be too abstract, i.e., for simply describing practical implementations in any empirical investigation (Corneli, 2016). The issue would be addressed by using different ways and with different emphases to suit various purposes (Jordanous and Keller, 2012).

To improve the efficiency of using SPECS, Jordanous (2013) also proposed three future tasks for simplifying and clarifying creative components. They are reducing the number of components used to define creativity, improving the presentation of the component evaluation results to compensate the scarification of useful details caused by the reduction of components, and making the components easier to understand and more clearly defined to eliminate ambiguity and reduce learning curve.

Among all of these components, social interaction, communication and domain competence were shown to be very important in understanding how musical improvisation creativity is manifested (Jordanous and Keller, 2012).

2.3.3.3 FACE/IDEA Model

Colton (2008) introduced the notion of Creative Tripod representing three essential behaviours: skill, appreciation and imagination as a set of necessary conditions for evaluating creativity. According to Creative Tripod, if a program was skillful, appreciative and imaginative, then, regardless of the behaviour of the consumer or programmer, the software should be considered creative when it does not lack any of the three behaviours (Colton, 2008).

Based on Creative Tripod, the FACE/IDEA model was developed to implement four stages: framing the system; judging the aesthetics of the framework; conceptualising within the judgment; and expressing concepts (Pease et al., 2001). The expression of concepts can be represented as algorithms, equations or procedures. Every stage includes the processes and artifacts that implement a loop of evaluating and creating, i.e., a continuous cycle of sensing, thinking and passive or positive action. FACE can also involve a chain of framing, processing and patterning (Pease et al., 2001).

IDEA is a cycle of Iteration, Development, Execution and Appreciation, within which software is engineered and its behaviour is exposed to an audience (Pease et al., 2001) to discover how such creative acts can have an impact on audience and society (Colton et al., 2011). A necessarily iterative process of generation and evaluation is the central feedback loop of creativity. Internal and external evaluation of a product according to aesthetic and framing considerations means that creative product acts as a trigger of creative evaluation and evaluation itself becomes a valuable product (Pease et al., 2001).

Dynamic as opposed to static evaluation reflects social influence (Colton et al., 2011). Rather than comparing only the output of systems, the model proposes to compare entire creative acts, including details of output and processes. Rather than assessing the value of artifacts produced with respect to given aesthetic measures, the model is intended to assess the impact of creative acts, where aesthetic measures may be invented, e.g. designers' works affect client's evaluation rules. Rather than describing/assessing individual generative acts, the model focuses on studying the tuples of generative acts comprising creative acts (Colton et al., 2011). The criteria for evaluating creativity continue to change when individuals experience and discuss novel experiences/works which could become more and more familiar and finally lose novelty. The IDEA model does not yet capture how creative acts can have impact by changing people's opinions, e.g. by introducing them to new aesthetics, nor does it model how people's opinions change through group discussion. Colton et al. (2011) suggested it would be a more powerful tool if group dynamics and causal elements were included.

2.4 Language and Creativity

Language is not only a communication tool but also an activator that stimulates brains to resolve advanced cognitive problems and guides human behaviours (Clark, 1997). In other words, language is a process of organising and exchanging information for performing situated actions (Barsalou, 1999) at the sociocultural level. Language contains signifier and signified. The former is represented as utterances while the latter is what the utterances denote such as object, event, human-beings and agents. Thus language is a process of associating meanings with utterances which are generated and evolved through the interactions between individuals and the environment.

Hockett and Altmann (1968) identified various features of language, of which semanticity, arbitrariness, discreteness, displacement, productivity (composition), duality and reflexiveness are important to creativity. In terms of semanticity, hyponymy can be used to replace one word with another word belonging to the same superordinate such as "cocks laying eggs" instead of "hens laying eggs"; prototype can be used to find unusual meanings such as "penguin" compared with "pigeon" which is a prototype of bird; synonymy, polysemy and homonymy can be used to connect various concepts by tracing between words and meanings (Bagha, 2011). The indirect connection between signals and their meanings (arbitrariness) may lead to ambiguity. Displacement and exchange can generate novel utterances by exploring alternatives. Novel concepts can be produced from existing concepts by breaking, connecting, recombining and compressing existing utterances. The quality of reflexiveness allows an abstract level to be reached using meta-language.

In computational creativity, language is often studied as a creative medium, e.g. poetry. But the primary function of language is not to archive information; it is instead to prepare individuals for situated action (Barsalou, 1999). In addition, memory and information processing related to language comprehension are the results of evolutionary history (i.e., language is a production of evolution). Its purpose is not to process information but to help individuals to cooperate to adapt to the environment efficiently and creatively (Barsalou, 1999). Therefore, language should be inextricably linked with computational creativity as a social process.

2.4.1 Evolution of Language

Language is a distributed and self-organising system that evolves from simple combination and transformation of utterances to a complex entity (Steels, 2000) via composition, decomposition and recursion. Simple utterances can evolve through natural selection such as a bird-call which can be genetically coded, replicated and variated, then selected by nature. But complex utterances have to evolve in the cultural environment by learning. For example, the juvenile male bird replicates his father's song, which might also

be variated and propagated to others. Human language is a culturally evolving system, in which speech, meanings and their relations change rapidly while different words are organised to express more meanings. A language with expressive power, great communicative success and a minimization of cognitive effort could survive and continue to evolve (Steels, 2016).

The evolution of computational complex systems might start from the combination and association of random distributions and the selection of entities with probabilities. In an evolutionary process, the contextual environment provides resources that languages adapt to and take advantage of to enrich the cultural environment.

In the evolution of language, a word can denote several relevant meanings relating to functions and structures. For example, "round", which represents a structural shape, also represents functions and behaviours such as "rotate" and "dynamic". Kirby et al. (2008) conducted the "alien language experiment" in which 100 people were asked to match random shapes and words to evolve an "alien" language. As a result, a non-random alignment of signals and meanings occurred partly due to underlying cognitive biases that iteratively shape the language.

Based on single words, "combinatorial" utterances could be generated to extend meanings matching new states of the environment (Scott-Phillips and Blythe, 2013). For example, putty-nosed monkeys can combine two alarm calls, "pyow" and "hack", to "pyow–hack" which directs the monkeys to a new location instead of simply combining the behaviour of climbing up to avoid leopards with "pyow" and the behaviour of climbing down to avoid eagles with "hack". However, such compositional signal is coded that has only one fixed meaning. In terms of context sensitivity, an ostensive "combinatorial" signal has more than one meaning matching different situations. For example, baby Karen may combine two gestures, pointing and twist, to ask parents to open an out-of-reach object, or just show parents their behaviours of opening the object. The different intentions can be shared among human-beings that make human language unique and powerful (Scott-Phillips and Blythe, 2013). In addition, a large set of languages can be transmitted through a narrow learning bottleneck by the aid of compositional rules, which enable learners to learn an almost infinitely expressive linguistic system (Kirby et al., 2014). Based on the compositionality of language, a number of interesting features such as incongruity, exaggeration, elaboration and recursion can be used in the simulation of evolving language for creative design.

2.4.2 Language Ecology

The interactions between language and diverse natural and sociocultural environments lead to the ecology of language besides symbolic ecology and cognitive ecology (Steffensen

and Fill, 2014). The diversity of language ecology provides a number of potentials of creative events by generating, transmitting and renewing information between individuals and environments. Communities evolving in an open-ended system can not only adapt to new environments but also take advantage of various contexts to improve the environments by exchanging information. For example, ants find food using chemical trails; and human beings cooperate using messages. Intelligence is used to structure our environment via physical action, language and culture so that human beings can be successful with less "intelligence" (Clark, 1997). In particular, speech is a specific and efficient cognitive tool that can be used to highlight new situations and shape problem-solving actions.

The efficient transmission of environmental information depends on activity in multiple situations dealing with different goals. Regarding language, the same meaning can be represented by different symbols and combinations thereof. This results in language diversity. Due to the encoded symbolic combinations saving memory and building meaningful associations, individuals are likely to construct complicated knowledge by using words or symbols directly besides transforming ideas to words. These symbols have the ability to attract and position additional intellectual elements (Clark, 1997).

2.4.3 Creative Features of Language

According to Clark (1997), language is the "ultimate artifact" whose primary purpose is not to communicate ideas between individuals but to overcome cognitive limitations of the human brain through the externalisation of complex thought in a grounded symbolic form. The creative symbolic forms can be developed using several features of language, which are divided into three categories: micro features, macro features and mechanism. The micro features consist of ambiguity, incompleteness, incongruity, scalability and extensibility while the macro features include dynamics and diversity. Mechanism refers to the role of composition and transformation in realising both micro and macro features.

2.4.3.1 Micro Features

The fundamental nature of language is the generation of associations between words and meanings. Multiple associations could become ambiguous, and can be extended to become elaborated, and rematched to become incongruent. In addition, the lack of partial associations may leave the space for developing new meanings and ideas. In brief, the features of language at the micro-level can be ambiguity, incompleteness, incongruity, scalability and extensibility.

Ambiguity: The prime property of language for creative design is ambiguity, such that a small stock of meaningless sounds can be combined in numerous permutations

to make up a large number of meaningful units (Fortuny, 2010). A single word may have multiple meanings and different words may have the same meaning (see Fig. 2.4) that result into the many-to-many mapping. Familiar words might also have unfamiliar meanings. Although such ambiguity may lead to misunderstanding, it can become, along with diversity, a source of creativity. Saunders (2011)'s multi-agent experiments showed that the variety of meanings held by a field for a single word increases greatly as a consequence of individuals searching for novel topics. In particular, the ambiguities that arise in domain-specific languages evolved from the perspective of modeling these languages (Saunders, 2011).

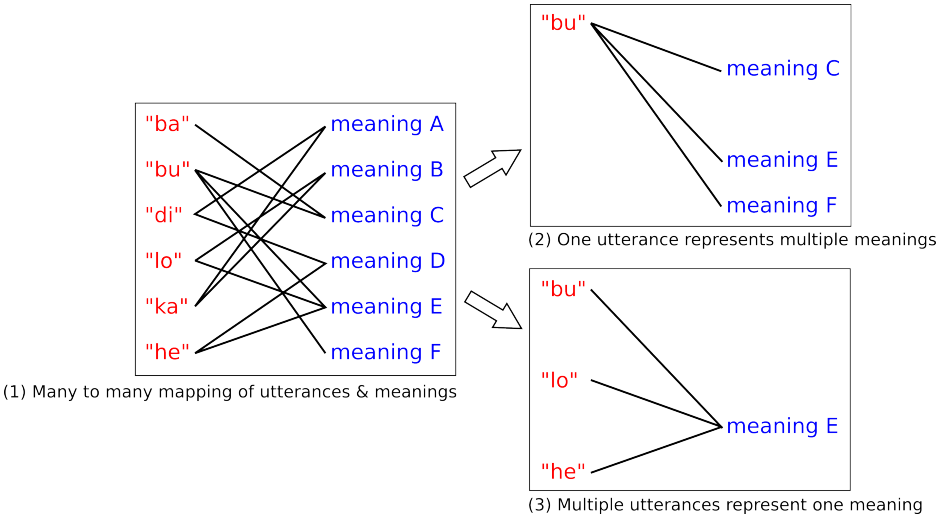


FIGURE 2.4: Many-to-many mapping of utterances & meanings

Practically, the many-to-many mapping generated by connecting multiple utterances and multiple meanings can be utilised to explore creative concepts. The path between seemingly unrelated concepts can be found in the many-to-many mapping. As shown in Figure 2.5, meaning B and meaning D are connected using the utterances including "ka", "di" and "he". In addition, the same brief provided by a client-agent may relate to different meanings for various designer-agents. This reflects the social diversity and creativity of situated agents (Gero and Fujii, 2000) and memory changes in habituation (Marsland et al., 1999).

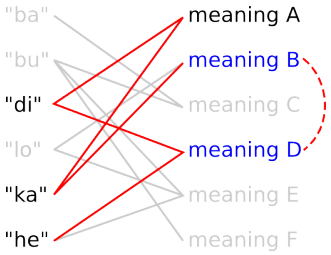


FIGURE 2.5: Connecting seemingly unrelated concepts via many-to-many mapping

Incompleteness: Incompleteness includes inevitable incompleteness and intentional incompleteness.

Inevitable incompleteness such as incomplete knowledge, mis-encoded information and incorrect representation may lead to mis-decoding meanings. The missing part might be generated by guessing or replaced with a new conceptual space (Sternberg, 1985) resulting in novel meanings affected by individuals' experiences and different contexts. The inheritance of languages from one generation to another is also incomplete, thus creating a "bottleneck" (Vogt, 2005). This may result in social creativity by evolving new parts of languages.

Intentional incompleteness (Herrmann, 1997) results from erasing or blocking out parts of memory and reshuffling or reorganising current knowledge that would create opportunities to discover new conceptual spaces. "Loss" of parts of the memory, which might be constraints on creativity, could let agents be more open to new possibilities.

Scalability (Exaggeration): The meaning of an utterance can be exaggerated without having seen an example by adding modifiers like "very" to this utterance. For example, "very small" is the exaggeration of "small". The meanings of some other adjectives such as "medium, large, soft and hard" can also be exaggerated. Interesting designs may potentially be selected by adding a modifier to produce new utterances. A fuzzy set could be used to categorise flexible degrees such as "very small" and "extremely small" with fuzzy boundaries.

Incongruity: Some inconsistent combinations such as unusual embedment of objects into certain contexts, e.g. the composition of fish and sky, may become creative ideas (Berlyne, 1971). Incongruity can also be generated by combining different parts of an artifact such as a suit consisting of a straw hat, a business coat and a colourful skirt.

Extensibility (Elaboration): The existing utterances can be recombined into new utterances or replaced with more complicated sequences via recursive combinations to obtain interesting abundant meanings. For example, the utterances "ka" and "he" can be composed to "ke" via crossover or combined to "kahe" (see Fig. 2.6), as well as "kakahehe" via recursion, which may represent new meanings.

Elaboration also occurs when a sentence is changed by conjoining or embedding (Den Ouden, 1975). For example, the sentence, "A cat is in a whale's stomach", can be changed into a more complicated sentence, "A cat with wings is flying in a whale's stomach", by conjoining, or made longer by embedding a clause, "A cat with wings is flying in a whale's stomach, which is filled with a lot of fish". It can also be transformed into a simpler sentence, "A cat is flying", via deduction and by deleting some details.

2.4.3.2 Macro Features

The flexibility and vitality of language can be represented with the features of dynamic and diversity resulting from emergence of communications on a large scale.

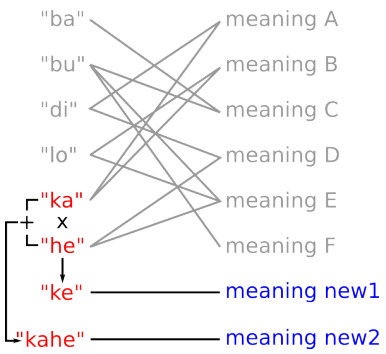


FIGURE 2.6: The extensibility of compositional language

Dynamic: Individuals communicate with each other to propel change in the associations between utterances and meanings through divergent and convergent behaviours. The relations between meanings and utterances are uncertain and changeable due to continuous interactions between individuals, communities and environments.

Diversity: Creativity is related to the diversity of language since different symbols and their combinations can not only represent the same meaning but mutate the meaning to adapt to different situations. The evolution of diverse meanings which are open-ended rather than pre-determined is adaptive to reflexive behaviours.

2.4.3.3 Mechanisms

Different algorithms for combination and association can be used to develop various structures of language. A number of compositional operations including crossover, mutation, overlapping and functional division, and transformations such as analogy and projection can be applied into the evolution of language for generating creative concepts.

Composition: Composition is the integration of combination and association (see Table 2.4). Combination generates a chain of units into long utterances while association links words to all the others that might appear in the same place (Barthes, 1977). Association is related with associability and identity. Associability means that the same results may be obtained by combining units in different ways, or by transforming from one concept to another concept; identity means combining an element with its inverse annuls it, i.e., finding the opposite concept to identify the concept itself (Helmore, 1969). For example, the colour red can be visually strengthened by comparing with its complementary colour green, and be reduced by combining with green.

Creative meanings occur when associating different symbolic combinations. For example, "K-L-M" may be mapped to "N-O-P", but the new sequence "N-O-P" may not represent the same meaning as the sequence "K-L-M". Some interesting combinations can be

generated using genetic algorithm such as "I am a banana" instead of "I eat a banana" by mutation as well as "flying fish" and "swimming birds" by crossover.

TABLE 2.4: The relationship between combination (row) and association (column)

Article	Colour	Noun	Verb
A	red	hat	fly
The	green	helmet	run
Any	blue	cap	swim

The overlapping of two patterns rather than the projection between two spaces might provide additional possibilities for generating creative products, because two separated concepts can be connected by the items in the overlapping area. Overlapping could be used in multi-domains, which include both spatial and historical dimensions—for instance, between two cultures or between early and later stages of one culture. In addition, collective comprehensiveness may be established by the overlapping parts, i.e., boundary objects, in a way that is dynamic for adapting to local needs and constraints while robust enough to maintain a common identity across domains for shared recognition (Fischer et al., 2005).

A language can evolve as a hierarchical structure by composition and decomposition. Decomposition is a process of separation of concepts from general to embodied detail by tracking their associations and categories. For example, "combination" may be decomposed to "overlapping" and "touching". In addition, "overlapping" can be separated as "intersecting" and "corner overlapping". High-level combination can be produced by combining words with different types such as the combination of "fly" (verb) and "fish" (noun) to generate new concepts. Combining words with the same type could also generate new ideas. For example, the combination of "road" (noun) and "train" (noun) translates the idea of a train to run on the road in the form of very long trucks. Further, the features of hierarchy and recursion of language could be simulated via bio-inspired algorithms such as developmental systems and cellular automaton (Floreano and Mattiussi, 2008).

Transformation: The transformation of representations can be realised by discerning the similarities between pair vectors based on the same structure. For example, given the matrix of words in Table 2.4, "a red hat" can be transformed to "any blue helmet" or "some big cap" with different associations, or, as studies of synaesthesia suggest, can be transformed to other signal types such as sound, music and colour as well as shape (Ramachandran and Hubbard, 2001).

The translation from one domain-specific language to another may be completed via the replacement of relevant symbols from each language that represent the same or similar meanings. Consequently, communication may be established and developed between different domains.

Transformation can be used for concept projection: Expect change in conceptual system A → explore new system B → find appropriate concept in system B → replace a concept in system A with the discovered concept in system B → test if failed, restore the original system. → run next cycle... (Sternberg, 1985). During projection, a missing or mismatching part of one space can be represented with a new part or by merging some parts reflected in another space. The transformation of conceptual space may be realised via the separation of rules of representation from search space (Wiggins, 2006a), such that, a new conceptual space may be created by changing the existing rules.

2.4.4 Language and Design

A language is a set of symbols, which reference knowledge. Knowledge includes direct experiential knowledge and indirect conceptual knowledge. The former is received from the environment through senses while the latter is constructed from the former with experiences and imagination. For example, the concept "purple cows" is generated by combining direct knowledge "purple" and "cows". The relationship between languages and concepts is that the former reference the latter while the latter enrich the former. Human knowledge is first and foremost experiential then embellished by logic, reflection and imagination. Words are not the things they represent but tools that help people to explore concepts (Sherzer, 2009).

It can be creative to search descriptions of design spaces besides searching design spaces directly. One meaning and multiple utterances are analogous to one concept and multiple forms. Interesting designs may be found by operating on utterances using composition, decomposition, analogy-making, crossover and mutation. The meanings of these utterances may be different from those of their "parents". Similar utterances may also generate different meanings representing new design spaces.

Concepts are connections between symbols and chunks (Hori, 1994). A chunk is a group of information bound together into a meaningful whole that was used in short term memory and long term memory for efficiently collecting, storing and extracting data (Neath, 1998). For example, the symbolic combination "sports-car" is connected with the chunks, "the power of the car" and "the shape of the car". The meaning of a concept is affected by its relations to other concepts (Borensztajn, 2006). Structured concepts can be developed through classification and categorisation, and represented using compositional languages. Structured design concepts are meaningful and dynamic which can be recombined into novel designs by using the features of compositional language such as ambiguity and incongruity.

Hori (1994) developed a computational system for uncovering conceptual relations which are not apparent to humans by connecting words with different associative distances. The users only need to provide some basic symbols and a few of relations. A word "C"

may approach "A", when "A" has relation to "B" and "B" has relation to "C". This may surprise a user who is unaware of the relation between "A" and "C". Existing words are mapped onto Euclidean space, which can be taken as a stimulus for detecting relations. The system provides multiple planes for arranging words in different contexts. A number of words from plane-A can be copied to plane-B, and new words are put on plane-B to make new connections with the old words. On each plane, a row corresponds to a state while a column corresponds to a type of words. Besides these two dimensions, multi-dimensional presentations can be realised as a spatial arrangement (Hori, 1994).

The system can be used in the process of product development (experiment → measurement → calculation → consideration → design → production). Other processes, such as characteristics-state-aspect-function-performance-theory and object-action-effect, were embedded into the system to facilitate the generation of new concepts. To strengthen the role of connection, the function of setting the weights of words, i.e., the importance of words, the weight of connection, i.e., the importance of relations, and the direction of connection can be added (Hori, 1994).

The application of language in design described above is emphasised on using a system of rules and principles mainly related with syntax and semantics to explore novel conceptual relations. Concepts may represent natural objects, artifacts or abstract ideas and knowledge. a concept does not depend on language but needs to be expressed by language. Novel concepts could be revealed by recombining experienced meanings, which may still be embodied objects but indicate several abstract meanings such as surprising, unusual, incongruent. This thesis is emphasised on compositing the utterances evolved in the experience of "physical" world related with colours, shapes and their relations to get new interesting colours, shapes and relations. Thus only grounded languages related with simple rules and certain interesting meanings are explored without evolving abstract meanings or meta-concepts such as logical structure.

In terms of the social aspect of design, language as a form of communication can be utilised such that design may be modelled as a process of conversations between clients and designers. Conceptual design may be modelled as the analysis of client requirements and the generation of design concepts using a suitable mechanism, e.g., Hori (1994)'s method of spatial arrangement of words. In addition, conversations between designers and clients can be modelled using different types of "language games" such as "guessing games" (Steels, 1995) or "generation games" (Saunders and Grace, 2008). The next section discusses this approach to modelling the social aspects of creative design.

2.5 Computational Models of Language and Social Creativity

Wittgenstein's original proposition that languages can be learned through "language games" (Wittgenstein, 1958) is the foundation of the computational models of the evolution of artificial languages, which were developed by Steels (1995), Kirby (1998) and Vogt (2005). Steels (1995) utilised language games, such as Talking Heads, to evolve artificial language in multi-agent systems. In a guessing game, the initiator-agent describes an object using a simple utterance to the recipient-agent, who attempts to identify the topic of the utterance based on its experience of the previous utterances (Steels, 1995). Consequently, language games can generate a shared lexicon of words and their associated meanings (Kirby, 1998). Steels (1995) showed that repeated playing of language games is capable of evolving languages grounded in shared experiences. Vogt (2005) used language games to evolve artificial languages based on the mechanisms including chunking, exploration and exploitation.

The evolution of grounded language for creative design can start from distributing symbolic units randomly. Then various utterances evolve to represent different meanings, which could be connected to adapt surroundings to embodied design requirements. Saunders and Grace (2008) introduced a new type of language game, generation game, to model a common form of interaction in design at sociocultural level. The introduction of generation games to an artificial creative system enable agents to not only exchange artifacts and evaluations but to negotiate descriptions and meanings for their products.

As Steels (2006b) claimed, the evolution of language through language games is a dual process of alignment and innovation. In terms of alignment, a common language is evolved and shard among agents to maintain a stable community, in which a single individual leaving or joining does not affect the stability of the shared language among enough agents acting as a collective memory. Regarding innovation, new utterances representing new meanings are continually generated to adapt to the open-ended dynamic environment and propel the linguistic system forward (Steels, 2006b). The system may evolve from naming objects to representing categories at a more abstract level. Language games are grounded in symbols connected with abstract concepts by categorising the objects and identifying their relationship within context. The evolved abstract components are contributed by the features of language mainly including composition and ambiguity, which support multiple mappings between ontologies and lexicons (Steels, 2006b). Ontologies may contain abstract concepts, compositional meanings and unit-meanings. Their relationship can be established using association rules connecting roots and expansions, which construct a hierarchical recursive conceptual network. The roots could be abstract and compositional concepts (e.g incongruent scene) whilst the expansions may be abstract or embodied in particular for terminal expansions such as the primary features of an object including shape, colour, size, location and direction.

The relations between these ontologies and lexicons can be represented using association rules connecting meanings and multiple weighted utterances. The evolved association rules as grounded representations are superior for simulations of creative societies than predefined conceptual representations because of their ability to serve a critical role in generative systems. Using such a grounded representation it is possible to construct an entirely new utterance, which can form a goal for a design agent despite the realisation of the utterance being non-trivial.

Various possible ideas and answers can be generated from ill-defined problems given ambiguous conditions via the evolution of artificial language using language games at sociocultural level rather than logical reasoning which might lack flexibility and adaptability. Saunders and Grace (2008) demonstrated that language games can become optimised as a method of communication between client and designer agents in multi-agent simulations for evolving artificial languages, in which coherent lexicons may be developed via the spread of concepts and bias pressure among individuals. In addition, the evolution of specific languages related with certain domains can be used to explore particular "interesting" works. Further, Saunders and Grace (2008) proposed to use the adaptable capacity of curious agents to develop the models of multiple domains that agents can move between. This type of movement may enhance the diversity and creativity of artificial social systems.

The computational models of language for social creativity described above are composed of three types of language games consisting of imitation game, guessing game and generation game, which are described below.

2.5.1 Imitation Game

An imitation game is played by a producer-agent and an imitator-agent (Steels, 2000). The producer provides an utterance randomly generated or selected from its memory while the imitator recognises it based on its own associative memory and then reproduces the utterance. Then the producer tries to identify the imitator's utterance. When it is similar to its own, the game succeeds otherwise it fails. Consequently, the two agents evolve and share a language throughout numerous imitation games (Steels, 2000).

Imitation may be used to computationally model creativity since repetition can be regarded as a resource by means of which conversationalists together create a discourse, a relationship and a world. Imitation game was used to explore the self-organisation of vowel systems (De Boer, 2001) and the evolution of music (Miranda et al., 2003). In these systems, agents imitate each other by producing expressions of the sounds they perceive. The goal of the vowel-formation systems (De Boer, 2001) was to investigate the structural tendencies of vowel systems and to determine where a coherent vowel system could emerge in a simulation.

2.5.2 Guessing Game

A guessing game comprises an initiator-agent, a recipient-agent and a context including several topics such as shapes, colours and objects. In the game, the initiator (i.e., speaker) selects a topic from the context, represents it with an utterance, and sends the utterance to the recipient (i.e., listener), who attempts to identify the topic of the utterance based on its experience of the previous utterances (Steels, 2001a). In other words, the answer is guessed by associating the existing concepts and accessing the answer via the coherence of each agent's own symbolic system. If the guess is correct, the relationship between the topic and utterance is strengthened by increasing its weight. If the guess is incorrect, the relationship between the topic and utterance built by the initiator (i.e., speaker) is learned by the recipient (i.e., listener) while the relationship of the same utterance with different topic built by the recipient (i.e., listener) is weakened by decreasing its weight.

After playing the guessing game a number of runs, agents can complete the game with a high degree of success using the language evolved as a result of the learned mappings from utterances and topics.

Agents continue to reduce misunderstanding and develop a shared domain language during guessing games. Guessing games were also used to discriminate between different agents, between other topics in the agents' context and between real objects as presented to a pair of robotic "Talking Heads" (Steels, 1998) driving the emergence of a shared lexicon.

2.5.3 Generation Game

A generation game is played by a client-agent and a number of designer-agents. In the game, the client-agent expresses the requirements to the designer-agents by encoding the values of features of requirements which could be creative; then designer-agents decode the requirements and produce actions based on their own experiences, and generate designs to the client. If the products satisfy a client's requirements, the game succeeds, the participants will learn the relation between the requirement and the accepted designs; otherwise, they will unlearn the relation (Saunders and Grace, 2008). The more sophisticated the requirements are, the more solutions may be produced.

The difference between guessing games and generation games is that the recipient agents in guessing games select the topics from existing contexts, whereas the designer agents in generation games generate objects based on their experiences and client's requirements. Compared to guessing games, generation games are more open since multiple solutions may be generated by designer-agents (e.g. various shapes generated via the variation of prototypes) that can satisfy the requirement. If the novelty of the new shape falls into

the preferred range for the client agent, the shape may be selected. The preferred range of novelty for an agent is defined by an internal model of preference based on the Wundt curve (see Section 3.7), where similar-yet-different perceptual experiences are preferred (Saunders, 2011).

In generation games, the combination or reconstruction of existing words generated by client-agents may represent new ideas and questions to guide designer-agents in exploring new design areas and realise novel ideas. Some design briefs may be generated by combining several symbols. For example, the combination, "kika + ajad", would provide some new ideas and design problems that need to be revealed and resolved. Similarly, "black triangle" is generated from the reconstruction of existing combinations, "black square" and "white triangle". These linguistic combinations are generated by agents who see the relevant shapes. Then "black triangle", which does not have a real relevant shape in advance, could be a result of problem finding (client's design brief). A new shape will be generated to match the meaning, "black triangle", which is different from existing shapes including "black square" and "white triangle". Through such processes, a number of new designs could be generated by designer-agents to satisfy the client-agent's requirements. Consequently, generation games evolve with the conversations between client-agents and designer-agents who learn from each other.

Most studies of language games are emphasised on the evolution of languages (Steels, 2016) (Vogt, 2005) without considering some specific features of languages related with design creativity except the generation games developed for social creativity (Saunders and Grace, 2008). This thesis focuses on the social creativity (see Chapter 6) of designing supported by language games. And the language features such as ambiguity (see Sec. 5.1), compositionality (see Chapter 4), exaggeration (see Sec. 5.2) and incongruity (see Sec. 5.3) are explored to fill the gap between communication and creative design.

2.6 Conclusion

Two aspects of background knowledge including the models of creativity and computational creativity, and the relationship between language and creativity as well as the related computational models are reviewed. Firstly, the models of creativity related with cognitive behaviours and social interactions are described and the systems models of social creativity, in particular, Domain-Individual-Field-Interaction Framework (DIFI) are reviewed. Then, the computational models of individual creativity and social creativity, and the evaluation of computational creativity were discussed. Secondly, the relationships between language and creativity are described. The evolution of language and the features of language related with creativity including ambiguity, diversity and composition etc., are discussed and the computational models of language and social creativity consisting of imitation game, guessing game and generation game are presented.

Linguistic researchers mainly focus on evolving language and exploring the originality of language but not consider the creative features related with generating new compositional concepts and transmitting creative design information between individuals. Design researchers normally take language as a medium of processing design or study design language such as shape grammar. In other words, the connection between language and design is mainly related with design expression and description that lacks deep understanding of the creative role of language played in design communication. This thesis tries to fill the research gap between communication and creative design via multi-agent simulations of the evolution of language in design contexts, and take language as a creative social tool for design more than a medium, design syntax or artifact semantics.

The next chapter presents a computational model building on the background research covered in this chapter to provide a foundation for the experiments presented in Chapters 4, 5 & 6.

Chapter 3

Computational Model

This chapter describes a computational model of evolving language for creative design based on the DIFI framework (see Sections 2.1.3.2 and 2.3.2) and the implementation of this model used to simulate language games for social creativity in the experiments described in the following chapters. Firstly, a computational model of the DIFI framework (Saunders, 2002) is presented. Secondly, the computational model expands on earlier attempts completed by Saunders and Grace (2008) to model the DIFI framework by incorporating different types of interactions such as guessing/learning, encoding/decoding, and generating/producing in the evolution of language. At last, the implementations of the computational model makes use of artificial neural networks including Self-Organising Maps (SOM), Adaptive Resonance Theory (ART) networks and Growing Neural Gases (GNG), to generate symbols and transmit information through the communication of multi-agents with both short-term memory (STM) and long-term memory (LTM).

To evolve artificial languages for knowledge representation and creative designing, a computational framework is developed to implement three processes consisting of representation, communication and evaluation. The first is to generate "utterances" and match them with some existing design knowledge, e.g. the generation of consonant-vowel pairs matching different combinations of topics such as colours and shapes using associative neural networks. It requires the establishment of appropriate representations of design knowledge and "utterances". The second is to produce new "utterances" by recombining existing artificial utterances and transforming them to new design concepts related to novel design spaces via the communication of client-agents and designer-agents, e.g. "house" $\rightarrow$ "movable house" (recreational vehicle). Effective communication requires the selection of suitable computational algorithms to satisfy the requirements of complicated transformations, i.e., the mapping from utterances to concepts (design knowledge) through inductive means in language games. Finally, the criteria for evaluating design creativity are relative to the design requirements. Appropriate criteria for the evaluation of design creativity include criteria for the evaluation of designs, e.g. novelty and value

considerations, or the process of designing, e.g. novel techniques. In this model, one of important criteria for the evaluation of creativity is the unexpectedness of a design given its description in the form of an utterance. Unexpectedness is determined by an agent's ability to predict an as-yet-unseen event (Saunders, 2002). For example, the predictions made by client-agents about future designs do not match the designs provided by designer-agents in generation games. Such unexpectedness could indicate novelty.

3.1 A Computational Model of the DIFI Framework

The computational model presented here is based on the Domain-Individual-Field-Interaction (DIFI) framework (see Fig. 3.1). In this computational model, agents develop an artificial language by negotiating and obtaining a mainly group-accepted, e.g. more than 70%[1], scheme that maps utterances to design features. Divergent generation of works is completed by individuals; convergent collection is made by field; and individuals are trained by exposure to a domain. Some parts of the scheme will be different in each agent's associative memories although for communicative success they are likely to share a significant part of the language. The individual differences can be the sources of social creativity in multi-agent simulations.

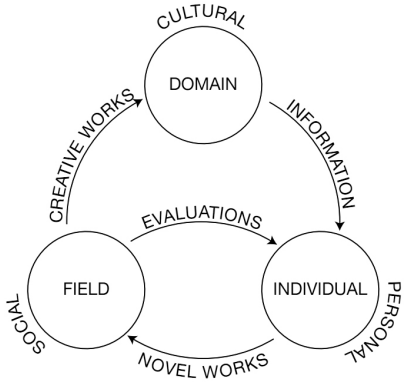


FIGURE 3.1: The Domain-Individual-Field-Interaction (DIFI) framework (Saunders after Csikszentmihalyi)

3.1.1 Domain

The computational model of a domain is a repository for shared knowledge, including the accepted design works and shared symbolic representations, e.g. the domain-specific languages developed to improve communication between agents (Saunders, 2011). The languages include simple design grammars and relevant semantics that reflect design structures and features particularly in the experiments, Incongruity (see Section 5.3) and Extensibility and Other Features (see Section 5.4). Different domains or sub-domains

[1] 70% is sufficient for agents to share a common language while retaining individual differences.

can be subsequently used as initial settings for the production of new generations of designs (Saunders, 2011). The interaction of domains is implemented in the experiment, Clique Formation (see Section 6.3).

Design knowledge is composed of a number of rules such as repetition, variation, contrast, rhythm, self similarity etc. For example, the specific constructing rules of rectilinear volumes mainly include piercing, wedging and cradling. These structures can be represented by utterance combinations such as "co-we-ya" and "fe-ge-ge", which are generated and shared by interactions between individuals, field and domain (Saunders, 2011). And these combinations can be mapped to different designs.

Domain knowledge can be generated in a connectionist model to impose constraints on solutions such as Boolean operations on geometric shapes in the experiment, Compositional Language for Shape Combination (see Section 4.3). The domain knowledge is distributed among individuals rather than being stored in the central repository that in such cases the domain is considered to be that part of the learned meaning of utterances that is shared by a significant proportion of a field (Saunders, 2011).

Designs and languages are shared in a public or local domain. Different clients can have different evaluation criteria. Some clients can even adopt the failed works rejected by other clients that results into local domain and clique formation.

3.1.2 Individual

The types of individuals include adaptive agents and curious agents. The former such as listener-agents evolve basic domain languages in guessing games while the latter including client-agents and designer-agents evolve creative design briefs and works respectively in generation games. Curious agents are capable of evaluating and selecting topics, utterances and designs, and extracting interesting features based on the evaluation of novelty using hedonic functions.

In guessing games, agents can also be curious when selecting interesting topics or generating novel utterances representing these topics. A curious speaker-agent is capable of selecting a novel but not too novel topic by comparing current topics with the topics it has experienced in using hedonic functions. It can also choose a unique topic from current topics by comparing their features such as the differences of shapes, colours or sizes. Thus an interesting topic can be selected according to its novelty and distinctiveness. A novel utterance can also be generated by selecting a different combination of characters compared with the utterances which are stored in its memory, or by combining some of these stored utterances to a new compositional utterance.

In generation games, designer-agents learn basic rules and methods from the domain, then generate new works and modify them to satisfy clients' requirements in the field.

Agents adjust the preference of communication and categorise topics using short-term memory and long-term memory respectively (Saunders, 2002).

The process of communication among multi-agents is generating, exchanging and evaluating messages between addressers and receivers in various contextual environments (Lidov, 1999). Statements are mainly generated by initiator/speaker-agents and client-agents. These statements constitute several features such as ambiguity, scalability and extensibility, which can be used to generate novel works by designer-agents. These works are sent to client-agents for evaluation. The repeated communications among agents lead to semantic assimilation and grow a full-blown artificial language by generating utterances and expanding relevant meanings (Saunders, 2011).

3.1.3 Field

Field is responsible for assessing generated designs, extracting interesting works, and sending them to the Domain. The assessment criteria identify the difference between objects and the relationships among them (Hannah, 2002). The evaluations of generated works obey some common criteria such as similar-yet-different although different individuals make various judgments according to their own experiences and roles. Not only the designs generated by designer-agents but also the utterances generated by speaker-agents and client-agents can be evaluated. For example, "babadula" is more interesting than "babababa" as a design brief. Such accumulated successful works and relevant representations evaluated and extracted from the Field enrich the Domain.

In the computational model of the DIFI framework, individuals learn about the common knowledge of the Domain while retaining their personalities due to the differences in the sequence of experiences that they have, i.e., the situatedness of the agents (Gero and Fujii, 2000). This leads to sustainable creative behaviours by exchanging knowledge and re-generating new knowledge. Emergence occurs via large scale communications from local niches (Saunders, 2002) via the evolution of language in the Field.

3.2 Representation

The representations of topics and works include extracted meanings, linguistic utterances and their relationships. The transformation between meanings and utterances is a process of production and parsing. Firstly, an object such as a square is represented as a vector containing several dimensions such as coordinates and size, e.g. [x:0.1, y:0.3, size: 0.5]. Then the vector is mapped with a compositional utterance, e.g. "ba-du-la". A number of vectors with the same topic, e.g. square, can be generated to form a context for playing language games. A context composed of several features could be represented in multi-dimensions, where each dimension denotes a feature. For example, a 2-dimensional

context is {colour: [0.09 0.56 0.38 0.09 0.99 0.92 0.99 0.93], size: [0.6 0.6 0.59 0.28 0.01 0.84 0.22 0.83]}. The representation of objects is transformed through a mapping process to a representation of utterances. Lastly, utterances are generated to represent meanings in compositional and hierarchical structures. For example, the sequences "$c_i v_i c_i v_i \ldots$" could be generated by combining consonants, "bcdfghjklmnpqrstvwxyz", and vowels, "aeiou", such as"qojila". A new (imagined) object may be generated by recombining existing utterances and transforming this into a representation for the utterance that can be mapped back into a representation.

Thus the relationships between conceptual representations (meanings) and languages (utterances) can be established and developed by many-to-many mapping. Each connection of a meaning and an utterance can be weighted according to the frequency of usage or different context. Based on many-to-many mapping, compositional sequences representing containment or coordinate meanings can be constructed. In this thesis, the associations between holistic/compositional utterances and meanings are represented using weighted associations, Holographic Reduced Representations and Weighted Context Free Grammar.

3.2.1 Association Between Meanings and Utterances

The primary representations utilised in the following experiments are weighted associations between meanings and utterances[2]. A meaning can be represented with a number of weighted utterances while an utterance may be associated with several meanings. For example, square could be associated with three weighted utterances, "ba 0.12", "ki 0.34" and "lu 0.23". Among these, "ki" is much more easily selected to represent square due to its higher weight. The probability of selecting "ki" is 49.3% $\leftarrow$ 0.34 / (0.12 + 0.34 + 0.23) while that of selecting "lu" is 33.3% and that of selecting "ba" is 17.4%. "ki" could also represent other meanings such as triangle (weight: 0.11) and circle (weight: 0.02). Combination can be represented using compositional utterances. For instance, blue square can be represented as "du-ki", which is the combination of "du" representing "blue" and "ki" denoting "square".

The combinations of colours and shapes can be associated with compositional utterances. New meanings may be generated by recombining these utterances. For example, a new utterance, "red triangle" can be generated by crossing over "red square" and "blue triangle". New utterances can also be generated by recombining topics in the same dimension. For instance, the combination of "red" and "green" could represent "reddish green", which is different from "greenish red".

[2] "Meanings" are meaningful words (e.g. "square") directly representing the signified meanings while "utterances" are artificial words (e.g. "ki") generated by computational agents to associate with the "meanings" in the evolution of artificial languages.

General compositional representations are utilised in most of the experiments including Compositional and Holistic Language (see Section 4.1), Compositional Language for Shape Combination (see Section 4.3), Ambiguity (see Section 5.1), Growing Population (see Section 6.1), Education in Guessing Game (see Section 6.2) and Clique Formation (see Section 6.3). Some examples of the representations are described below.

(1) Compositional and Holistic Language (see Fig. 4.3)

Compositional language:
$rule$ = {*size*: {"a": 0.54, "o": 0.03}, *colour*: {"e": 0.12, "i": 0.34}, *shape*: {"u": 0.79, "a": 0.35}, *red*: {"m": 0.63, "f": 0.02},...}
$utterances_{feature}$ = {*size*: "a", *colour*: "i", *shape*: "u"}
$utterances_{attribute}$ = {*small*: "b", *red*: "m", *triangle*: "k"}
small-red-triangle → "bamiku"

Holistic language:
$rule$ = {*small-red-triangle*: {"xs": 0.92, "cj": 0.14}, *medium-yellow-square*: {"gj": 0.02, "rx": 0.67},...}
small-red-triangle → "xs"

(2) Compositional Language for Shape Combination (see Fig. 4.20)

$boolean$ = {*union*: 0.01, *intersection*: 0.02, $difference$: 0.01}
$nodes_{hexagon}$ = {[0,1]: 0.11, [1,2]: 0.03,...}
$nodes_{star}$ = {[10,1]: 0.24, [1,2]: 0.03,...}
$combine(hexagon, star) \rightarrow intersection(hexagon[0,1], star[10,1])$

$rule$ = {*hexagon*: {"c": 0.67, "v": 0.03,...}, *star*: {"g": 0.12, "p": 0.34,...},...}
hexagon-star → "cp"

(3) Ambiguity (see Fig. 5.2, 5.3)

A rule is a dictionary that contains feature, utterances, category and weight, e.g. {feature: "shape", utterance: "a", category: 0, weight: 0.01},

An instance is also a dictionary that contains prototype, utterance and frequency, e.g. {prototype: [0.1,0.3], utterance: "ab", frequency: 1}.

A prototype is a combination of colour and shape, while frequency is the number of times that the same relation between a prototype and utterance is used and accepted.

(4) Growing Population and Education in Guessing Game

A colour is represented with four features including $red, green, blue$ and *alpha*. Each feature is connected with a number of weighted rules associating its categories and utterances. An example of generating a compositional utterance using four selected rules is described as follows.

$rule_{red}$ = {*category*: 1, *utterance*: "ce", *weight*: 0.32}
$rule_{green}$ = {*category*: 0, *utterance*: "yi", *weight*: 0.21}
$rule_{blue}$ = {*category*: 2, *utterance*: "xa", *weight*: 0.57}
$rule_{alpha}$ = {*category*: 1, *utterance*: "re", *weight*: 0.11}
combine(*red*, *green*, *blue*, *alpha*) → "ceyixare"

(5) Clique Formation

A vase is represented with four features including $r1, r2, r3$ and $r4$. Each feature is connected with a number of weighted rules associating its categories and utterances. An example of generating a compositional utterance using four selected rules is described below.

$vase = [r1, r2, r3, r4]$
$rule_{r1}$ = {*category*: 2, *utterance*: "se", *weight*: 0.27}
$rule_{r2}$ = {*category*: 1, *utterance*: "ro", *weight*: 0.14}
$rule_{r3}$ = {*category*: 1, *utterance*: "lu", *weight*: 0.28}
$rule_{r4}$ = {*category*: 0, *utterance*: "ta", *weight*: 0.61}
combine($r1$, $r2$, $r3$, $r4$) → "seroluta"

A building is represented with five features including $length, width, height, scale$ and $twist$. Each feature is also connected with a number of weighted rules associating its different value ranges (i.e., categories) and utterances. An example of generating a compositional utterance using five of these rules is described as follows.

$building = [length, width, height, scale, twist]$ (see Fig. 6.9, 6.11)
$rule_{length}$ = {$range_{value}$: {3: 0.9, 4: 1, 5: 0.9}, *utterance*: "hu", *weight*: 0.72}
$rule_{width}$ = {$range_{value}$: {2: 0.9, 3: 1, 4: 0.9}, *utterance*: "ro", *weight*: 0.84}
$rule_{height}$ = {$range_{value}$: {6: 0.9, 7: 1, 8: 0.9}, *utterance*: "ji", *weight*: 0.45}
$rule_{scale}$ = {$range_{value}$: {1: 0.9, 2: 1, 3: 0.9}, *utterance*: "ru", *weight*: 0.23}
$rule_{twist}$ = {$range_{value}$: {5: 0.9, 6: 1, 7: 0.9}, *utterance*: "si", *weight*: 0.49}
combine(*length*, *width*, *height*, *scale*, *twist*) → "hurojirusi"

3.2.2 Compression via Holographic Reduced Representations

Compositional representations such as the representations of compositional geometric meanings can be generated using Holographic Reduced Representations (HRRs) (see Fig. 3.2) compressing several basic representations with the same length of sequence (Plate, 2003)[3]. The differences between the representations generated by HRRs can be easily measured using Cosine Similarity. Two representations are more different when the cosine of the angle between the two vector-representations is smaller. HRRs are

[3]The capability of representing and differentiating different topics by Holographic Reduced Representations is based on the generated vectors which are different from each other. Each one contains a great number (e.g. 1024) of randomly generated numbers in range [0,1].

capable of associating relative concepts, organising the clusters of ideas and concepts, and using memory efficiently (Plate, 2003). These capabilities are useful for representing compositional meanings such as geometric relations. For example, a square A containing another square B $(V_A, V_{contain}, V_B)$ can be represented as $V_{contain(A,B)}$ with the same length of each vector using circular convolution that saves memory. At the same time, the relation $V_{contain(A,B)}$ can be distinguished from other relations such as $V_{touch(A,B)}$ and $V_{intersect(A,B)}$ easily. Their differences can be measured using cosine similarity. Thus HRRs help evolving hierarchical compositional representations efficiently.

A language can evolve to a complex hierarchical network via composition and decomposition within Holographic Reduced Representations (HRRs). High-level combination can be produced by combining several words to generate innovative concepts. Complex design concepts can also be decomposed into embodied elements by tracking their associations and categories. Both of these are realised using circular convolution ($t = c \otimes x$, see Equation 1) composing items, and circular correlation ($y = c \oslash t$, see Equation 3) decoding convolution (Plate, 1995).

3.2.2.1 Circular Convolution and Correlation

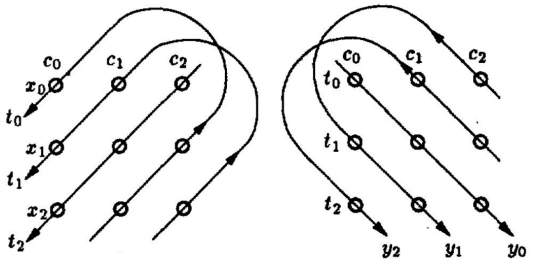


FIGURE 3.2: Circular convolution and circular correlation (Plate, 1995)

HRRs use circular convolution to associate items represented with vectors. The representation of an association is a vector with the same dimensions as the associated vectors. Arbitrary variable bindings, short sequences of various lengths, simple frame-like structures and reduced representations can be represented in a vector with fixed dimensions. It allows the construction of representations of objects with compositional structure. For example, associations between the vectors, "red" and "triangle", can be combined via circular convolution (Plate, 1995).

$$t_j = \sum_{k=0}^{n-1} c_k x_{j-k} \tag{1}$$

for j=0 to (n-1) (Subscripts are modulo-n)

$$\begin{bmatrix} t_0 = c_0x_0 + c_2x_1 + c_1x_2 \\ t_1 = c_1x_0 + c_0x_1 + c_2x_2 \\ t_2 = c_2x_0 + c_1x_1 + c_0x_2 \end{bmatrix} \tag{2}$$

The result of circular convolution, vector $\boldsymbol{t}$, can be obtained by operating vector $\boldsymbol{c}$ and vector $\boldsymbol{x}$ using Equation 1. In this equation, t_j is the j^{th} item of $\boldsymbol{t}$, c_k is the k^{th} item of $\boldsymbol{c}$, and x_{j-k} is the $(j-k)^{th}$ item of $\boldsymbol{x}$. The three vectors $\boldsymbol{t}$, $\boldsymbol{c}$ and $\boldsymbol{x}$ have the same dimensions, n. If $n = 3$, the result can be expressed as Equation 2, in which $\boldsymbol{t} = [t_0, t_1, t_2]$.

$$y_j = \sum_{k=0}^{n-1} c_k t_{j+k} \tag{3}$$

for j=0 to (n-1) (Subscripts are modulo-n)

$$\begin{bmatrix} y_0 = c_0t_0 + c_1t_1 + c_2t_2 \\ y_1 = c_2t_0 + c_0t_1 + c_1t_2 \\ y_2 = c_1t_0 + c_2t_1 + c_0t_2 \end{bmatrix} \tag{4}$$

The result of circular correlation, vector $\boldsymbol{y}$, can be obtained by operating vector $\boldsymbol{c}$ and vector $\boldsymbol{t}$ using Equation 3. In this equation, y_j is the j^{th} item of $\boldsymbol{y}$, c_k is the k^{th} item of $\boldsymbol{c}$, and t_{j+k} is the $(j+k)^{th}$ item of $\boldsymbol{t}$. The three vectors $\boldsymbol{y}$, $\boldsymbol{c}$ and $\boldsymbol{t}$ have the same dimensions, n. If $n = 3$, the result can be expressed as Equation 4, in which $\boldsymbol{y} = [y_0, y_1, y_2]$.

Algorithm 1 Circular Convolution and Circular Correlation

```
1:  function convolveCorrelate(x, y, type)
2:      n ← length(x)
3:      result ← emptyList
4:      for j = 0 → (n − 1) do
5:          item ← 0
6:          for k = 0 → (n − 1) do
7:              if type =′ convolve then
8:                  item ← item + x[k] × y[mod((j − k + n), n)]
9:              else if type =′ correlate then
10:                 item ← item + x[k] × y[mod((j + k), n)]
11:             end if
12:         end for
13:         result ← append(result, item)
14:     end for
15:     return result
16: end function
```

Circular convolution (see Equation 1 and 2) and circular correlation (see Equation 3 and 4) can be implemented using Algorithm 1. In both of them, the elements are summed

along the indicated transdiagonals (see Fig. 3.2). circular correlation is an approximate inverse operation of circular convolution. If a pair of vectors ($\boldsymbol{c}$, $\boldsymbol{x}$) is convolved together to give a memory trace $\boldsymbol{t}$, then one member of the pair, e.g. $\boldsymbol{c}$, can be correlated with the trace $\boldsymbol{t}$ to get the other member, e.g. $\boldsymbol{y} \approx \boldsymbol{x}$, of the pair (Plate, 1995).

The alternative of circular convolution is addition, which is simple and functional (Plate, 1995). But addition memory may produce ambiguity, particularly for multiple objects. Such ambiguity should be avoided, especially when a vector representing an object needs to be extracted from the compositional vector representing multiple objects. Addition memory, however, can be used to build up the associations between pair-vectors since adding together two high dimensional vectors gives a vector in which each is similar to the other, that can be used to distinguish the pair-vectors from other unrelated concepts.

A desirable feature of HRRs employed here is their ability to encode recursive structures of arbitrary complexity. Recursive representations such as "Hunger caused Mark to eat the fish" can be realised via HRRs, which are described below (Plate, 1995).

$S_1 = eat + eat_{agt} \otimes Mark + eat_{obj} \otimes fish$
$S_2 = cause + cause_{agt} \otimes hunger + cause_{obj} \otimes S_1$
$S_2 = cause + cause_{agt} \otimes hunger + cause_{obj} \otimes eat + cause_{obj} \otimes eat_{agt} \otimes Mark + cause_{obj} \otimes eat_{obj} \otimes fish$

Hence, hierarchical meanings can be represented and clarified by HRRs. For example, "square inside triangle" can be distinguished from "square inside circle" via HRRs: $(inside + inside_{s1} \otimes square + inside_{s2} \otimes triangle)$ is different from $(inside + inside_{s1} \otimes square + inside_{s2} \otimes circle)$. Topological relations can also be represented as $R(a, b, c) = P_1 \otimes F_a + P_2 \otimes F_b + P_3 \otimes F_c$. Here, R denotes relation, P denotes relative position and F is the features of an object.

3.2.2.2 The Evaluation of HRRs

Cosine similarity can be used to evaluate two vectors represented by HRRs with the same dimensions. The two vectors are the same when the result of cosine similarity is one ($\cos(0) = 1$). If it is less than one, they are different. When the result becomes smaller, the difference between them becomes greater. Therefore the cosine of the angle between two vectors can be used to measure the similarity between geometric relations represented via HRRs (Pang-Ning et al., 2006).

3.2.2.3 Interleaving Representations

Circular convolution might not be capable of distinguishing accurate meanings such as the distance among topological relations "in", "touched-in", "intersect", "on" and

"to" (see Fig. 3.3). Alternatively, interleaving technique, i.e., values of taken from the two vectors to be combined with equal probability, may be used to do "accurate" representations.

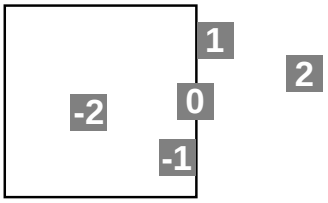


FIGURE 3.3: Topological relations (0: intersect, -1: touched-in, -2: in, 1: on, 2: to)

As can be seen from Fig. 3.3, the relation "in"(-2) is far away from "to"(2), whereas "touched-in"(-1) is close to "on"(1), and they overlap "intersect"(0). The HRR representations of these relations can start from "intersect", and recursively operate on it with "inward" to "touched-in" and "in", and with "outward" to "on" and "to" (see Table 3.1). Three operators including circular convolution, addition and interleaving are used respectively to represent the relations. Then the cosine distances[4] between the relations are measured (see Table 3.2).

TABLE 3.1: The rules of representing topological relations

No.	Rule		Expansion
0	operator	→	convolve / add / interleave
1	inward	→	(normal-randomize 1024)
2	outward	→	(normal-randomize 1024)
3	intersect	→	(normal-randomize 1024)
4	touched-in	→	(operator inward intersect)
5	in	→	(operator inward touched-in)
6	on	→	(operator outward intersect)
7	to	→	(operator outward on)

TABLE 3.2: Cosine distances between topological relations

Cosine-distance	Convolve	Add	Interleave
$D_{Cosine}(in, touched-in)$	1.030	0.051	0.240
$D_{Cosine}(in, intersect)$	1.016	0.548	0.738
$D_{Cosine}(in, on)$	0.998	0.671	0.842
$D_{Cosine}(in, to)$	1.032	0.787	0.902
$D_{Cosine}(on, to)$	1.027	0.052	0.232

The results of cosine distances (see Table 3.2) show that similar topological relations are represented by more similar vectors generated using interleave technique[5] than circular convolution. For example, the distance between "in" and "touched-in" is smaller than that between "in" and "intersect", and the distance between "in" and "intersect" is

[4]Cosine distance is opposite to Cosine similarity. The vectors are more similar to each other when their Cosine distance is smaller, whereas they are more different when their Cosine similarity is smaller.

[5]The mechanism of interleaving makes the results from the same data a little different each time but in an acceptable range.

smaller than that between "in" and "on". The differences between these distances are matched very well by using interleaving technique instead of circular convolution. the results of addition also matches the differences between the relations, but the distribution of the results is more uneven than that generated by interleaving technique.

HRRs are utilised in the experiment Compositional Representation of Rectilinear Relation (see Section 4.2) to test the possibility of representing compositional geometric meanings. An example of the representation is described below.

Two rectangles share a corner (see Equation 14 and 15):
$ec_1c_2 = (edge + shape_1 \otimes corner + shape_2 \otimes corner) + axis(0)$
$utterance[ec_1c_2] = char_1[m] \otimes char_2[0] + char_1[n] \otimes char_2[1] + char_3[u] + char_4[v] \rightarrow$ "fabeji"

3.2.3 Expansion via Weighted Context Free Grammar

Grammars have a long history of use in computational studies of design, e.g., both design grammars and shape grammars (Stiny et al., 1980). A context-free grammar (CFG) (Sipser, 1997), which is a set of recursive rewriting rules, i.e., productions, can be utilised to generate strings or patterns representing designs. A CFG contains 4 components (see Equation 5). They are a set of nonterminal symbols V, a set of terminal symbols Σ, a set of production rules R and a start symbol S (Sipser, 1997). Interesting representations of designs could be generated using CFG. For example, "house on boat in sky" can be generated using the following rules: S $\rightarrow$ (N PN), PN $\rightarrow$ (P N), N $\rightarrow$ [S, "house", "boat", "sky"], P $\rightarrow$ ["on", "in"]. The process is S $\rightarrow$ (N PN) $\rightarrow$ (N (P N)) $\rightarrow$ (S (P N)) $\rightarrow$ ((N PN) (P N)) $\rightarrow$ ((N (P N)) (P N)) $\rightarrow$ (N P N P N) $\rightarrow$ "house on boat in sky".

$$G = (V, \Sigma, R, S) \tag{5}$$

Weighted Context Free Grammar (WCFG) is similar to Probabilistic Context Free Grammar (Charniak, 1997). Weight rather than probability is used to denote the influence of the expansion compared with other expansions of the same rule. Probabilities always have to sum to 1.0 within a grammar, whereas weights do not have this limitation. The expansion with higher weight has more opportunities to be selected during the process of tracing forward its rule. The weight can be strengthened or weakened after each successful or failed language game. Utterances and their combinations are saved in individuals' memory. They use these symbols to expand meanings (expanding the meanings of the words considering different weights or even based on different contexts). The speed of expansion could be exponentially fast.

TABLE 3.3: A sample of context free grammar

No.	Rule		Expansion
0	S	→	(NP VP)
1	NP	→	(D A N)
2	VP	→	(V PP)
3	PP	→	(P NP)
4	D	→	a / an / the
6	A	→	red / yellow / green / blue
10	N	→	fish / bird / duck / cat / dog / sky / sea / land
18	V	→	fly / swim / walk / jump
22	P	→	in / on / over

Some generated phrases using Context Free Grammar (CFG) (see Table 3.3) could be meaningful, e.g. "a yellow duck swims on the sea", but most look very unusual, e.g. "a cat walks in a big fish", strange, e.g. "a sky flies over a bird", or even meaningless. In regular use of WCFG, the weight of normal composition is very high, whereas that of an unusual composition is very low and that of a strange composition[6] is extremely low or even zero. Interesting and creative results rather than normal compositions may be obtained by selecting the compositions with low weight.

Elaborated representations and tree-like structures can be developed using CFG with S-expressions. Hierarchical compositional structure can be elaborated by tracing-forward or simplified by tracing-backward. Tracing-forward (TF) is the process of expanding rules to a complicated nested chain using CFG started from a symbol, which does not have to be a root. The process of generating "house on boat in sky" is tracing-forward. Tracing-backward (TB) is the opposite process of tracing-forward, e.g. "house on boat in sky" → (N P N P N) → ... → S. By combining tracing-backward and tracing-forward, some new compositions based on "house on boat in sky" could be generated. For example, "house on boat in sky" → "house" P N "in sky" → "house" P S "in sky" → "house" P (N PN) "in sky" → "house" P (N (P N)) "in sky" → "house" P N P N "in sky" → "house in boat in house in sky". In this example, the first step is tracing-backward (i.e., $depth_{TB} = 1$) while the remaining steps are tracing-forward.

Novel compositions could be generated if inverse weighted random choice is used to select the candidates on some tracing-steps. For example, firstly, a number of weighted compositions are generated using WCFG based on experience of the real world, such as "bird in sky", "cloud in sky", "fish in sea" and "cat in house on land". Then in the process of tracing-backward & forward of "bird in sky" → N "in sky", "fish" has more chance than "cloud" to be selected due to the low weight of association between "fish"

[6] A simple composition may denote a relation between two objects. An unusual composition would occur when changing object(s); and a strange composition might be generated when changing relation. For example, given $on(roof, wall)$, an unusual composition $on(roof, tree)$ is generated by replacing $wall$ with $tree$ while a strange composition $under(roof, wall)$ is produced by replacing on with $under$. But the meanings of the two types of composition could be overlapping. For example, given $in(fish, sea)$, the meaning of unusual composition $in(fish, sky)$ is similar as that of strange composition $above(fish, sea)$.

and "sky" if inverse weighted random choice is used. So, a novel composition, "fish in sky" could be generated. This strategy may be used to generate novel designs which could be unusual, strange or incongruent.

The extension of context free grammar is used to generate compositional rules which associate labels and utterances. Labels are also associated with different categories or prototypes or features. A nested dictionary, including functions, can be embedded in CFG. The following example is an expanded compositional structure. The original door is "(rectangle up large)" while the elaborated door could be "((in up) ((intersect left) (triangle right small) (circle up medium)) (rectangle up large))" by using three rules—"object → (relation object object)", "relation → (topological-relation direction)", and "object → (shape direction size)".

An example of using a weighted context free grammar (WCFG) in language games for creative designing is described as follows. A few operators consisting of $<, =, >$ $, -, +, and, or$ can be used to generate s-expressions representing compositional features of a scene-sample. For example, if $(> \ y \ h)$ represents "sky", where y is the vertical position of a point in a scene and h is the position of the horizon; then $(> \ (- \ y_c \ r_c) \ h)$ may represent "circle in the sky", where r_c is the radius of the circle, which could be simplified to "circle sky" by connecting two nodes "circle" and "sky" with a graph-edge representing "in". In addition, some more complicated compositional utterances may evolve such as "large square across large sky and small sea" related to the equation $(and \ (large \ r_s) \ (< \ (- \ y_s \ r_s) \ (small \ h)) \ (< \ (small \ h) \ (+ \ y_s \ r_s)))$. In this equation, r_s is the radius of the inscribed circle of "square", y_s is the y-coordinate of "square", h is the height of "ground". Such expressions represent various instances. Some instances could occur more frequently than others. For example, the weight, $W(Circle \mid Sky)$ would be higher than $W(Triangle \mid Sky)$ after a number of guessing games. These results of weights of instances can be stored in each agent's memory. Then, agents can evaluate whether a new instance is incongruent or exaggerated in new guessing games, or whether a new generated compositional utterance such as "square sea" is unusual in new generation games.

WCFG is utilised in the experiments, Scalability (see Section 5.2), Incongruity (see Section 5.3) and Extensibility and Other Features (see Section 5.4) to explore the creative features of language and apply them in compositional conceptual design. Some examples of the representations are described below.

(1) Scalability

Scalability 1 (see Fig. 5.11, 5.12)

$size \in [0.25, 0.75]$
$rule_{size0.257} = \{prototype$: 0.257, $tolerance$: 0.15, $utterance$: "za", $weight$: 0.995$\}$

$size_{exaggerated} \in (0, 1]$
$size_{exaggerated} \rightarrow [size,\ very] \rightarrow [0.257,\ very] \rightarrow$ [“za”, “vo”] $\rightarrow$ “zavo”

Scalability 2 (see Fig. 5.13, 5.14)

$size \in (0, 1]$
$rule_{small} = \{size\text{: } small,\ utterance\text{: } \{\text{“bo”: } 0.01,\ \text{“sa”: } 0.28,\ \text{“do”: } 0.92\}\}$
$size_{exaggerated} \rightarrow [very,\ small] \rightarrow$ [“ya”,“do”] $\rightarrow$ “yado”

(2) Incongruity

$circle = \{x\text{: } 0.31,\ y\text{: } 0.72,\ r\text{: } 0.13\}$
$square = \{x\text{: } 0.48,\ y\text{: } 0.45,\ r\text{: } 0.11\}$
$triangle = \{x\text{: } 0.75,\ y\text{: } 0.24,\ r\text{: } 0.12\}$
$height_{land} = 0.46$
$rule = \{relation\text{: } (< (- y\, h)\, r),\ shape\text{: } \{circle\text{: } 0,\ square\text{: } 0.51,\ triangle\text{: } 0.83\}\}$

weighted-random-choice$(rule) \rightarrow triangle$
$[triangle,\ (< (- y\, h)\, r)]$
$\rightarrow \{triangle$: “rena”, (: “do”, <: “mico”, −: “ke”, y: “la”, h: “ni”, r: “guce”,): “va”$\}$
$\rightarrow$ “rena do mico do ke la ni va guce va”

$[incongruent,\ (< (- y\, h)\, r)]$
$\rightarrow \{incongruent$: “gi”, $(< (- y\, h)\, r)$: “do mico do ke la ni va guce va”$\}$
$\rightarrow$ “gi do mico do ke la ni va guce va”
inverse-weighted-random-choice$(rule) \rightarrow circle$

(3) Extensibility and Other Features (see Table 5.18 and Fig. 5.34)

$rule_{(house,\, land)} = \{relation\text{: } (on\ house\ land),\ utterance$: “jite qiku wuse wowi gigi”$\}$
$rule_{house} = \{relation\text{: } (on\ roof\ (contain\ wall\ (window\ door))),\ utterance$: “jite qiku su jite ka xuko jite ma geha gigi gigi gigi”$\}$
$rule_{roof} = \{features\text{: } \{shape\text{: } triangle,\ direction\text{: } up,\ size\text{: } medium\ \},\ utterance$: “su”, $weight$: $0.01\}$

$elaboration$: $(triangle\ up\ medium)$
$\rightarrow ((on\ (right\ up))\ (triangle\ up\ medium)\ (triangle\ left\ medium))$ (see Table 5.19)
$unusualness$: $(in\ fish\ sea) \rightarrow (in\ fish\ sky)$ (see Table 5.20)
$strangeness$: $((on\ up)\ roof\ wall) \rightarrow ((in\ up)\ roof\ wall)$ (see Table 5.21)
$exaggeration$: $(triangle\ up\ medium) \rightarrow (triangle\ up\ (very\ (very\ large)))$ (see Table 5.22)
$incongruence$: $(intersect\ boat\ sea) \rightarrow (intersect\ tree\ sea)$ (see Table 5.23)

3.3 Communication

The communication in this thesis is the process of evolving languages and generating interesting design works among agents playing different roles, such as speaker and listener in guessing games, and client and designer in generation games. In each language game, the agents learn successful relations between topics and utterances using Equation 6 (initial weight: $w_0 = 0.01$, learning rate: $\eta = 0.1$) to strengthen their weights, and unlearn failed relations using Equation 7 to weaken their weights.

$$w \leftarrow \eta + (1 - \eta) \times w \tag{6}$$

$$w \leftarrow (1 - \eta) \times w \tag{7}$$

In each language game, weighted random choice (i.e., weighted choice using roulette wheel selection $p(x) = w_x / \sum_{i=1}^{n} w_i$) of utterances or designs is typically better than either "best" choice (i.e., choosing the item with the highest weight) or random choice (i.e., equal probability of choosing any item). Compared with "best" choice, weighted random choice enables an item with low weight still having a small chance of being selected; compared with random choice, it provides more reasonable results.

Whether to end or continue language games is determined by success rate (e.g. when $rate_{success} > 0.7$, the game will stop and new type of game will start) and the number of games being played (e.g. $n_{min} = 100$, $n_{max} = 1000$, $n_{min} <= n <= n_{max}$. Setting n_{min} is to avoid premature success). The success rate is calculated using Equation 8.

$$r_s = n_s / n \tag{8}$$

r_s is the current success rate, n_s is the number of successful games while n is the total number of games played by agents.

The processes of communication involve serial processes and parallel processes. In serial processes, artifacts are created and brought into a social venue so that others can build on them. During parallel processes, components are produced separately then brought together and combined into one new product. Something new could be jointly created by multi-agents simultaneously (Fischer et al., 2005). Analogically, a complicated utterance could be generated by the cooperation of multi-agents obeying a chain rule, a star-like rule or a mixture of both. In the chain rule, one agent initiates a small utterance, then other agents one by one adjust or extend the utterance changed by the previous agent. In the star-like rule, each agent generates its own part of the utterance and all of these would then be combined in various ways.

With the development of artificial languages for creative design, two types of language games are conducted: guessing games between speaker-agents and listener-agents to

evolve grounded language; and generation games between client-agents and designer-agents seeking to generate creative designs in the evolution of compositional languages.

3.3.1 Communication in Guessing Games

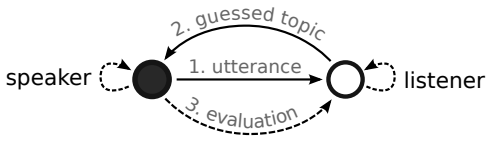


FIGURE 3.4: Guessing game (after Saunders and Grace (2008))

Normally, two individuals play the roles of speaker and listener respectively when they communicate in guessing games (see Fig. 3.4) to evolve a grounded language based on context and previous simple utterances initially generated via random selection[7]. They would develop and share a domain language representing certain topics such as colour and shape by continuing to name what they see, "talk about" what they get and learn from each other. The general process of a guessing game is described as follows.

Firstly, the speaker-agent A_s selects a topic T_s from the context C_t, which contains a number of topics such as shapes and colours, and describes the topic T_s with an utterance U_s. Then this utterance is sent to the listener-agent A_l who parses it to a guessed topic T_l based on the context C_t. When speaker A_s receives listener's guessed topic T_l, it compares its own topic T_s with the listener's topic T_l. If the guessed topic is accepted by the speaker according to its criteria such as the same or in the same category, both speaker and listener learn (see Equation 6) their own representations of the association between topic (instance) and descriptions, including the extracted meaning and generated utterance (A_s: learn(association T_s U_s), A_l: learn(association T_l U_s)). Otherwise, the listener unlearns (see Equation 7) its representation (A_l: unlearn(association T_l U_s)) and learns the association between the speaker's selected topic with related meaning and utterance (A_l: learn(association T_s U_s)).

3.3.2 Communication in Generation Games

Language can be continually developed and applied in generation games played by a client-agent and several designer-agents (see Fig. 3.5) based on the grounded language evolved in guessing games. In each generation game, the client-agent A_c generates a new utterance U_c by crossing over, mutating or elaborating existing utterances according

[7] In early stage of guessing games, most utterances are generated randomly to represent selected topics due to agents' initial empty memories. Once one or more associations between utterances and topics are memorised by participant-agents in the previous guessing games, they will try to find relevant utterances from these memorised associations in next guessing games. If relevant utterances can not be found, they will still be generated randomly. The mappings of utterances to topics are thus transformed from random connections to relevant associations gradually according to agents' accumulated experiences during guessing games.

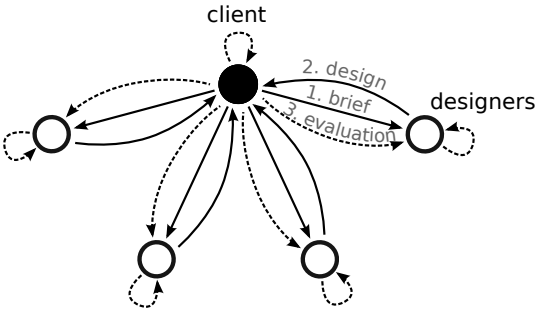


FIGURE 3.5: Generation game (after Saunders and Grace (2008))

to the frequency of current utterances or the weights of various associations between prototypes P_c, meanings M_c and utterances U_c; in other words, based on its memorised associations and the criteria of evaluating the interestingness of utterances or prototypes. When the designer-agent A_d receives the client's requirement, i.e., the new generated utterance U_c, it generates a new design D_d by parsing the utterance to relevant meanings M_d which could relate to certain prototypes P_d in terms of its own criteria of evaluating novelty and utility based on previous experience. After all designs, which are likely to be different from each other due to designers' different experiences, are generated and submitted to the client-agent, the client-agent evaluates each design according to its criteria of creativity related to hedonic functions or certain definitions of specific interesting concepts such as incongruity and exaggeration. If a designer-agent's work is accepted, both the client-agent and the designer-agent learn the new association between the design-works and requirement (A_c: learn(association D_d U_c), A_d: learn(association D_d U_c)). Otherwise, the designer-agent would unlearn its representation of the design (A_d: unlearn(association D_d U_c)) and try to learn some other designer-agents' works, in particular the best design selected by the client-agent (A_d: learn(association D_{winner} U_c)) (Saunders, 2011).

The relationship between client-agents and designer-agents, and the mechanics of their communication can be investigated to improve the efficiency of transforming clients' description language, i.e., requirements, to design concepts. During the repeated generation games, some complex linguistic combinations may emerge to support complicated design briefs encouraging creative meanings.

3.4 Evaluation

The process of evaluation is dynamic insofar as the evaluation of creativity is relative to the continuous change of individuals, context and the interactions among them. For instance, the novelty of artworks based on changing experiences is explored and measured continuously via the Habituating Self-Organising Map (HSOM) (Marsland et al., 2000). Through evaluation, individuals should learn to capitalise on their strengths at the same

time as they compensate for their weaknesses in whatever ways are available (Sternberg, 1985). To make dynamic and flexible evaluation, variations can be used in the criteria to define creativity and judge the output items (Ritchie, 2001).

A number of features including novelty, appropriateness, influence, diversity and efficiency will be evaluated respectively in two aspects, individual and society. Some features could be more important than others due to different types of simulation and different stages of the same simulation. These features or evaluation guidelines can be tested by comparing the evaluation results with those from human beings (Jordanous, 2013) or vice versa. The evaluations are completed by both curious agents themselves and by systemic evaluation.

3.4.1 Novelty

Novelty is the fundamental feature for evaluating creativity. Individual novelty relates to the artifacts and utterances generated by each agent while societal novelty includes social structures and the ratio of novel artifacts/requirements over normal products/utterances generated by all participants.

Individual novelty is the novelty of artifacts, concepts and utterances, i.e., designs and requirements, which can be evaluated using linear or nonlinear hedonic functions transforming novelty into interest. A novelty detector is used to detect novelty, e.g., a neural network like a SOM. For example, the Euclidean distances between designs and original prototypes are measured, then these distances are evaluated using hedonic functions to get the value of interest based on novelty. The design with the highest score on the evaluation will be selected.

Societal novelty contains novel social structures and novel social processes. The former can be new types of networks, new types (roles) of agents, new relationships between these agents, the communication density between agents, and the density of creative agents, designs and utterances. Among them, the density of interesting utterances can be evaluated by the frequency of the utterances, the similarity between the utterances, and the frequency of characters in all utterances collected from all participants. The latter can be new types of communication, new mechanisms of evaluation, and new chains of creative events in group etc.

3.4.2 Appropriateness

Appropriateness is an essential feature of creativity. Suitable representations can make new concepts really become creative. And creative social interactions depend on not only their novelty but also their constructiveness.

Relevant associations between utterances and topics/designs can be evaluated via the degree of discrimination and that of consistency. The former refers to how much the topics are distinguished from each other by the evolved languages while the later refers to the extent to which agents share the same language (see Section 4.1).

Societal appropriateness denotes appropriate ratio of agents with different roles, rational methods of communication in different stages, and suitable communication strategies etc. For example, the appropriateness of education game is evaluated by comparing different ratio of teacher-agents and student-agents (see Section 6.2). The appropriate process of evolving creative artificial language is from guessing games to generation games (see Section 5.1). And a strategy of selecting suitable participants can be based on previous communication success rates (see Section 6.3).

3.4.3 Influence

Influence is the impact of an individual's or a group's behaviours on others' behaviours through communication and exhibition using generated utterances or designs. Great influence is related with high transmitting speed, large scale of affection, and maintaining the affection for a long period. It can be evaluated at individual level and at societal level respectively.

Individual influence is the impact of an agent on other agents through its generated utterance-like requirements or designs which are accepted by other agents. For example, a client-agent's influence can be evaluated by calculating the number of its utterances accepted by designer-agents. A designer-agent's influence can be evaluated by calculating the number of its designs accepted by client agents and other designer-agents. An individual's influence may also be measured according to the number of its neighbours of a graph network and their successful communications.

Societal influence is the whole strength of spreading utterances and designs through networks. It depends on the structure of networks developed by agents communicating with each other. It can be evaluated by calculating the density of the structure, the level of information flow, and the inheritance of knowledge from one generation to another generation.

3.4.4 Diversity

Diversity could open up more opportunities for creativity (Gassmann, 2001). An individual could produce more abundant products with more potential of generating novel and interesting artifacts if it was experienced with diverse prototypes. Similarly, a community or society could be more creative if it was composed of diverse and complex individuals and groups.

Individual diversity is measured with the ratio of an individual's stored meanings to all agents' stored meanings (see Equation 9), that of an individual's stored utterances to all agents' stored utterances (see Equation 10), and that of an individual's stored connections of meanings and utterances to all agents' stored connections of meanings and utterances (see Equation 11). Individual diversity is also influenced by the size of the neighbourhood and the differences among the neighbours (Liu et al., 2010).

$$r_{mi} = n_{mi}/N_m \tag{9}$$

$$r_{ui} = n_{ui}/N_u \tag{10}$$

$$r_{ci} = n_{ci}/N_c \tag{11}$$

r_{mi} denotes Agent i's individual diversity of stored meanings, n_{mi} is the number of Agent i's stored meanings while N_m is the total number of all agents' stored meanings. r_{ui} denotes Agent i's individual diversity of stored utterances, n_{ui} is the number of Agent i's stored utterances while N_u is the total number of all agents' stored utterances. r_{ci} denotes Agent i's individual diversity of stored connections of meanings and utterances (CMU), n_{ci} is the number of Agent i's stored CMU while N_c is the total number of all agents' stored CMU.

Societal diversity can be measured by social division, that is, the degree of variation among individuals and groups. Diversity may not be applicable to the comparison of the creativity of holistic and compositional languages because composition biases complexity. It could, however, be used to confirm that composition produces greater diversity than holism. Diversity is not an essential feature of creativity. However, the more diverse the space is, the less predictable are its contents and their locations, and the more value observers are likely to place on them when they are found (Wiggins, 2006a).

Simpson (1949) 's Diversity Index (see Equation 12, 13), which takes into account richness and evenness, can be utilised to calculate both individual and societal diversity. For individuals, this comprises the number of instances related to each category and the total number of instances, the number of each utterance and the total number of utterances in instances, and the number of each prototype (meaning) and the total number of prototypes (meanings) in instances. For society, this comprises the number of each type of agent (e.g. clients/designers, teachers/students and naive/mature agents) and the total number of agents.

$$I_{SD} = 1 - \frac{\sum_{i=1}^{k} n_i \times (n_i - 1)}{N \times (N - 1)} \tag{12}$$

$$N = \sum_{i=1}^{k} n_i \tag{13}$$

I_{SD} denotes Simpson Diversity Index, which ranges between 0 and 1. The diversity is greater when the value is higher. n_i is the number of Object i while N is the total number of all objects. k is the number of object-types.

The diversity of utterances and representations (Bird et al., 2009) and entropy (Kan and Gero, 2009) can also be evaluated.

3.4.5 Efficiency

Efficiency and fluency are not necessary for measuring P-creativity. But if the same creative result was generated by two agents, and agent A spent less time or used fewer resources or a more concise method than agent B, agent A is more likely to realise H-creativity.

Individual efficiency refers to the speed with which an agent generates rules associating utterances with topics, and instances (i.e., the shortest time spent generating interesting concepts and the degree of interestingness).

Societal efficiency can be evaluated by measuring the speed of social development such as the speed of accumulating and updating domain knowledge as well as its transmission speed, e.g. fluent communication among individuals. Societal efficiency may be affected by the rules of communication between agents, and different structures of organisations such as hierarchical, flat or mixed structures (see Section 6.1). In terms of the DIFI framework, societal efficiency depends on the speed and span of transforming and transmitting information among individuals, filed and domain.

3.4.6 Conclusion

Computational creativity can be evaluated by measuring the novelty, appropriateness, influence, diversity and efficiency of both results and processes at individual and social levels. Suitable novelty, i.e., the combination of novelty and appropriateness, is the primary criterion for evaluating creative design briefs and design works.

In this thesis, hedonic functions related to the Wundt curve are used to evaluate similar-yet-different works in the experiments, Ambiguity (see Section 5.1) and Clique Formation (see Section 6.3). Cosine similarity is used to measure the differences between compositional topological relations of two rectangles to evaluate the utility of HRRs in the experiment, Compositional Representation of Rectilinear Relation (see Section

4.2). Inverse weighted random choice is utilised to select interesting incongruent artworks in the experiment, Incongruity (see Section 5.3). The diversity of artworks is evaluated by calculating the number of collected shapes and their types with different selecting-thresholds in the experiment, Compositional Language for Shape Combination (see Section 4.3). Social creativity is evaluated by analysing gaming times in the experiments, Compositional and Holistic Language (see Section 4.1) and Education in Guessing Game (see Section 6.2); and by measuring discrimination, consistency, and density of utterances in the experiment, Compositional and Holistic Language (see Section 4.1), as well as the average max degree of graph networks, i.e., the average of all agents' greatest numbers of connections of a node (meaning or utterance) to other nodes, in the experiment, Ambiguity (see Section 5.1). Cultural creativity is evaluated using a distance map to measure the hierarchical relations of agents forming cliques in the experiment, Clique Formation (see Section 6.3).

3.5 Multi-Agent Simulations

Multi-agent simulations consist of multiple software programs, called agents, that interact with each other and environment (Wooldridge, 2002). Multi-agent systems can be used to solve problems that are difficult for an individual agent or a monolithic system to solve. Most of the following experiments use multi-agent simulations with the agents using neural networks to engage in language games. In a multi-agent environment, each individual agent follows a set of unique rules for the generation and evaluation of works (designs/artworks) while the shared domain rules are contributed by all agents, in particular expert agents (Wiggins, 2006a). The domain rules can also be distributed to each agent. The result of each game is evaluated by the participating agents to inform how their rules of generation and evaluation may be adjusted and changed.

Multi-agent simulations can be used to test the results of different strategies of choosing various probabilities by different agents who may be highly desirous of extremely novel compositions or may only require a slightly different composition based on its experiences and goals. Unusual compositions may be related with imagination, which is thinking about concepts that are not present in the context, e.g. the context provided in the guessing game, but associated with the context.

3.5.1 Self-Organisation and Collective Intelligence

Multi-agent simulations are based on self-organisation through which emerging complex behaviours that represent more intelligence than that of the predefined behaviours (Floreano and Mattiussi, 2008). To make a multi-agent simulation successful, the evolution of a complex system requires some constraints. Initially, several simple but necessary

constraints are important for coordinating agents' behaviours, since these constraints can both inhibit and propel different activities which support a succession of recursive responses to solve problems. Guidance from these constraints would lead to the maximisation of rewards for some structures. When the basic structure is established, the regularly accumulated historical and cultural resources would take the main role in the evolution of a complex system such as language (Clark, 1997). Given the interaction with environments, open-ended artificial evolution is similar to reinforcement of neural networks to exploit existing resources and explore new space. The evolution of language is affected by both the communication between individuals and the interaction with external environments (Clark, 1997).

As a type of social intelligence, collective intelligence has several advantages resulting from the dynamic interactions through bottom-up and top-down processes. Each agent gradually receives the whole map of information via communication. In other words, agents might obtain an overall global perspective by sharing their local views. Hence, not only is the precision of the overall solution increased by sharing results, but results could also be received more quickly by sharing solutions, and new tasks can be completed more efficiently. As a kind of collective intelligence, partial global planning has three aspects: (1) each agent has its own goals and short term plans; (2) goals and plans interact via information exchange; (3) the activities of agents could be better coordinated by altering local plans (Wooldridge, 2002). Multi-agent decision making involves making group decisions, forming coalitions, allocating scarce resources, bargaining (negotiation) and arguing based on multi-agent interaction with its logical foundations (Wooldridge, 2009).

Multi-agent simulation is also based on the principle of the transformation between part and whole (Clark, 1997). The whole is composed of numbers of parts whose interactions contribute to the overall behaviour, which could also affect the behaviour of each part. The stability of the whole depends on the dynamic balance between two sets or multi-sets of forces that influence and maintain each other. The meaning can change even if the form does not because the environment is dynamic as the results of both globalisation and localisation. Multi-agent simulation is the exchange of information between different local settings and dynamic nodes. The relations between games and agents are the relations of global and local patterns from the perspective of a graph network, which could be a large hierarchical neural network, in which society and individuals are analogised to brain and neurons.[8]

[8] A type of graph network has been developed with hash tables and utilised in the experiments, "Incongruity" and "Extensibility and Other Features"

3.5.2 General Implementation of Simulations

The simulations in this thesis aim to explore the role of grounded language played for design in artificial creative systems. The process of simulation is illustrated in Figure 3.6. Firstly, a game environment is initialised by setting the number of agents, context size, the number of designers for each generation game, the success thresholds of guessing games and generation games, minimal game runs and maximal game runs. Then the loop of guessing games is implemented. when the success rate of guessing games reaches its threshold, the loop is stopped and new loop for generation games are implemented. After the success rate of generation games gets to its threshold, the simulation is completed.

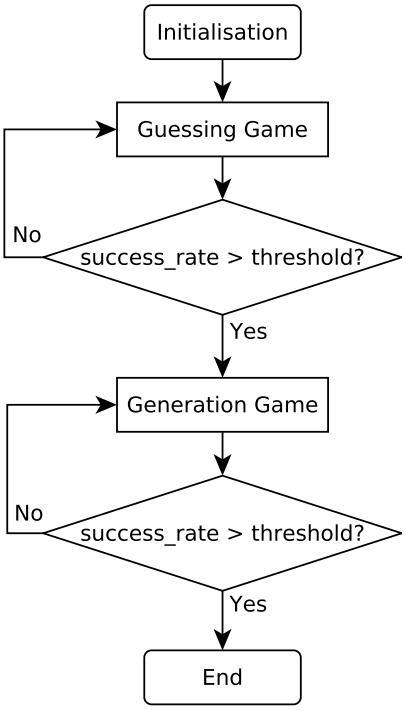


FIGURE 3.6: The Process of Multi-Agent Simulation for Evolving Languages

The agents in these simulations have the ability to produce simple utterances and their combinations, assess the utterances generated by other agents or themselves, and use these utterances to generate new designs and evaluate them. Through simulation, artificial languages may evolve via neural networks with distributed representations. Goals can be fed to language games for specific curiosity such as incongruence, exaggeration and elaboration.

Colours, shapes and geometric relations are used as sample-subjects. Numerous topics can be generated based on different weight settings and random variations of the samples. Agents generate utterances to represent new topics which could become designs

or artworks. The many-to-many mappings between utterances and meanings (i.e., topics) evolve through repeated language games. In guessing games, if 70% to 80% [9] of communication succeeded, i.e., most of associations between utterances and meanings are shared by agents, they can start playing generation games to generate requirements (i.e., new utterances) and new designs based on the shared languages.

In the simulations, the associations between utterances and their represented design features, relations and combinations are strengthened or weakened through repeated language games. Consequently, some representations become knowledge stored in the domain for training next generations. A group of agents would prefer a certain set of combinations and evolve particular representations that are likely to develop a culture based on their shared languages forming a clique.

3.5.3 Agents' Functions

To satisfy the requirements of the multi-aget simulations for evolving languages, a number of functions are developed for agents to play language games. For guessing games, agents are capable of selecting topics from contexts, generating utterances to represent the topics, and guessing topics according to their related utterances and contexts. To save memory and build up relations between different topics, the topics need to be categorised. Each category may be associated with a few of weighted utterances. These associations can be regulated as association rules stored in agents' long term memories. The methods of categorisation can be artificial neural networks or the usage of general prototypes and tolerances.

For generation games, agents should be able to generate compositional utterances representing certain design briefs as requirements, generate designs according to the requirements, and select designs by the aid of certain evaluation technologies such as hedonic functions. To organise and process compositional meanings and utterances, a number of rules based on context free grammar may be developed. Compositional utterances can be generated by tracing forward rules to expansions, then mapping the expansions to relevant utterances by the function named production. On the contrary, new designs could be generated by parsing the utterances to terminal meanings, which could be traced backward to certain rules, then tracing forward these rules to get new terminal meanings. In addition, some hierarchical relations could be stored in graph networks, and new designs could be generated by tracing edges through different paths.

To realise the above functions and some communication strategies, short term memory (STM) and long term memory (LTM) are used by each agent. The former is mainly used to store the last communication histories (see Algorithm 3) such as the last ten

[9] 70% to 80% is suitable for agents to share a common language while retaining some ambiguity for creative actions in the following generation games.

results of playing language games with other agents. This information can be used to select its partners for next language games that has been used in the simulation of clique formation (see Sec. 6.3). And STM is also utilised to store the keys of active rules and related active expansions, and the keys of active terminal meanings and related active utterances for un/learning them at the end of each language game. The latter, long term memory, is utilised to not only store categories generated by the evaluation based on general prototypes and tolerances/variances or artificial neural networks such as Self-Organising Map, the network based on Adaptive Resonance Theory, or Growing Neural Gas, but also store association rules and instances of topics and their related utterances. Association rules include the associations between meanings/categories and utterances such as {small:[(0.37,"qe"),(0.09, "po"),(0.98, "hu")]}, the associations between rules and expansions such as {rule:compoSize, expansions: [(0.34 very size), (0.12 size very), (0.45 size size)]}. The numbers in the rules are the weights of associations. The instances contain all accepted associations between topics/designs and utterances with frequency which can be used to select familiar utterances.

The abstract structure of an agent's functions is illustrated in Figure 3.7. The procedures of related functions are described in Appendix A including primary functions, the functions for playing guessing games and the functions for playing generation games.

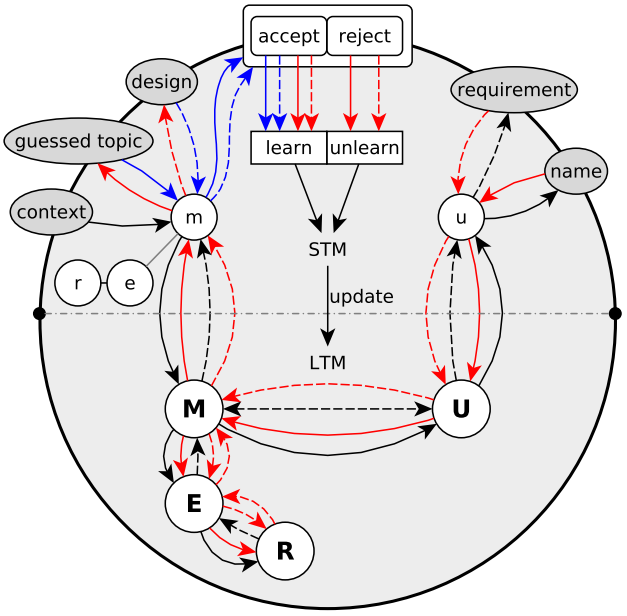


FIGURE 3.7: The structure of an agent for playing language games

In Figure 3.7, "m" is meaning, which could be the category of selected topic from context, the category of guessed topic, or the category of design. "u" is utterance, which may be the name of selected topic or the design brief of client-agent's requirement. The pair of "m" and "u" is the active associations between meanings and utterances, Aa(m,u). "r" is rule while "e" is expansion. the pair of "r" and "e" is the active associations between

rules and expansions, Aa(r,e). Both Aa(m,u) and Aa(r,e) are stored in Short Term Memory (STM). "M" is a set of collected meanings while "U" is a group of accumulated utterances. "R" means rules while "E" means expansions. The associations of "M" and "U", A(M,U), and the associations of "R" and "E", A(R,E), are stored in Long Term Memory (LTM). A(M,U) and A(R,E) are the collections of a number of A(m,u) and A(r,e) respectively through language games while A(m,u) and A(r,e) are generated in each language game or extracted from A(M,U) and A(R,E).

The structure of an association between a meaning and utterances is A(m,u) = {meaning, a number of weighted utterances, prototype, tolerance}. For example, the association of small size and utterances could be {small: [(0.37,"qe"),(0.09, "po"),(0.98, "hu")], prototype:0.2, tolerance:0.2}. As can be seen from this example, a weighted utterance = [weight, utterance]. In addition, a meaning can not only be a unit-meaning represented by a unit-utterance but also be a combination of unit-meanings represented by a compositional utterance, i.e., a sequence of unit-utterances.

The structure of an association of a rule and expansions is A(r,e) = {rule, a number of weighted expansions}. For instance, the association rule of compositional size could be {compo-size: [(0.34, very, size), (0.21, size, very), (0.27, size, size)]}. The number of recursions should be considered for some rules such as {very: [(0.01, very, very, (1 2))]}, which is limited with max-depth 2: "very" → "very very very very". 1 means this rule has been used once: "very" → "very very". Thus a weighted expansion = [weight, expansion, max-depth]. An expansion is a set of sub-rules or a number of unit-meanings.

The procedures of playing a guessing game and a generation game are illustrated with solid arrows and dashed arrows respectively (see Fig. 3.7). Solid black arrows represent the flow of selecting topic from context (see Algorithm 31, 32, 33) and naming the topic (see Algorithm 34) by playing the role of speaker. Solid red arrows show the procedure of guessing topic from its name and context (see Algorithm 35), and learning (see Algorithm 25, 27) or unlearning (see Algorithm 26, 28) the game result by playing the role of listener. And solid blue arrows direct the flow of evaluating guessed topic by comparing it with "m" which may be the selected topic or category, and learning (see Algorithm 25, 27) the succeeded game result by playing the role of speaker.

Dashed black arrows display the process of generating requirement (see Algorithm 36) by playing the role of client. The start point can be the combination of existing utterances (see "U" in Fig. 3.7) or generating a compositional prototype by tracing forward the associations of rules and expansions (see "R" → "E" → "M" in Fig. 3.7). Dashed red arrows represent the procedure of generating deign for the requirement (see Algorithm 37) and learning (see Algorithm 25, 27) or unlearning (see Algorithm 26, 28) the game result by playing the role of designer. If the rules of Context Free Grammar, i.e., the associations of rules and expansions are used by tracing backward the meanings parsed from requirement to certain rules (see "M" → "E" → "R" in Fig. 3.7), then tracing

forward the rules to new meanings (see "R" → "E" → "M" in Fig. 3.7), interesting novel design could be generated. Dashed blue arrows instruct the procedure of evaluating design (see Algorithm 38, 6), and learning (see Algorithm 25, 27) the succeeded game result by playing the role of client. At the end of each language game, long term memory (LTM) mainly including the associations between meanings and utterances and the associations between rules and expansions are updated.

3.5.4 Simulation with Graph Networks

A multi-agent simulation can be implemented using graph networks to store the structure of simulation, run the simulation and evaluate the results. Both the environmental settings such as the context of subjects and population, and the memories of agents related with the communication recording, representations of design prototypes with the association between topics and utterances, and context free grammar rules for generating designs and utterances can be stored in graph networks. For example, some edges of the networks are the rules of connecting nodes of operators and operands to generate and expand new designs. The storage of simulation using graph networks has been implemented in the experiment, Extensibility and Other Features (see Section 5.4).

Besides storing data, graph networks can be used to run simulations by selecting and implementing functions using conditional edges. For example, functions and their input and output are represented as nodes while conditions for filtering functions and connecting or disconnecting different functions are represented as edges. This can be easily realised using s-expressions because they take functions (operations) as data (e.g. '(let ((n (car (get-hash-value ,g '(nodes b input))))) (if (< n 10) (incf n) (progn (remhash 'ba (get-hash-value ,g 'edges)) n)))). A test of running functions in a graph network is presented in Algorithm 2. Two nodes $'a$ and $'b$ are added in a graph network. They are connected with $edge('a,'b)$ and $edge('b,'a)$ that means an infinite loop occurs as $'a \rightarrow' b \rightarrow' a \rightarrow \ldots$ To prevent from the infinite cycling, some functions are added into the nodes, For node $'a$, its input is updated from the output of node $'b$, and its output maintains the same as the first item of its input by running its own function. For node $'b$, its input is updated from the output of node $'a$, and its output is updated by running its own function. Each time, the output is increased 1, when it reaches 10, $edge('b,'a)$ is removed that breaks the cycle $'a$-$'b$-$'a$ and the tracing process is stopped. Thus the loop is broken after running 10 times via implementing the functions embedded in the nodes. The related algorithms of graph networks are listed in Appendix B. More practical implementations of integrating functions in the graph networks would be explored in future work.

After completing simulations, the results can be evaluated using graph networks. For example, the degree of instances is the number of associations between design prototypes and utterances. The centrality of a network could be used to uncover some mechanisms of

Algorithm 2 Running Functions in A Graph Network

```
1: procedure RUNFUNCTIONSGN
2:     g ← makeGraph()
3:     g ← addNode(g, 'a, {'input : [0], 'function : (return 'input[0]), 'output : 0})
4:     g ← addNode(g, 'b, {'input : [0], 'function : (n ←' input[0], if n < 10 : n ←
   n + 1, else: removeEdge('ba, g['edges]), endIf, return n), 'output : 0})
5:     g ← addEdge(g, 'ab, ['a,' b])
6:     g ← addEdge(g, 'ba, ['b,' a])
7:     trace ← traceForwardNodes(g, ['b])
8:     return trace
9: end procedure
```

social creativity. The evaluation using graph networks has been successfully implemented in the experiment, Ambiguity (see Section 5.1)

3.6 Machine Learning

Topics are categorised using machine learning to find unusual and interesting concepts in the evolution of artificial languages. Compositional languages evolve at a sociocultural level[10] using both short- and long-term memory[11] related with three types of artificial neural networks: Self-Organising Map, the network based on Adaptive Resonance Theory, and Growing Neural Gas combined with probability theory.

3.6.1 Short Term Memory

Short term memory (STM) is used to store flexible, sensitive and dynamic data facilitating adaptation to real-time changes. Timely information such as current activated CFG rules in the games used to explore incongruity (see Sec. 5.3) and extensibility (see Sec. 5.4) can be stored in STM. STM is also used in the simulation, Clique Formation (see Section 6.3), to adjust communication preferences simultaneously according to the last seven number of gaming results with each agent (see Algorithm 3). In other words, an agent's success rates of recent language games can be stored in its STM for selecting whom to play with in next language game.

[10] To model a sociocultural system, ecological-adaptation is realised by both guessing games (see Section 3.3.1) adapting to the environment and generation games (see Section 3.3.2) influencing the environment. Firstly, a number of objects in context are categorised and named, and the categories and names are adjusted and shared among agents through guessing games; then the agents generate new utterances based on their shared languages to drive the production of novel designs through generation games, some novel designs would become new parts of the environment. In addition, historical-diffusion is reflected by the communication between mature agents and naive agents (see Section 6.1 and 6.2) and the acculturation of multi-cultures influenced by the tolerance of agents and the number of communications (see Section 6.3).

[11] Communication preference and categorisation are implemented by storing and updating data in short term memory and long term memory, respectively.

Algorithm 3 Updating Short Term Memory

```
1: function UPDATESTM(stm, gamingResult, otherAgent)
2:     if otherAgent ∉ hashKeys(stm) then
3:         stm[otherAgent] ← [0, 0, 0, 0, 0, 0, 0]
4:     end if
5:     stmOA ← stm[otherAgent][1 :]                ▷ remove the first item of stmOA
6:     stmOA ← append(stmOA, gamingResult) ▷ add new gaming result at the end of stmOA
7:     stm[otherAgent] ← stmOA
8:     return stm
9: end function
```

3.6.2 Long Term Memory

Long term memory (LTM) is for storing relative stable information which are filtered and categorised compared with that stored using short term memory (STM). STM is capable of responding timely change of communication and storing provisional information which may be not necessary to be retrieved in future. On the contrary, LTM is responsible of collecting information over a long period of time for evaluating current results based on previous categorised data and predicting future trends. It is important for curious agents to grow from naive to mature, and develop complex networks for dealing with the environmental change on a macro-level using LTM, which is partly supported by STM providing raw information and the results of timely responses.

In this thesis, LTM is operated using neural networks including Self-Organising Map (SOM), the network based on Adaptive Resonance Theory (ART), and Growing Neural Gas (GNG) for categorisation. The categorisation with prototypes can be adjusted by setting different tolerances.

3.6.2.1 General Neural Networks

Self-Organising Map (SOM) is used as a general neural network to map utterances on objects as well as their compositions. Categories and relevant nodes (prototypes) are continually being adjusted to map samples such as 2D coordinates. For example, in the simulation, Compositional Representation of Rectilinear Relation (see Section 4.2), several utterances compete to signify a given geometric meaning. Both utterances and geometric meanings are represented by HRRs using high dimensional vectors. The network develops systematically regular mappings between meanings and utterances through a number of generations.

The combination of HRRs and SOM could be an explanatory mechanism for both language evolution and acquisition due to the potential of neural-like representations for addressing the symbol-grounding problem (Levy and Kirby, 2006). The generated SOM

would be taken as an attractor and trigger to generate new SOMs by mutating, crossing-over, mirroring and reversing etc.

In SOM, the number of nodes (i.e., neurons or units), however, is fixed, which limits the function of categorising a continuously growing data space. To address this issue, some extensible neural networks are implemented.

3.6.2.2 Extensible Neural Networks

Extensible Neural Networks used in this thesis are the network based on Adaptive Resonance Theory (ART), and Growing Neural Gas (GNG).

ART network (Grossberg, 1976) is applied to the categorisation of multi-dimensional data consisting of different topics such as colours and shapes, and their relations. The advantage of ART is that it not only retains existing categories but also adds new categories for unfamiliar input that is useful for categorising the growing number of utterances and meanings generated in language games.

GNG is based on Self-Organising Map but has the ability to extend categorisations dynamically by adding more neurons when new input surpasses the threshold of tolerance of existing neurons (Fritzke et al., 1995). It has advantages over both SOM which is not extensible, and ART which lacks the ability to predict unseen situations based on the inputs (Saunders, 2002).

3.6.2.3 Prototype and Tolerance

The number of categorisations is not only determined by the scale of input data but also affected by the tolerance of a neural network. In this thesis, the change of tolerances and generated prototypes is regulated in the normal range [0,1], which can be applied using the probability density function of truncated normal distribution (see Fig. 5.18 and Fig. 5.19) based on probability theory.

The degree of "similar-yet-different" can be adjusted by setting different tolerance values. Each prototype is the middle point of a fuzzy range (Zadeh, 1996) while the width of the range is determined by tolerance. These fuzzy ranges may overlap each other. With the aid of fuzzy sets, the boundaries of initialised categories become flexible. In addition, fuzzy categorisation may be used to represent the ambiguous meanings of different sizes, such as "very small" or "large". The primary representations can be developed via the evolution of holographic reduced representations of number 0 to 10 mapping to the range [0,1] for fuzzy logic and categorisation. The overlap of these categories could be the source of creativity. Further, mapping one feature with another feature belonging to different categories could be accomplished by matching, swapping and displacing

relative degrees of these features, e.g. "red": $D_{colour}(0.1) \rightarrow$ "triangle": $D_{shape}(0.1)$. Exaggeration may also be implemented, e.g. "very red": $D_{modifier}(0.9) \otimes D_{colour}(0.1)$ and "little red": $D_{modifier}(0.1) \otimes D_{colour}(0.1)$.

Hierarchical categorisations can be developed by setting up different tolerances of neurons to a neural-network with various levels. A neuron with more tolerance is looser and more ambiguous that is capable of covering more areas with fuzzy boundary. By contrast, a neuron with less tolerance is more accurate and stricter that is good at differentiating meanings with small differences. For example, three levels of neural networks can be developed by dividing neurons into three parts with different tolerances—0.1 (accurate), 0.3 and 0.5 (loose)—thus allowing concepts to be connected by more tolerable neurons, and differentiated by less tolerable neurons. An application of this method could be categorising sizes on different levels (see Table 5.20). Further, different categorisations for the same input data could be realised by the aid of multiple tags related to the attributes of quality and quantity.

3.7 The Methods of Evaluating Interest

The methods of evaluating the interestingness of designs and utterances in this thesis are using hedonic functions (Saunders and Gero, 2001a) and graph networks (West et al., 2001). Hedonic functions include linear hedonic function and nonlinear hedonic function based on the Wundt curve (see Fig. 3.8). Several Wundt curves with different extrema and different points of inflection, or only part of a Wundt curve, such as reward or punishment, can be utilised to evaluate interestingness (see Algorithm 4). Graph networks can be used to construct the relationship between utterances and meanings (e.g. design prototypes), and evaluate the change of their structures, specifically in relation to new utterances, novel meanings and new links between utterances and meanings in language games.

The calculation of the hedonic value of novelty can be implemented using Algorithm 4. Given $novelR = 0.25$, $novelP = 0.5$, $maxR = 0.85$, $maxP = 0.95$, $pR = 2$, $pP = 2$, $scale = 10$, the Wundt curve in range [0,1] can be generated as Figure 3.8. $novelR$ and $novelP$ determine the x-coordinate of peak-point, i.e., preferred novelty, of the Wundt curve. The relationship between them is $novelR < preferredNovelty < novelP$. And the distance between $novelR$ and $novelP$ affects the shape of the peak area, which becomes flat when the distance increases, whereas it becomes sharp when the distance decreases. $maxR$ is the max value of reward while $maxP$ is the max value of punishment. Too novel design could be evaluated as negative interesting design because $maxP > maxR$. The smoothness of the curve is influenced by pR and pP. Small pR and pP make the curve very smooth even flat.

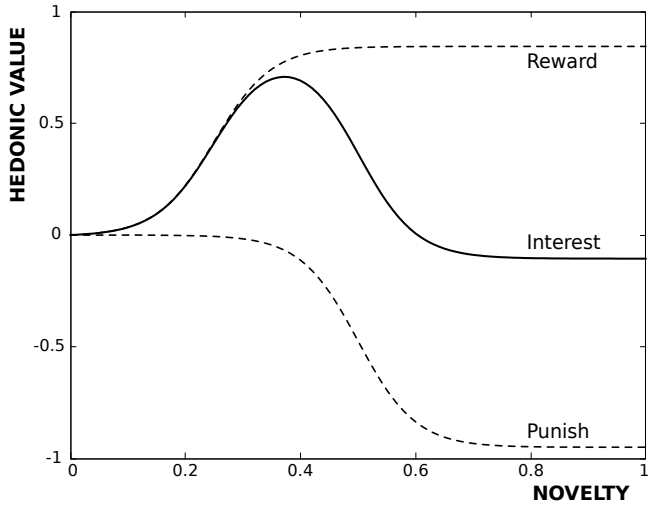


FIGURE 3.8: The Wundt curve

Algorithm 4 The Wundt Curve

function WUNDTCURVE($novelty, novelR, novelP, maxR, maxP, pR, pP, scale$)
  $novelty \leftarrow novelty \times scale$
  $novelR \leftarrow novelR \times scale$
  $novelP \leftarrow novelP \times scale$
  $R0 \leftarrow maxR/(1 + \exp(pR \times novelR))$
  $P0 \leftarrow maxP/(1 + \exp(pP \times novelP))$
  $R \leftarrow maxR/(1 + \exp(-pR \times (novelty - novelR)))$ ▷ Key line
  $P \leftarrow maxP/(1 + \exp(-pP \times (novelty - novelP)))$ ▷ Key line
  $R \leftarrow R - R0$ ▷ $R = 0$ when $novelty = 0$
  $P \leftarrow P - P0$ ▷ $P = 0$ when $novelty = 0$
  return $R - P$
end function

If a designer-agent uses a hedonic function, it will motivate the search of the space of possible designs to locate novel designed objects which can be measured by the distance from a set of prototypes constructed from previously generated patterns (Saunders, 2011). Random combinations of words might produce new ideas and questions. These ideas could be realised by agents fitting these words into a particular design space, which can be evaluated using hedonic functions. It is possible to integrate language games with models of individual and social creativity without undermining the grounding of words for describing works within an evolving language (Saunders, 2011).

Some features of a graph network such as degree and centrality could be used to evaluate interest. To evaluate the interest of an artificial language, a network of the language should be generated by connecting the nodes representing meanings and utterances. The connections are many-to-many because a meaning can be represented with several utterances while an utterance can represent several meanings. The degree of a node is the number of edges connecting the node with other nodes. The difference between the highest degree and the lowest degree, as well as average degree, may be used to evaluate

novelty and appropriateness. The centrality of the nodes, such as closeness centrality or degree of centrality, can be measured to find the most interesting nodes.

3.7.1 Variation of Wundt Curve (Sine Curve)

The Wundt Curve can be simplified to a sine curve [0,1] for convenience in computation of dynamic change of interesting peak-points (see Fig. 3.9 and Fig. 3.10). Preferred similarity (i.e., the opposite of novelty) can be easily changed by adjusting the peak-point of sine curve (see Fig. 3.10). And threshold is added into sine curve to constrain the selecting range (see Fig. 3.9) that can take the role of negative impact of excess novelty shown in the Wundt Curve (see Fig. 3.8).

In sine curve, novelty is evaluated by measuring the similarity between design and prototype. And similarity is regulated into range [0,1]. Interesting design can be selected by checking if the measured similarity was in the range of client-agent's preferred similarity. So it is not necessary to consider the negative impact of excess novelty, which may also be realised in sine curve by adjusting the range of y-axis: INTEREST from [0,1] to [-1,1].

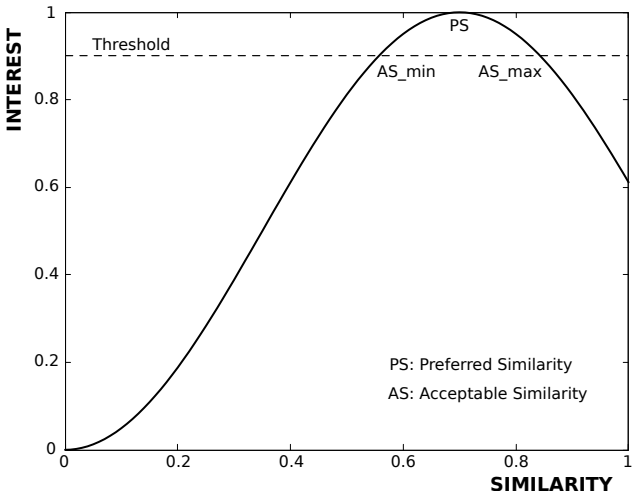


FIGURE 3.9: Sine curve (preferredSimilarity=0.7, selectionThreshold=0.9)

As can be seen form Figure 3.9, the preferred similarity (peak-position) of prototype and design is set to 0.7, which means that the difference between original object and new works should be only 0.3; the selection threshold is set to 0.9, which means that the tolerance of preferred similarity is 0.288 (min:0.560, max:0.848).

The algorithm of sine curve [0,1] used to compare the similarity of prototype and design (x denotes similarity) with the preferred similarity (x_0 denotes preferred similarity) is described in Algorithm 5.

The similarity of prototype and design, especially their outlines, can be measured using OpenCV (Open Source Computer Vision) (Bradski et al., 2000). If the score obtained by

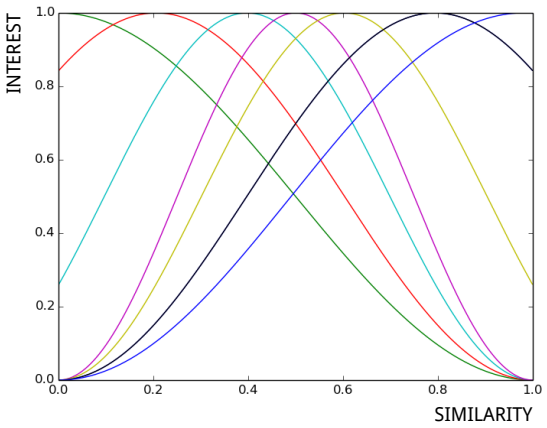


FIGURE 3.10: Sine curves

Algorithm 5 Sine Curve

function SINECURVE(x, x_0)
  $A \leftarrow 1/2$
  $\varphi \leftarrow -\pi/2$
  $D \leftarrow 1/2$
  if $x_0 < 0.5$ **then**
    $x \leftarrow x + (1 - 2 \times x_0)$
    $x_0 \leftarrow 1 - x_0$
  end if
  $w \leftarrow \pi/x_0$
  return $A \times sin(w \times x + \varphi) + D$
end function

comparing the measured similarity with the preferred similarity via sine curve surpasses the selection threshold, the design becomes a candidate. The design with the highest score among the candidates is selected as the winning design. The procedure of selecting design using sine curve is described in Algorithm 6. In this algorithm, the given value of threshold and that of preferredSimilarity, and the calculated value of similarity and that of score are all in range [0,1]. The Wundt curve can also be used in this algorithm.

To test the impact of various evaluation criteria on design creativity, it is useful to combine preferred similarity with selection threshold to simulate the dynamic change of preferred similarity from small difference to great difference by gradually adjusting the preferred similarity from 0.9 to 0.1 (see Fig. 3.10). At the same time, the tolerance can be narrowed or expanded by increasing or decreasing the selection threshold respectively.

3.7.2 Euclidean Distance and Cosine Distance

Both the Euclidean distance and cosine distance are used to measure the distance between two numerical arrays in the same dimensions, which denote topics or designs. If the difference of the whole change of the data is more important than the difference

Algorithm 6 Interesting Choice

function INTERESTINGCHOICE($designs, prototype, threshold, difference_{prefer}, curve$)
  $candidates \leftarrow emptyList$
  for $design$ in $designs$ **do**
    $d \leftarrow difference(prototype, design)$
    if $curve =$ 'wundt **then**
      $difference_{reward},\ difference_{punish} \leftarrow difference_{prefer}$
      $score \leftarrow wundtCurve(d,\ difference_{reward},\ difference_{punish})$
    else if $curve =$ 'sine **then**
      $similarity \leftarrow 1/(d + 0.000001)$
      $similarity_{prefer} \leftarrow 1/difference_{prefer}$
      $score \leftarrow sineCurve(similarity,\ similarity_{prefer})$
    end if
    if $score > threshold$ **then**
      $candidates \leftarrow append(candidates,\ [score, design])$
    end if
  end for
  if $candidates$ **then**
    $winningDesign \leftarrow weightedChoice(candidates)$
  else
    $winningDesign \leftarrow False$
  end if
  return $winningDesign$
end function

of each item, cosine distance may be better than the Euclidean distance. For example, given $a = \{0.2, 0.4, 0.1, 0.6\}$, $b_1 = \{0.3, 0.3, 0.2, 0.5\}$, $b_2 = \{0.3, 0.6, 0.3, 0.9\}$, the results of their cosine distances and the Euclidean distances are $D_{Cosine}(a, b_1) = 0.034$, $D_{Cosine}(a, b_2) = 0.008$, $D_{Euclidean}(a, b_1) = 0.2$, $D_{Euclidean}(a, b_2) = 0.424$. As can be seen from these results, the cosine distance between a and b_1 is greater than that between a and b_2. They match the fluctuation differences (see Fig. 3.11) very well compared with the results of the Euclidean distance. But the Euclidean distance can be used to measure the accumulated distances between each pair of items.

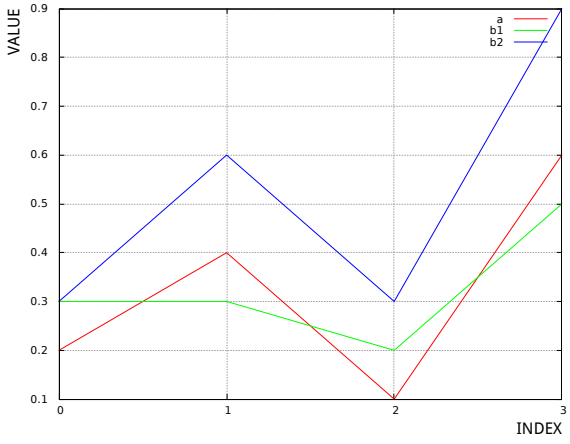


FIGURE 3.11: The difference between Cosine distance and Euclidean distance

To test the creativity of the evolved languages, besides the represented designs, symbolised rules and utterances may also need to be evaluated using String Metric algorithms, which include Levenshtein distance (Levenshtein, 1966) and Damerau–Levenshtein distance (Brill and Moore, 2000), Hamming Distance (Forney, 1966), Smith–Waterman algorithm (Mott, 2005), Sørensen–Dice coefficient (Duarte et al., 1999), Jaro–Winkler distance (Cohen et al., 2003), and Most frequent k characters (Seker et al., 2014) etc.

3.8 Conclusion

The evolution of artificial languages for creative designing is based on the Domain-Individual-Field-Interaction framework and implemented by following the processes of representation, communication and evaluation. In the DIFI model, individuals play the main roles of discovering new meanings, generating representations and developing knowledge via the cycling of domain, individual and field. The procedure (i.e., representation → communication → evaluation) is also a cycling process because it not only continually updates and accumulates representations, exchanges information through guessing games and generation games, but also maintains creativity and activity via the evaluation of design briefs and design works using continually updated similar-yet-different criteria.

The experiments described in this thesis are implemented in multi-agent simulations to simulate the communication between speaker-agents (initiators) and listener-agents (recipients) in guessing games and that between client-agents and designer-agents in generation games to evolve artificial languages for conceptual designing. In the evolution of artificial languages, weighted associations between utterances and meanings, Holographic Reduced Representations and Weighted Context Free Grammar are utilised to represent topics, designs and association rules. To evolve communication from random selection to specific selection of players, short term memory is used to store and update the newest successful or failed communications among agents. To categorise continuous input samples or generated works during simulation, long term memory is adopted to store categories, prototypes, association rules and instances and update them based on several machine learning algorithms including Self-Organising Map (SOM), the network based on Adaptive Resonance Theory (ART) and Growing Neural Gas (GNG). Interestingness is evaluated using hedonic functions based on the Wundt Curve and its sine curve variation with the measurement of difference between prototype and designs using cosine similarity and the Euclidean distance.

The computational model is developed based on existing work including the DIFI framework (Csikszentmihalyi, 1999), guessing games (Steels, 1998), generation games (Saunders and Grace, 2008) and curious agents (Saunders, 2002). The contributions of this thesis are listed below.

- A process of evolving compositional artificial languages for creative design via the combination of guessing games and generation games (see Fig. 3.6) is established based on the framework introduced by Saunders and Grace (2008).
- A basic model of curious design agent (see Sec. 3.5.3) for exploring creative features of language in design communication is built up based on curious agents developed by Saunders (2002).
- Weighted Context Free Grammar (WCFG) (see Sec. 3.2.3) is developed to generate elaborated representations which could be interesting (i.e., unusual, strange, incongruent) by the aid of inverse weighted random choice of rules.
- Interleaving Representation (see Sec. 3.2.2.3) is suggested to be an alternative of circular convolution for more "accurate" representations.
- Sine Curve in range [0,1] (see Sec. 3.7.1) is developed for the convenience of measuring interest based on the dynamic change of interesting peak-points.
- The evaluation of creativity (see Sec. 3.4) is established at two levels, individual level and societal level. The measurement of individual diversity is developed using Equation 9, 10, and 11.
- Some features of a graph network such as degree and centrality are used to evaluate interest (see Sec. 3.5.4).
- Cosine distance rather than the Euclidean distance is suggested to be used when the difference of the whole change of the data is more important than the difference of each item (see Sec. 3.7.2).

Chapter 4

Evolving Compositional Languages for Creative Design

Achermann et al. (2001) identified three aspects of compositional language: components, connectors and compositional rules. Connectors are transformed to a part of more complicated components via the evolution of compositional language towards hierarchical structures with continuously generated new meanings and representations reflecting creative features such as ambiguity, scalability and extensibility. In brief, not only can the extraction of meanings from objects be innovative but the composition of new meanings and new descriptions in a hierarchical structure can be more creative. The rules of connecting syntax and semantic ontology and the mechanism of evolving them to complex networks can be realised via multi-agent simulations doing both "observation" (discovering) and "thinking" (creating and rediscovering).

4.1 Compositional and Holistic Language

The aims of this experiment are to test whether compositional language is more efficient and useful than holistic language in design creativity and to evaluate different methods of topic selection. The difference between holistic language and compositional language is related to different generation processes. For example, as holistic language, "small-red-triangle" is obtained by finding the relevant part in three-dimensional space (axis-x: size, axis-y: colour, axis-z: shape) directly. As a compositional language, "small-red-triangle" is generated by combining the previously generated utterances "ba-lo-ke" related to the meanings, "small", "red" with "triangle". The relationship between holistic and composition could be explained as Gestalt, "the whole is other than the parts instead of rather than the parts or more than the parts" (Roser and Hebela, 2015).

4.1.1 Subjects

The subjects of this experiment comprise three sizes – small, medium and large, six colours – red, yellow, green, cyan, blue and magenta, and five regular shapes – triangle, square, hexagon, circle and star. 90 samples are generated by combining them (see Fig. 4.1).

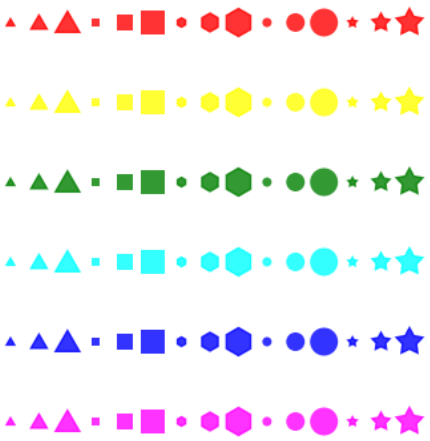


FIGURE 4.1: The combinations of size, colour and shape

4.1.2 Implementation

The experiment is implemented using guessing games based on the DIFI framework to test the effectiveness of both compositional language and holistic language (see Fig. 4.2).

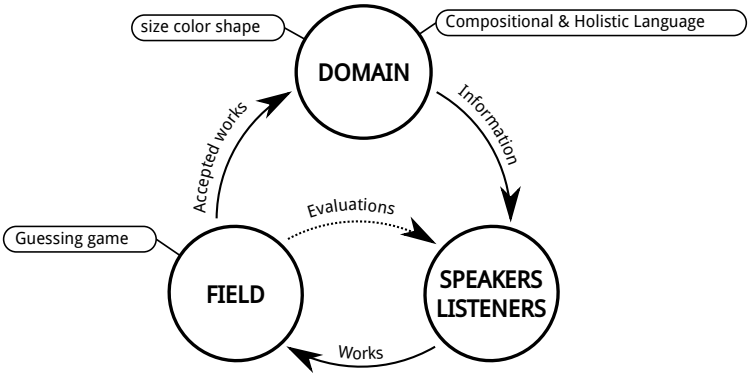


FIGURE 4.2: The Domain-Individual-Field-Interaction (DIFI) framework of evolving compositional and holistic language

Six agents continue playing guessing games (see Fig. 4.3) to represent the combinations of size, colour and shape with compositional utterances (see Table 4.1) and holistic utterances, respectively, until the success rate reaches 70%. To test the impact of topic

selections on the efficiency of language games, several methods of selecting topics are adopted, including "random, common, different, confident and unconfident" (see Table 4.2). "Common" means selecting the most common one, which is the most difficult to distinguish from others; "different" is the opposite of "common"; "confident" means that the speaker-agent always selects the one it feels confident about, i.e., the association between the selected topic and the related utterance is very strong in its memory; "unconfident" is the opposite of "confident".

TABLE 4.1: Compositional representations (size, colour & shape)

Association	Feature (F)	Attribute (A)
Meaning	size	small, medium, large
	colour	red, yellow, green, cyan, blue, magenta
	shape	triangle, square, hexagon, circle, star
Utterance	vowel (V)	consonant (C)

Compositional representation:
$A_{size}F_{size}A_{colour}F_{colour}A_{shape}F_{shape} \rightarrow C_{size}V_{size}C_{colour}V_{colour}C_{shape}V_{shape}$

Holistic representation:
$A_{size}F_{size}A_{colour}F_{colour}A_{shape}F_{shape} \rightarrow char_0char_1$

TABLE 4.2: Topic selection

SelectTopic	Random	Common	Different	Confident	Unconfident
Composition	cRm	cCn	cDn	cCt	cUt
Holistic	hRm	hCn	hDn	hCt	hUt

Existing utterances are recombined to generate novel utterances for testing the creativity of compositional languages. For example, "reddish yellow" is the combination of "red" and "yellow" relating to the holistic colour "orange". In addition, if the order of a sequence or the proportion of pigments is considered, "reddish green" could be different from "greenish red" (see Table 4.3). In language games, the objects can be pairs of coloured shapes such as "blue triangle and blue circle", "green circle and red triangle". Agents evaluate the similarity or difference between each item of the compositional objects to find relevant meanings such as the same colour ("blue") and different shapes ("triangle" and "circle").

TABLE 4.3: The rules of combining colours

No.	Rule		Expansion
0	combine colour	→	(operator colour colour)
1	operator	→	gradiate / mix / blend
2	colour	→	red / green / blue
3	reddish green	→	(operator red green)
4	greenish red	→	(operator green red)

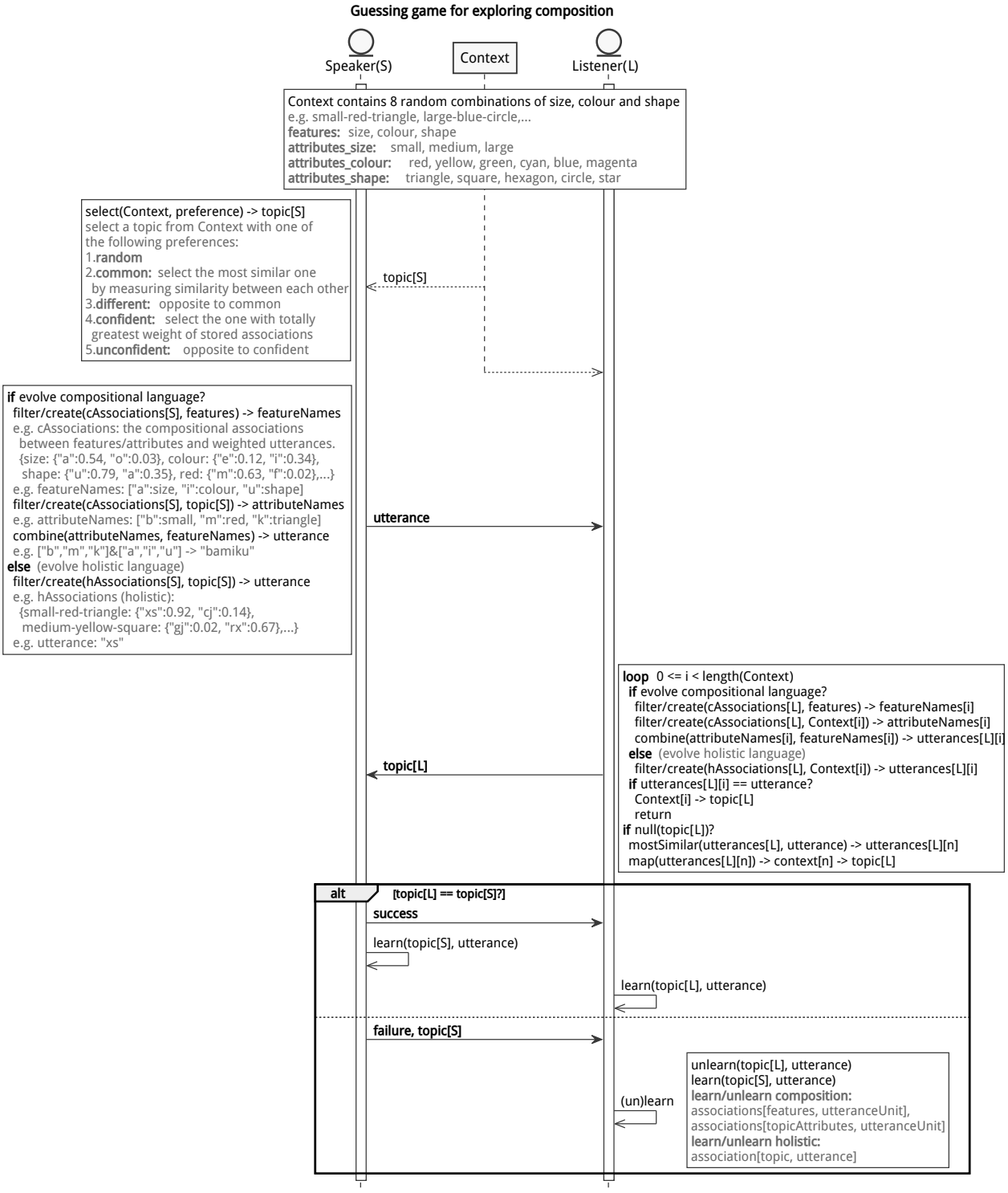


FIGURE 4.3: Guessing game of compositional & holistic language

4.1.3 Results

Four types of data—the number of games being played, discrimination, consistency, and density of utterances— are collected and analysed.

TABLE 4.4: The number of games being played

No.	Type	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6	Sim7	Sim8	Sim9	Sim10	Mean
0	cRm	460	390	194	320	220	327	423	337	303	230	320
1	cCn	283	566	243	586	287	327	287	440	290	233	354
2	cDn	134	243	223	87	127	240	240	110	184	380	196
3	cCt	44	77	107	34	60	21	164	80	84	37	70
4	cUt	759	2466	3325	2383	4157	1092	1132	889	962	1218	1838
5	hRm	4496	4047	3827	3924	4287	4932	3834	3618	4446	4094	4150
6	hCn	4456	4283	4087	3312	3688	4047	3441	4310	4443	4177	4024
7	hDn	4167	4024	3788	3654	3744	3821	4014	3741	4030	4506	3948
8	hCt	380	483	476	530	423	430	453	480	453	387	449
9	hUt	264932	262633	329365	289109	291981						287604

4.1.3.1 The Number of Games Being Played

Data are collected on the number of guessing games being played in 10 types of simulations (see Table 4.4). The results show that the guessing games evolving compositional languages are more efficient than that evolving holistic languages (see Fig. 4.4). The average cost of the former is 70 to 1838 runs while the average cost of the latter cost is 449 to 287604.

Although compositional language is more efficient than holistic language, in both simulations agents need to play a large number of guessing games to reach the success rate, 0.7, when the speaker-agent selects an unconfident topic in each game. By contrast, agents can reach the success rate by playing only a small number of guessing games when the speaker-agent selects a confident topic in each game (see Fig. 4.5). This is because it is easy to be successful when the speaker-agent selects the most familiar topic to communicate with the listener-agent each time.

In the games of evolving compositional languages, "common" selection costs more times than "different" selection to achieve success whereas both "common" and "different" selections cost almost the same number of runs in the games of evolving holistic language (see Fig. 4.5). This is because a holistic language takes every sample as an independent topic, so it does not matter whether the selected topic is the most common or the most different compared to other samples. A compositional language, by contrast, generates each utterance representing different topics by combining their shared features. The most common sample shares more similar attributes than the most different one compared with other samples in context. Therefore, it is easier for the listener-agent to make mistakes when guessing the common topic.

4.1.3.2 Discrimination

Discrimination refers to how much these 90 samples are distinguished from each other by the evolved languages. The results show that all types of simulations surpass 75% of

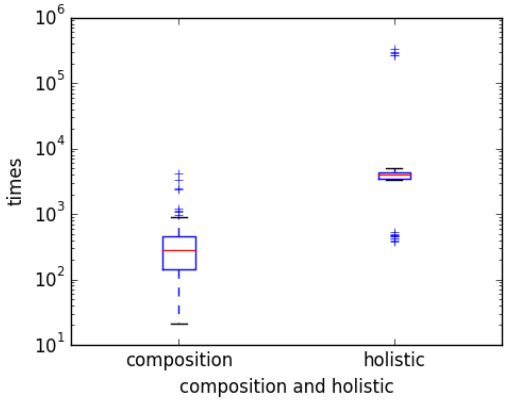


FIGURE 4.4: The number of games being played (general composition and holistic)

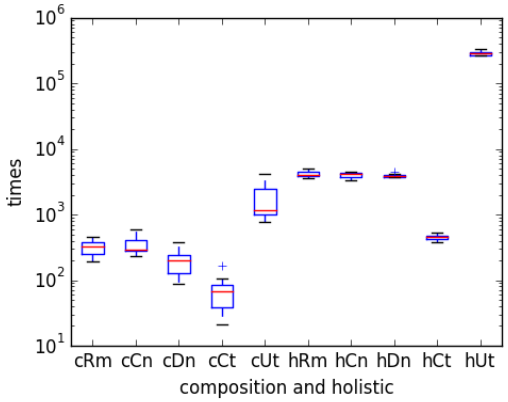


FIGURE 4.5: The number of games being played (detailed composition and holistic)

successful discrimination (see Table 4.5 and Fig.4.6). The average lowest result is obtained for the "confident" topic selection in the compositional languages and the average highest result is obtained for the "unconfident" selection in the holistic languages. The former may be due to the pre-mature evolution of the compositional languages when agents only play guessing games on familiar topics, and vice versa. Arguably, the most interesting result is that random topic selection in the compositional languages obtains the highest discrimination score of all selections in the compositional languages. This might mean that topic selection without preference could get the most diverse topics for evolving compositional languages since it reaches higher discrimination during the simulation.

4.1.3.3 Consistency

Consistency refers to the extent to which agents share the same language. As can be seen from Table 4.6 and Fig. 4.7, the results for most types of topic selections surpass 85% consistency with the exception of the "confident" selection in both the compositional languages and the holistic languages, which averages 73% and 36% respectively. The

TABLE 4.5: Discrimination

No.	Type	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6	Sim7	Sim8	Sim9	Sim10	Mean
0	cRm	0.865	0.938	0.906	0.917	0.844	0.99	0.938	0.813	0.896	0.854	0.896
1	cCn	0.885	0.813	0.865	0.771	0.792	0.917	0.75	0.833	0.969	0.938	0.853
2	cDn	0.792	0.844	0.865	0.896	0.781	0.802	0.75	0.792	0.854	0.823	0.820
3	cCt	0.833	0.771	0.875	0.813	0.823	0.823	0.771	0.802	0.781	0.813	0.810
4	cUt	0.906	0.844	0.875	0.771	0.844	0.875	0.771	0.906	0.844	0.833	0.847
5	hRm	0.948	0.966	0.927	0.97	0.97	0.944	0.979	0.953	0.978	0.931	0.957
6	hCn	0.934	0.921	0.929	0.961	0.908	0.985	0.949	0.88	0.94	0.94	0.935
7	hDn	0.948	0.961	0.927	0.959	0.94	0.942	0.955	0.938	0.961	0.942	0.947
8	hCt	0.985	0.964	0.978	0.957	0.978	0.983	0.974	0.966	0.976	0.976	0.974
9	hUt	0.985	1.0	0.994	0.998	1.0						0.995

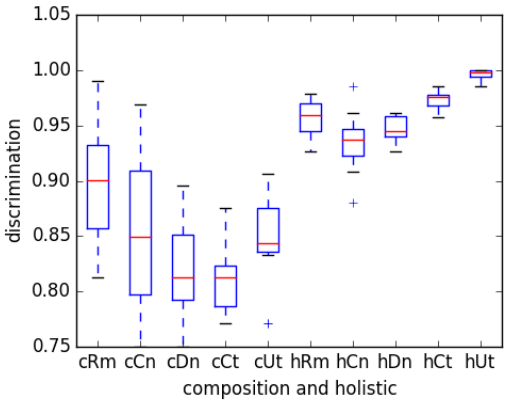


FIGURE 4.6: Discrimination (composition and holistic)

cause of low consistency is probably that agents focus on playing guessing games with familiar topics. When a speaker-agent meets a listener-agent who has the same familiar topic as that selected by the speaker, it will be easy to achieve success, and vice versa. Hence, some of the 30% failure rate may be attributable to those occasions in guessing games on which the speaker-agent selects a confident topic on which the listener-agent is not confident. Another reason for the low consistency of the holistic languages evolved via "confident" topic selection could be their early-maturing evolution. The data show that, on average, guessing games are played only 449 times before 70% success rate is achieved. This is much less than the result for other selections evolving holistic languages. However, cliques may be easily formed using the strategy of "confident" topic selection.

4.1.3.4 Density of Evolved Language

The density of a evolved language refers to how many features (for composition) or samples (for holistic) are covered by the total utterances from all agents at the completion of the guessing games. Results show that most types of topic selections in both the compositional languages and the holistic languages achieve 100% density with the exception

TABLE 4.6: Consistency

No.	Type	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6	Sim7	Sim8	Sim9	Sim10	Mean
0	cRm	0.941	0.953	0.941	0.835	0.929	0.941	0.929	0.953	0.918	0.965	0.930
1	cCn	0.929	0.976	0.918	0.953	0.953	0.941	0.918	0.941	0.953	0.918	0.940
2	cDn	0.871	0.929	0.871	0.847	0.847	0.871	0.929	0.835	0.859	0.906	0.876
3	cCt	0.718	0.718	0.847	0.659	0.753	0.565	0.835	0.824	0.776	0.612	0.731
4	cUt	0.918	0.906	0.882	0.941	0.871	0.906	0.953	0.894	0.953	0.918	0.914
5	hRm	0.951	0.933	0.953	0.938	0.949	0.94	0.942	0.964	0.953	0.942	0.947
6	hCn	0.949	0.938	0.933	0.947	0.929	0.947	0.931	0.949	0.94	0.956	0.942
7	hDn	0.956	0.942	0.947	0.947	0.931	0.936	0.938	0.953	0.949	0.942	0.944
8	hCt	0.311	0.358	0.367	0.389	0.36	0.353	0.356	0.382	0.376	0.318	0.357
9	hUt	0.88	0.898	0.929	0.942	0.933						0.916

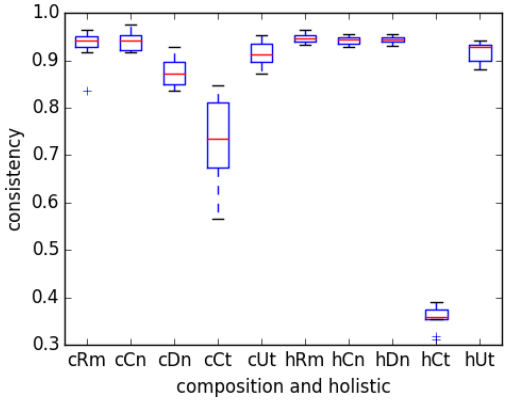


FIGURE 4.7: Consistency (composition and holistic)

TABLE 4.7: The density of all agents' utterances

No.	Type	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6	Sim7	Sim8	Sim9	Sim10	Mean
0	cRm	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
1	cCn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
2	cDn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
3	cCt	1.0	1.0	1.0	1.0	1.0	0.99	1.0	1.0	1.0	0.941	0.993
4	cUt	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
5	hRm	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
6	hCn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
7	hDn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000
8	hCt	0.996	1.0	0.998	0.998	0.994	0.998	1.0	1.0	1.0	0.996	0.998
9	hUt	1.0	1.0	1.0	1.0	1.0						1.000

of the "confident" topic selection which achieves 99.3% and 99.8% density respectively (see Table 4.7). This may be mainly due to the early-maturing evolution of languages when using "confident" topic selection. The "confident" selection only costs 35.7% and 11.4% of runs costed by the second lowest, "different" selection, in the compositional languages and the holistic languages respectively.

4.1.4 Conclusion

Compositional languages are more efficient than holistic languages regarding the number of games being played (see Fig. 4.4). Compositional languages may also be more creative than holistic languages due to the ambiguity of the former is greater than that of the later (see Fig.4.6, less discrimination means more ambiguity, which is a creative feature of language). In the evolution of compositional languages, both "random" and "common" topic selections are the winning strategies for the efficient and appropriate development of artificial languages. This is because they spend smaller time-steps to complete guessing games compared with the evolution of holistic languages and, at the same time, they obtain higher discrimination and consistency of utterances as well as higher coverage of topics compared with other topic selections in the evolution of compositional language. "Common" topic selection can be chosen if consistency is preferred while "random" topic selection may be adopted if the capability of distinguishing topics is preferred. Therefore, compositional languages are more effective and efficient than holistic languages for evolving artificial languages.

4.2 Compositional Representation of Rectilinear Relation

To find useful methods of representing compositional geometrical shapes for the evolution of languages in creative design, a hybrid computational model composed of Holographic Reduced Representations (HRRs) (Plate, 1995) and Self-Organising Map (SOM) (Kononen, 1990) are developed and tested. A similar model was used successfully by Levy and Kirby (2006) in their experiment to develop regular mappings between meanings and sequences.

Geometrical relations can be represented using associative representations, and be explored and enriched via machine learning to find new geometric relations based on the matched "utterances" with basic geometric relations. Such hybrid system could become an important part of the "brain" of a curious agent playing language games with other agents in a multi-agent environment to develop artificial languages based on the Domain-Individual-Field-Interaction framework (Csikszentmihalyi, 1999).

4.2.1 Subjects

To simplify the experiments and focus on testing the possibility of transformation between geometric relations and compositional utterances, rectilinear areas[1] (two dimensions), rather than rectilinear volumes (three dimensions) are selected as the experimental subjects (see Fig. 4.8).

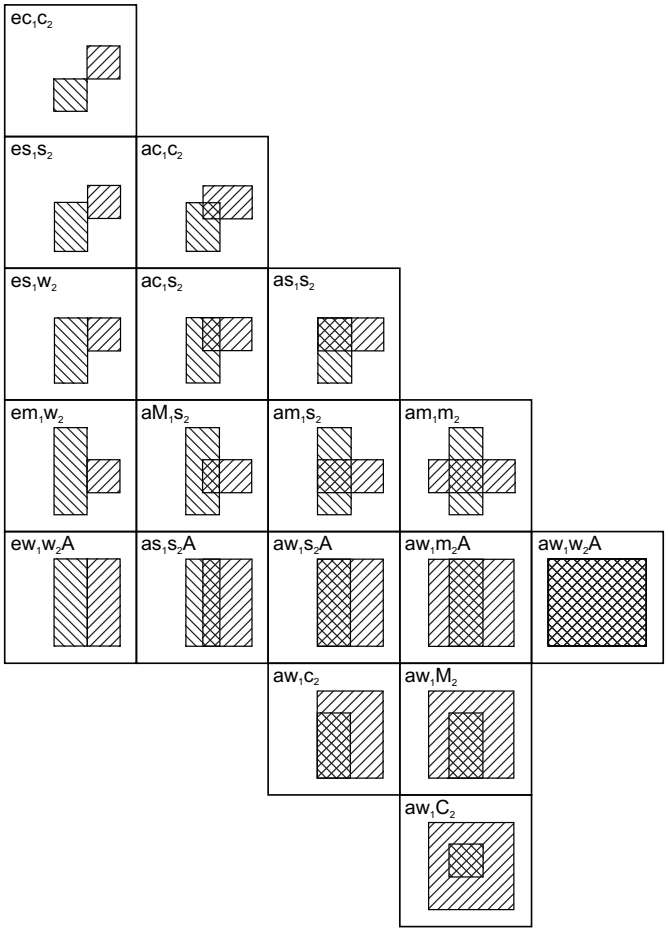


FIGURE 4.8: The relations between two rectangles

4.2.2 Implementation

Holographic Reduced Representations (HRRs) and the artificial neural network, Self-Organising Map (SOM) are utilised to implement the transformation from geometric representations to generative utterances.

In the computational experiments of rectilinear areas, the default setting for each HRR vector is the composition of 1024 (32×32) random numbers between 0 and 1. Both geometric meanings and symbolic sequences are represented by high dimensional vectors

[1] it is simplified from Rectilinear Volumes (Hannah, 2002) which is a classical design exercise for organising boxlike geometries to realize various relationships between multiple forms that is very helpful for novice students mastering the skills of space design.

of real numbers generated using circular convolution. The geometric meanings are the topological relations of two rectangles (see Equation 14) such as intersection or touch while the symbolic sequences are the compositions of simple utterances (see Equation 15).

$$R(r_1, r_2) = s_g + r_1 \otimes s_{p1} + r_2 \otimes s_{p2} + s_a \tag{14}$$

In Equation 14, r_1 and r_2 are the HRR vectors of two rectangles. Their relationship $R(r_1, r_2)$ is represented with the combination of three attributes including s_g, s_p and s_a. s_g means sharing certain geometry such as edge or area; s_p denotes sharing at least part of each rectangle such as corner, side or a middle part; and s_a identifies whether the two rectangles share an axis. $+$ is the addition of vectors while $\otimes$ is the circular convolution of a shape and its shared part.

$$U = C1_i \otimes V1_1 + C1_j \otimes V1_2 + C2_m + V2_n \tag{15}$$

In Equation 15, U denotes a compositional utterance, which is combined with two sets of consonants ($C1$, $C2$) and two sets of vowels ($V1$, $V2$). Totally 144 utterances are generated using this equation (also see Algorithm 9). These utterance-vectors are used as the neurons of a Self-Organising Map with 12×12 network, which will be mapped to the geometric-vectors, i.e., the HRRs of the geometric relationships between two rectangles $R(r_1, r_2)$.

The utterance-vectors and geometric-vectors are matched using SOM (see Algorithm 7, 8 and Fig. 4.9). The initial settings of SOM are described as follows: initial radius $r_0 = \sqrt{n_{neurons}}/2$, initial learning rate $l_0 = 0.9$, $epochs = 40$, and $iterations_{epoch} = 400$. Among them, $n_{neurons}$ is the number of neurons, i.e., units, used in SOM. They are 144 HRRs of compositional utterances distributed in a 12×12 network. After running SOM, the network develops systematically regular mapping between the utterances and the meanings of rectilinear relations (see Fig. 4.14).

4.2.3 Results

Through this experiment, the topological relations of two rectangles (see Fig. 4.8) are successfully categorised via HRRs. Following HRRs, the representations of rectangular relations are successfully matched with a number of HRR-utterances via SOM.

Algorithm 7 Self-Organising Map

$samples \leftarrow 18$ HRR $geometricVectors$
$units \leftarrow 144$ HRR $utteranceVectors$
$r_0 \leftarrow \sqrt{144}/2$
$initialLearningRate \leftarrow 0.9$
$categories_{recent} \leftarrow range(5)$
$time_{constant} \leftarrow epochs/\ln(r_0)$
$iterations_{epoch} \leftarrow 400$
$epochs \leftarrow 40$
$epoch \leftarrow 0$
do
    $epoch \leftarrow epoch + 1$
    $radius \leftarrow r_0 \times \exp(-epoch/time_{constant})$
    $learningRate \leftarrow initialLearningRate \times \exp(-epoch/(epochs - epoch))$
    for $i = 1 \rightarrow iterations_{epoch}$ **do**
        $sample \leftarrow randomChoice(samples)$
        $bmu \leftarrow bestMatchingUnit(units,\ sample)$
        for $unit$ in $units$ **do**
            $distance \leftarrow EuclideanDistance(unit,\ bmu)$
            if $distance < radius$ **then**
                $influence \leftarrow exp(-distance/(2 \times radius))$
                for $j = 1 \rightarrow length(unit)$ **do**
                    $unit_j \leftarrow unit_j + influence \times learningRate \times (sample_j - unit_j)$
                end for
            end if
        end for
    end for
    $categories_{recent} \leftarrow rest(categories_{recent})$
    $newCategories \leftarrow map(sample \rightarrow bestMatchingUnit(units,\ sample),\ samples)$
    $categories_{recent} \leftarrow append(categories_{recent},\ newCategories)$
while $(epoch < epochs) \wedge (length(removeDuplicates(categories_{recent})) \neq 1)$

Algorithm 8 Finding Best Matching Unit

function BESTMATCHINGUNIT($units,\ sample$)
    $bmu \leftarrow units_1$
    $distance0 \leftarrow EuclideanDistance(units_1,\ sample)$
    for $i = 2 \rightarrow length(units)$ **do**
        $distance \leftarrow EuclideanDistance(units_i,\ sample)$
        if $distance < distance0$ **then**
            $distance0 \leftarrow distance$
            $bmu \leftarrow units_i$
        end if
    end for
    return bmu
end function

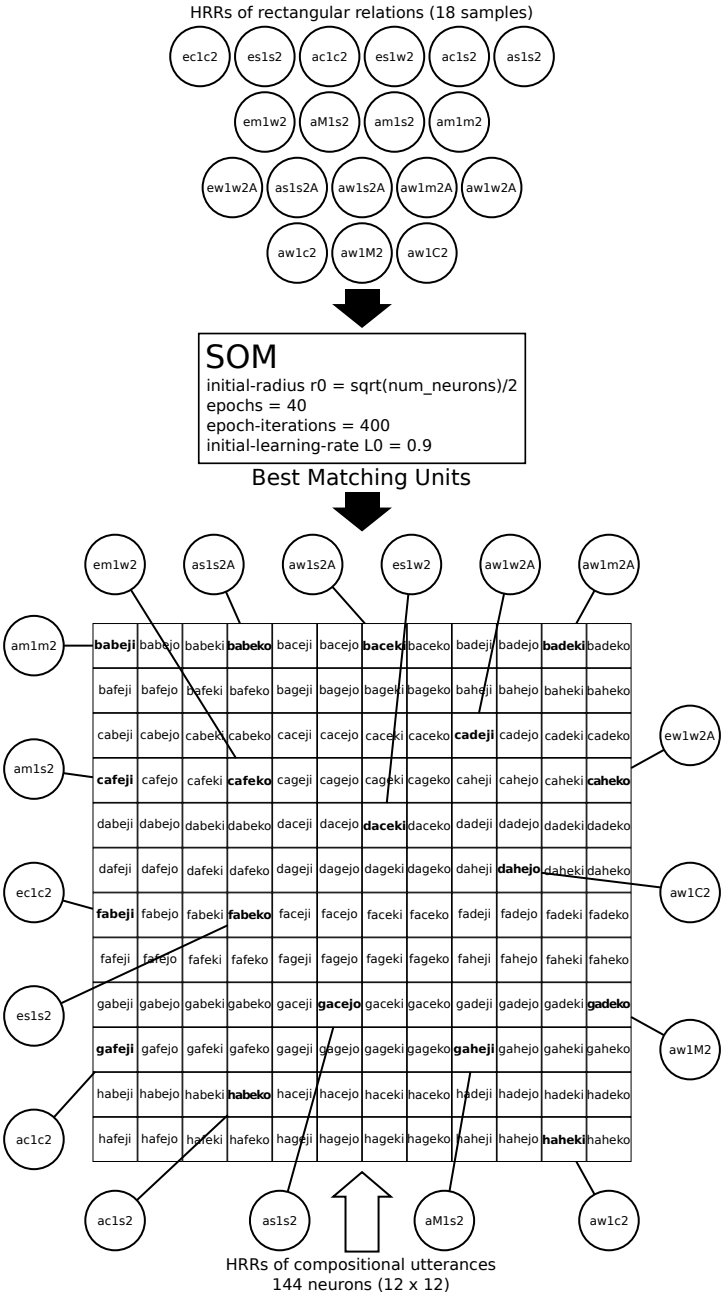


FIGURE 4.9: Mapping utterances to geometric relations using Self-Organising Map

4.2.3.1 Representations of Rectangular Relations

The topological relations between two rectangles are defined by three features: (1) the types of geometry including edge and area shared by two rectangles, (2) different parts including corner, side, middle, mid-side, centre and the whole shared with them (see Fig. 4.10), and (3) the axis shared by some rectangular relations (see Fig. 4.11). To begin with, each general feature such as shape, share-geometry, share-part and share-axis is represented as a random vector with 1024 (32×32) dimensions (see Table 4.8 and Fig. 4.12). Secondly, detailed elements are generated via addition or circular convolution with these general features (see Table 4.9 and Fig. 4.12). Then, 18 rectangular relations

are represented by combining these elements via addition and circular convolution of HRRs (see Table 4.10).

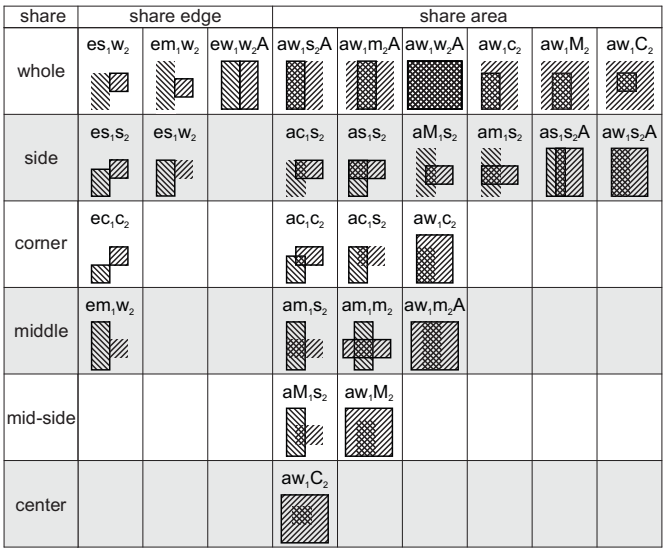


FIGURE 4.10: Catalog of sharing edge and area

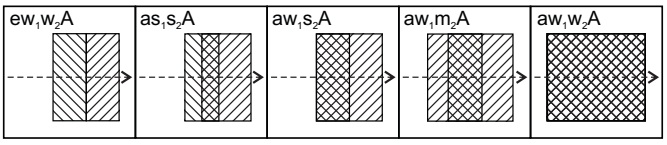


FIGURE 4.11: Share axis

TABLE 4.8: Holographic reduced representations (HRRs) of general features of rectangular relations

Shape	HRR("shape", random())
shareGeometry	HRR("geometry", random())
sharePart	HRR("part", random())
shareAxis	HRR("axis", random())

Each name of these rectangular relations is generated with three or four characters. The first character is selected between "e" and "a"; "e" means two rectangles share at least part of edge while "a" means they share at least part of area. The second and third characters are selected from "c", "s", "m", "M", "C" and "w". "c" means corner; "s" means side; "m" means middle; "M", which means partial middle (mid-side), is only for sharing area (see the vertical brick of aM_1s_2 compared with that of am_1s_2 at the centre of Fig. 4.8); "C" means centre; and "w" means whole. For example, aw_1C_2 means one rectangle shares its whole area while another only shares its centre (see the bottom relation in Fig. 4.8); ec_1c_2 means the two shapes both share a "corner" of their edges while ac_1c_2 means the two shapes both share a corner of their areas (see the top relations in Fig. 4.8). Some names have the fourth character, "A", which means two rectangles share at least one axis regardless of change in their sizes or in the ratio of width to height (see Fig. 4.11). In Table 4.10, "axis(0)" means it is not essential to

TABLE 4.9: Holographic reduced representations (HRRs) of detailed elements of rectangular relations

Shape	shape 1 = shape + HRR("shape1", random())
	shape 2 = shape + HRR("shape2", random())
ShareGeometry	edge = shareGeometry ⊗ HRR("edge", random())
	area = shareGeometry ⊗ HRR("area", random())
SharePart	corner = sharePart ⊗ HRR("corner", random())
	side = sharePart ⊗ HRR("side", random())
	middle = sharePart ⊗ HRR("middle", random())
	midside = sharePart ⊗ HRR("midSide", random())
	center = sharePart ⊗ HRR("center", random())
	whole = sharePart ⊗ HRR("whole", random())
ShareAxis	axis(0) = shareAxis ⊗ HRR("axis(off)", random())
	axis(1) = shareAxis ⊗ HRR("axis(on)", random())

share axis, or no axis is shared while "axis(1)" indicates that at least one axis should be shared.

TABLE 4.10: Holographic reduced representations (HRRs) of the relations between two rectangles

Name	HRRs
ec_1c_2	(edge + shape1 ⊗ corner + shape2 ⊗ corner) + axis(0)
es_1s_2	(edge + shape1 ⊗ side + shape2 ⊗ side) + axis(0)
ac_1c_2	(area + shape1 ⊗ corner + shape2 ⊗ corner) + axis(0)
es_1w_2	(edge + shape1 ⊗ side + shape2 ⊗ whole) + axis(0)
ac_1s_2	(area + shape1 ⊗ corner + shape2 ⊗ side) + axis(0)
as_1s_2	(area + shape1 ⊗ side + shape2 ⊗ side) + axis(0)
em_1w_2	(edge + shape1 ⊗ middle + shape2 ⊗ whole) + axis(0)
aM_1s_2	(area + shape1 ⊗ midside + shape2 ⊗ side) + axis(0)
am_1s_2	(area + shape1 ⊗ middle + shape2 ⊗ side) + axis(0)
am_1m_2	(area + shape1 ⊗ middle + shape2 ⊗ middle) + axis(0)
ew_1w_2A	(edge + shape1 ⊗ whole + shape2 ⊗ whole) + axis(1)
as_1s_2A	(area + shape1 ⊗ side + shape2 ⊗ side) + axis(1)
aw_1s_2A	(area + shape1 ⊗ whole + shape2 ⊗ side) + axis(1)
aw_1m_2A	(area + shape1 ⊗ whole + shape2 ⊗ middle) + axis(1)
aw_1w_2A	(area + shape1 ⊗ whole + shape2 ⊗ whole) + axis(1)
aw_1c_2	(area + shape1 ⊗ whole + shape2 ⊗ corner) + axis(0)
aw_1M_2	(area + shape1 ⊗ whole + shape2 ⊗ midside) + axis(0)
aw_1C_2	(area + shape1 ⊗ whole + shape2 ⊗ center) + axis(0)

To clarify the process of HRRs for representing rectangular relations, a tree structure of the representation of the rectangular relation named ac_1s_2 is illustrated in Fig. 4.12. As can be seen, the process consists of three steps. The first step is to generate general features shown on level 1. Each feature is represented by a list of random numbers with 32×32 dimensions. The second step is to produce elements shown on level 3 by combining relevant features and some new lists of random numbers shown on level 2. For example, "area" is generated via the convolution of "shareGeometry" and a

new list of random numbers named "area_". This convolution and another convolution of "shareGeometry" and some other new list for "edge" can be used to clarify the relationships between "shareGeometry", "area" and "edge". Among these, "area" and "edge" are two types of geometric share. In this example, area instead of only edge is shared between two rectangles. So "area" is selected to represent this relation. After all elements are generated, the third step is to select appropriate elements and combine them into one list of numbers with the same dimensions. In this example, "corner" is combined with "shape1" to produce "corner1 (c_1)" while "side" is combined with "shape2" to generate "side2 (s_2)". Then "corner1 (c_1)" and "side2 (s_2)" are combined with "area (a)" to generate a new list, which is combined with "axis(0)" to produce the final list named "ac_1s_2". Here, "A" is not added into this name because two rectangles do not share axis, i.e., axis(0), in this relation.

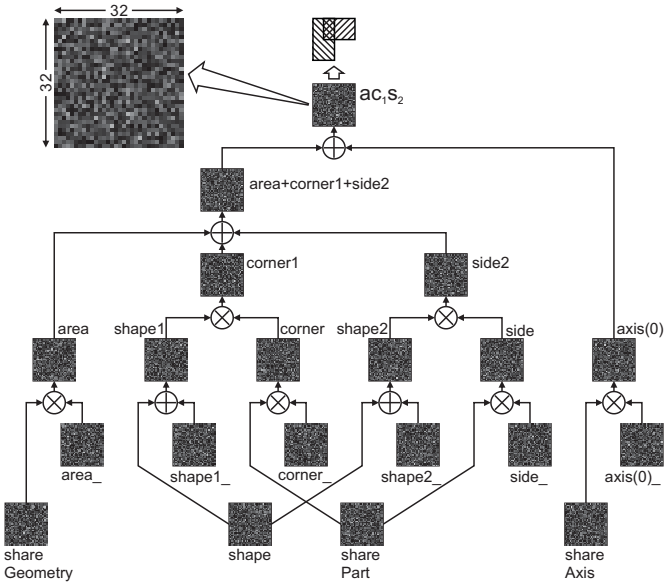


FIGURE 4.12: An example of the process of Holographic reduced representations (HRRs)

4.2.3.2 Results of Representations

The results of HRRs are clarified by using cosine similarity to analyse the difference between 18 rectangular relations. The similarities of these relations are shown in Fig. 4.13. The similarity between two relations is greater when the colour is lighter. The difference between each is clarified in relation to the range of similarities from 0.19 to 1.00 covering 81% of the whole possible distributed area. Therefore, the relations between two rectangles are represented clearly and successfully using HRRs.

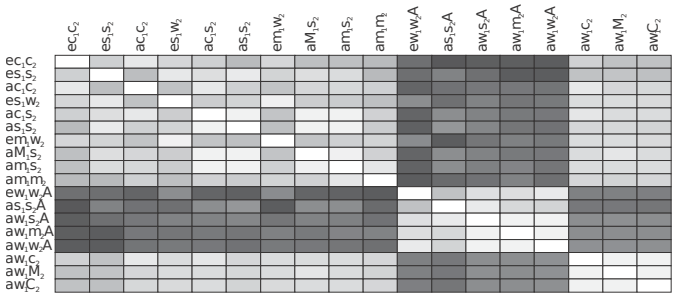


FIGURE 4.13: Cosine similarities between relations of two rectangles

4.2.3.3 Mapping Utterances to Rectangular Relations

The utterances for mapping are also generated via HRRs. Each utterance such as "fabeji" is composed of six elements selected from different sets including C1("b", "c", "d", "f", "g", "h"), C2("j", "k"), V1("a", "e") and V2("i", "o"). By combining all of these elements ($6 \times 6 \times 2 \times 2$), 144 utterances are generated for SOM to map the representations of 18 rectangular relations. The procedure of generating HRR-utterances is described in Algorithm 9.

Algorithm 9 Generating HRR-utterances (units) for SOM

$C1 \leftarrow list(\text{“}b\text{”},\ \text{“}c\text{”},\ \text{“}d\text{”},\ \text{“}f\text{”},\ \text{“}g\text{”},\ \text{“}h\text{”})$
$C2 \leftarrow list(\text{“}j\text{”},\ \text{“}k\text{”})$
$V1 \leftarrow list(\text{“}a\text{”},\ \text{“}e\text{”})$
$V2 \leftarrow list(\text{“}i\text{”},\ \text{“}o\text{”})$
$units \leftarrow emptyList$
for $i = 1 \rightarrow C1.length$ **do**
  for $j = 1 \rightarrow C1.length$ **do**
    for $m = 1 \rightarrow C2.length$ **do**
      for $n = 1 \rightarrow V2.length$ **do**
        $newUnit \leftarrow C1_i \otimes V1_1 + C1_j \otimes V1_2 + C2_m + V2_n$
        $units \leftarrow append(units,\ newUnit)$
      end for
    end for
  end for
end for

In the process of mapping, 49 epochs are implemented. Each epoch includes 400 iterations. As illustrated in Fig. 4.14, the process can be divided into three stages. The first stage is from epoch 1 to 17. In this stage, the mapping between relations and utterances is unstable in terms of extreme change of mapping from one epoch to another epoch. The average success rate is lower than 1.3%. In addition, an average of 16% of relations is not distinguished from others as shown with the degrees of difference; i.e., almost three relations are mapped with the same utterance in each epoch. In the second stage from epoch 18 to 26, the success rate increases sharply from 5.6% to 100%; and each relation is mapped to a different utterance. The last stage is from epoch 27 to 49. In

this stage, the mapping between relations and utterances becomes stable. No mapping is changed and each relation is mapped to the final appropriate utterance.

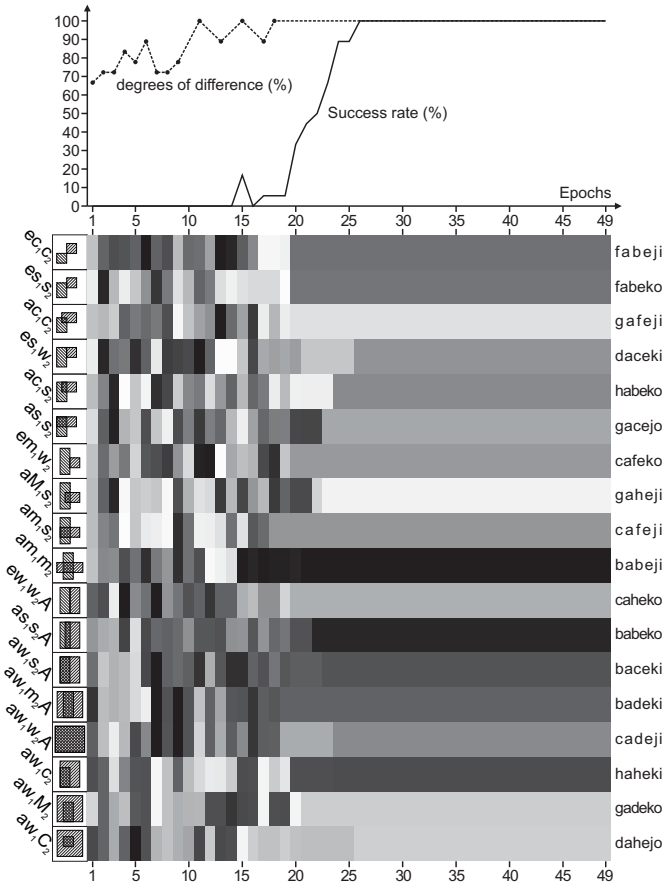


FIGURE 4.14: The process of mapping utterances to rectangular relations

4.2.3.4 Results of Mapping

Most results of mapping are obtained before half of the total epochs are executed (see Table. 4.11). The average number of epochs for obtaining results is 22 which accounts for 45% of the total epochs. In addition, The representation of each relation is distinguished by a different utterance in the early stage, i.e., the 18th epoch occurring 37% of the total epochs (see degrees of difference in Fig. 4.14). Initially, random utterances are selected to represent these relations. An utterance might represent more than one relation. After running 18 times, an utterance only represents one relation; other relations are represented by other utterances with higher associated weight. Therefore, The mapping of compositional utterances and geometric relations is realised using Self-Organising Map.

TABLE 4.11: The number of epochs when obtaining each final mapping result

Rectangular relation	Utterance	Epochs (total: 49)
ec_1c_2	fabeji	20
es_1s_2	fabeko	20
ac_1c_2	gafeji	20
es_1w_2	daceki	26
ac_1s_2	habeko	24
as_1s_2	gacejo	23
em_1w_2	cafeko	20
aM_1s_2	gaheji	23
am_1s_2	cafeji	18
am_1m_2	babeji	21
ew_1w_2A	caheko	20
as_1s_2A	babeko	22
aw_1s_2A	baceki	23
aw_1m_2A	badeki	24
aw_1w_2A	cadeji	24
aw_1c_2	haheki	24
aw_1M_2	gadeko	21
aw_1C_2	dahejo	26

4.2.4 Conclusion

The mappings between rectangular relations and artificial utterances are completed by using the hybrid system integrating HRRs and SOM. The success is mainly contributed by the compact structure of the HRRs of both geometric relations and utterances as well as the mapping between them using SOM. Each geometric relation is represented by two circular convolutions and three additions (see Table 4.10) while each utterance (unit) is generated with similar structure to that of rectangular relation. The total of utterances is 144, which is enough for matching appropriate units labelled with utterances to the 18 relative relations of two rectangles. In brief, the results of this experiment suggest the possibility of transforming between artificial languages and design concepts via the hybrid system within associative memories and artificial neural networks.

4.3 Compositional Language for Shape Combination

The aim is to evolve compositional languages for generating novel shapes based on original shapes via several operations such as “touch”, “overlap” and “contain/in”. If agents achieve 70% success rate on the representations of these types of connection, the agents will enter next stage to generate new designs guided by the generation of new compositional utterances based on previous evolved languages.

4.3.1 Subjects

The subjects of this experiment are regular shapes including triangle, square, hexagon, circle and star.

4.3.2 Implementation

This experiment is emphasised on the composition of utterances representing the same dimension that differs from those in the experiment, Compositional and Holistic Language (see Section 4.1), which is based on the combination of words denoting different dimensions including colour, shape and size. New interesting meanings can also be generated by combining utterances associated with different meanings belonging to the same dimension. For example, "roundish square" is the combination of "circle" and "square" in the same dimension, shape.

This experiment is implemented using both guessing games and generation games based on the DIFI framework (see Fig. 4.15). Firstly, primary shared utterances representing the compositions of shapes evolve in guessing games. Then these utterances are recombined to new utterances representing new interesting compositions of shapes.

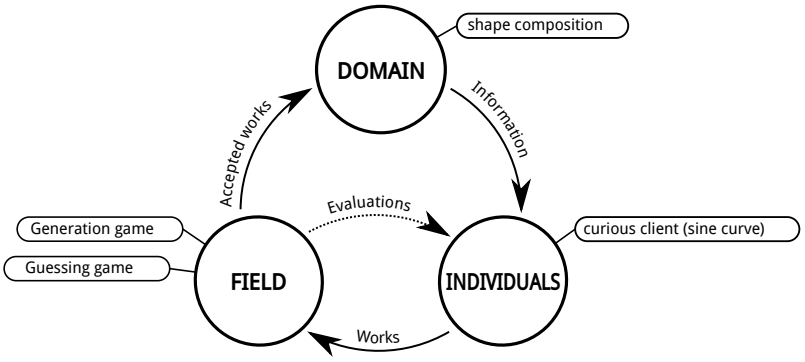


FIGURE 4.15: The Domain-Individual-Field-Interaction (DIFI) framework for generating compositional shapes

Two main rules including deformation and combination (see Rule 5 and Rule 6 in Table 4.12) can be used to generate new compositional works. These works are new shapes such as squished circle and roundish square, or new relations such as the intersection between circle and rectangle.

4.3.2.1 Deformation of Shapes

A polygon can be deformed to a closed string composed of continuous connected Bezier curves using Equations 16, 17, 18, 19, and 20 (see Fig. 4.16). The curve can be adjusted by changing the values, n_t and n_c. Given $n_t = 2$, if $n_c = 0.2$, it will be twisted; if

TABLE 4.12: The rules of generating compositional shapes

No.	Rule	Expansion	Example
0	new-work	operator shape shape	
1	operator	deform / Boolean-operation	
2	shape	triangle / square / hexagon / circle / star	
3	deform	change-Bezier-curve	
4	Boolean-operation	union / intersection / difference	
5	new-shape	deform shape shape	rounded rectangle
6	new-relation	topological-relate shape shape	circle intersecting rectangle

$n_c = 0.8$, it would be similar to the original polygon although each corner is smoothed; if $n_c = 1.8$, it will become very rounded (see Fig. 4.17).

$$f(a, b, n) = a + (b - a)/n \tag{16}$$

$$x_{t0}, y_{t0} = f(x_0, x_1, n_t), f(y_0, y_1, n_t) \tag{17}$$

$$x_{t1}, y_{t1} = f(x_1, x_2, n_t), f(y_1, y_2, n_t) \tag{18}$$

$$x_{c0}, y_{c0} = f(x_{t0}, x_1, n_c), f(y_{t0}, y_1, n_c) \tag{19}$$

$$x_{c1}, y_{c1} = f(x_{t1}, x_1, n_c), f(y_{t1}, y_1, n_c) \tag{20}$$

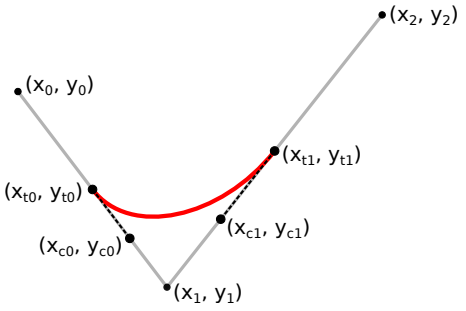


FIGURE 4.16: Bezier curve

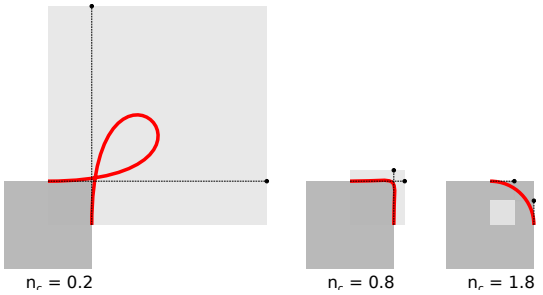


FIGURE 4.17: The mechanism of shape deformation ($n_t = 2$)

The samples of deformed simple shapes can be seen from Fig. 4.18.

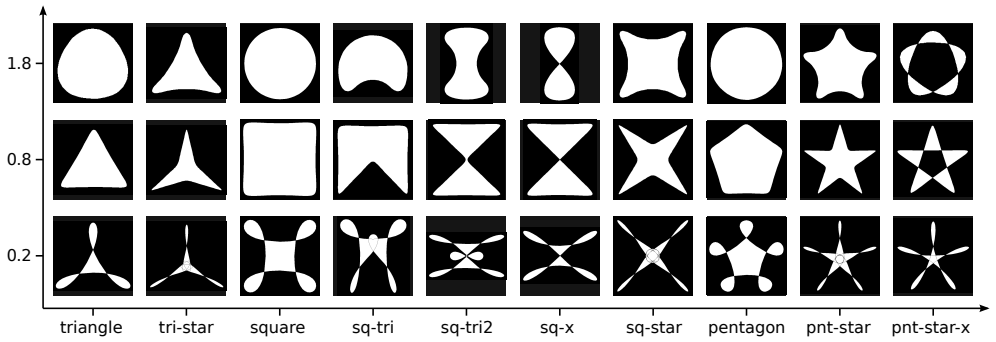


FIGURE 4.18: Shape deformation ($n_t = 2, n_c = 0.2, 0.8, 1.8$)

4.3.2.2 Combination of Shapes

New compositional shapes can be generated using Boolean (union, intersection and difference) operation on two shapes connected with different edges or branches. For example, "triangle-hexagon [0, 2][1, 0] i" is the intersection of two shapes connected with the edge(0,2) of a triangle and the edge(1,0) of a hexagon (see Fig. 4.19). Each shape has a centre node connecting several terminal nodes with edges and branches (see Table 4.13).

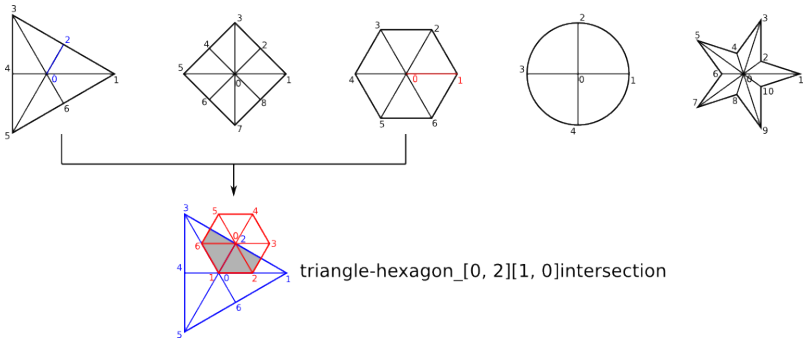


FIGURE 4.19: An example of combining shapes

TABLE 4.13: The attribute settings of regular shapes

Shape	Edges	Branches	Branch-lengths	Terminals
Triangle	3	6	2	3/6
Square	4	8	2	4/8
Hexagon	6	6	1	6/6
Circle	1	4	1	1/4
Star	10	10	2	10/10

Therefore, there are two steps in implementing of the combination of shapes. The first step is to connect the shapes by selecting and joining relevant nodes (centres, terminals), branches and edges. For examples, the centres of two shapes are joined; one centre and one terminal are joined; one edge/branch is joined with another edge/branch; or one branch/edge is joined with another edge/branch. Geometric transformation including rotation, scaling and moving are used to match two points with different coordinates or

two edges/branches of different lengths. The second setp is to operate on the connected shapes using intersection, difference or union.

Besides the combination of shapes described above, single shape can also be operated on via addition, displacement and subtraction (Di Mari, 2013). For example, a "trianglish-quadrangle" could be produced by cutting off the small top of a triangle. It may also become a star-like shape by changing every edge of a quadrangle to two edges of a triangle with acute angle.

4.3.2.3 Evaluation of Generated Shapes

The variation of the Wundt curve, Sine curve, is used by client-agents to evaluate the novelty of the compositional shapes generated by designer-agents. To study the impact of similarity-preference and selection-threshold (i.e., tolerance) on collecting design works, different combinations of similarity-preference and selection-threshold are tested (see Table 4.16, 4.17, 4.18).

The mechanism for measuring the difference between two shapes is the use of contour functions (Kindratenko, 1997). OpenCV is adopted to measure the similarities of contours (Bradski et al., 2000). The proportion of two combined shapes, i.e., distinguishing the primary shape from the secondary shape, is not considered in this experiment.

4.3.2.4 Guessing Game for Initialising Compositional Utterance

The process of guessing game is the same as that (see Fig. 4.3) implemented in the experiment, Compositional and Holistic Language. Here, only random topic selection is used; only compositional languages evolve; and only the evolved representations of shapes are used in generation games. The evolved utterances representing shapes through guessing games can be seen from Table 4.14. The first agent, who could plays the role of client in generation games, uses the same utterance"v" to represent both hexagon and circle while others use "c" to represent hexagon. The third agent uses "k" representing square while others use "w". These ambiguities may affect the results of this experiment for testing consistency although it makes the simulation more realistic.

TABLE 4.14: The results of guessing games for combining shapes

Agent	1	2	3	4	5	6
SHAPE	a	a	a	a	a	a
Triangle	y	y	y	y	y	y
Square	w	w	**k**	w	w	w
Hexagon	**v**	c	c	c	c	c
Circle	v	v	v	v	v	v
Star	p	p	p	p	p	p

4.3.2.5 Generation Game for Generating Compositional Shapes

In each simulation of generation games, the generation runs are set to 1000; population is 6; and only the first agent is taken as client. The operation of original shapes is mainly combining two regular shapes using union, intersection and difference. These Boolean-types are measured but not represented that is for evaluating the creativity of compositional shapes related with the same utterance (e.g. triangle-square (union 0.3, intersection 0.1, difference 0.5) → "ya-wa").

The process of the generation game is illustrated in Figure 4.20. First, client-agent selects a pair of shapes randomly, then maps it to a relevant compositional utterance (i.e., design brief) and compositional object (i.e., prototype). Designer-agents start to parse the design brief to pair-shapes and generate new compositional objects as designs matching the pair-shapes. Then client-agent selects the winning-design from these new designs. If there is a winning-design, both the client-agent and the designer-agents strengthen the association between the winning-design and design brief. At the same time, the designer-agents who fail will weaken the associations between their own designs and client-agent's design brief. If there is no winning-design, all designer-agents will weaken the associations between their own designs and client-agent's design brief.

The process of selecting winning-design by comparing client-agent's prototype and designer-agents' works is as follows: measuring similarity between prototype and designs with their contours → measuring interestingness of designs using sine curve → a design becomes a candidate if the score of its creativity is equal to or greater than the selection-threshold → selecting the design with the highest score as the winning-design if there are candidates, otherwise selecting nothing, i.e., no winning-design.

4.3.3 Results

At the completion of the generation games, information on the number of combined shapes is stored in the client-agent's association memory. As can be seen from Table 4.15 and Fig. 4.21, the client-agent, whose selection threshold is 0.7 and preference of difference is 60%, collects 52 compo-shapes with a total of 25 types of combinations in a simulation. In Table 4.15 the operation of union is represented as "u", intersection is represented as "i" and difference is represented as "d".

Client-agent's different selection preferences and selection thresholds affect the success rates of language games, and the collection of both designs and design types.

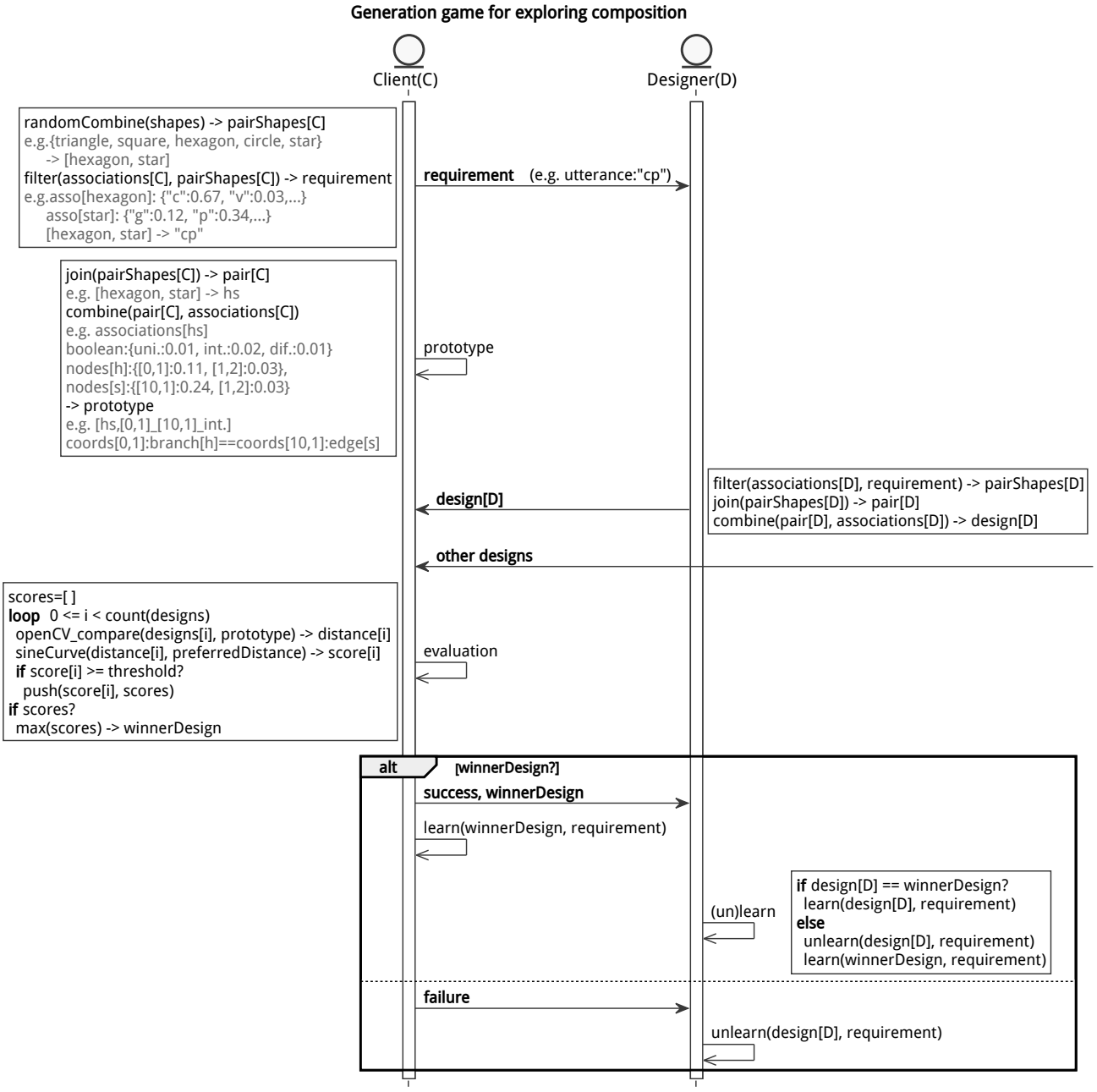


FIGURE 4.20: Generation game of compositional language

4.3.3.1 Success Rates

Results show that it is easy to be successful when the client-agent selects a design that is similar to its own prototype (see Table 4.16 and Fig. 4.22). The success rate surpasses 85% when the client-agent prefers to selecting the design that does not differ from its own prototype or only differs by around 20% after the generation game has been played 1000 times with both 0.7 and 0.9 selection threshold, although the former's success rate is higher than the latter's because of its greater tolerance. The success rate is below 60% when client's favourite design is more than 60% different from his own prototype

TABLE 4.15: Client's collected compositional shapes in a simulation (u: union, i: intersection, d: difference)

Compo-shapes	Utterance	Artwork 1	Artwork 2	Artwork 3	Artwork 4
triangle-triangle	"yaya"	[1, 2][4, 1]u			
triangle-square	"yawa"	[1, 4][1, 2]d	[1, 0][6, 2]i	[4, 1][1, 8]d	[4, 2][1, 4]i
triangle-hexagon	"yava"	[0, 2][1, 0]i	[6, 1][0, 1]i	[2, 1][0, 1]d	
triangle-circle	"yava"	[1, 0][1, 3]u			
triangle-star	"yapa"	[0, 2][1, 6]u			
square-triangle	"waya"	[1, 6][1, 0]d	[4, 2][2, 1]u		
square-square	"wawa"	[2, 6][1, 4]i	[0, 1][1, 4]u		
square-hexagon	"wava"	[4, 1][1, 2]u	[4, 1][1, 0]i		
square-circle	"wava"	[1, 0][1, 3]d	[4, 2][3, 1]d	[1, 8][0, 1]u	
square-star	"wapa"	[6, 1][2, 1]d	[1, 2][1, 4]d		
hexagon-triangle	"vaya"	[1, 2][4, 2]i	[1, 4][1, 3]i	[2, 1][0, 1]d	
hexagon-square	"vawa"	[3, 1][1, 0]d	[1, 2][0, 1]i		
hexagon-hexagon	"vava"	[1, 3][0, 1]d			
hexagon-circle	"vava"	[1, 4][3, 1]u	[2, 1][1, 2]d		
hexagon-star	"vapa"	[1, 4][2, 6]u	[0, 1][1,10]u		
circle-triangle	"vaya"	[2, 1][1, 3]i			
circle-square	"vawa"	[1, 2][2, 0]u			
circle-hexagon	"vava"	[1, 2][3, 1]i	[1, 2][1, 0]u	[3, 1][1, 2]u	
circle-circle	"vava"	[0, 1][2, 1]d			
circle-star	"vapa"	[0, 1][2, 0]i			
star-triangle	"paya"	[10,1][1, 4]i	[2, 6][2, 1]i	[1,10][1, 6]d	[1, 0][1, 6]i
star-square	"pawa"	[1, 8][2, 1]d	[1, 5][4, 1]u	[0, 1][1, 0]u	
star-hexagon	"pava"	[2, 0][1, 2]i	[10,1][3, 1]d		
star-circle	"pava"	[1, 6][0, 1]u	[1, 8][1, 2]i	[4, 2][1, 0]d	
star-star	"papa"	[1, 5][2, 1]i	[5, 1][3, 1]i		

with 0.7 selection threshold and when his favourite design is only more than 40% different from his own with 0.9 selection threshold. This is because the designer-agents in this experiment only generate designs matching the client-agent's requirement based on their previous experiences, which result in similar designs rather than quite different ones. Therefore, even multiple designer-agents with considerably different designs cannot satisfy the client-agent's requirement, which is "too" novel. This could change if the designer-agents can catch up with the client-agent's preference for similarity then adjust their designs to match the difference.

4.3.3.2 Different Evaluations of Collecting Shapes

Results (see Table 4.17 and Fig. 4.23) show that the client-agent collects more designs when its difference preference is between 0.2 and 0.6. By contrast, when its difference preference is very low, such as 0, or very high, such as 0.8 and 1, it collects a comparably small number of designs. This is because if it only selects the design that is the same as its own prediction, there would be no further change in its association memory and no

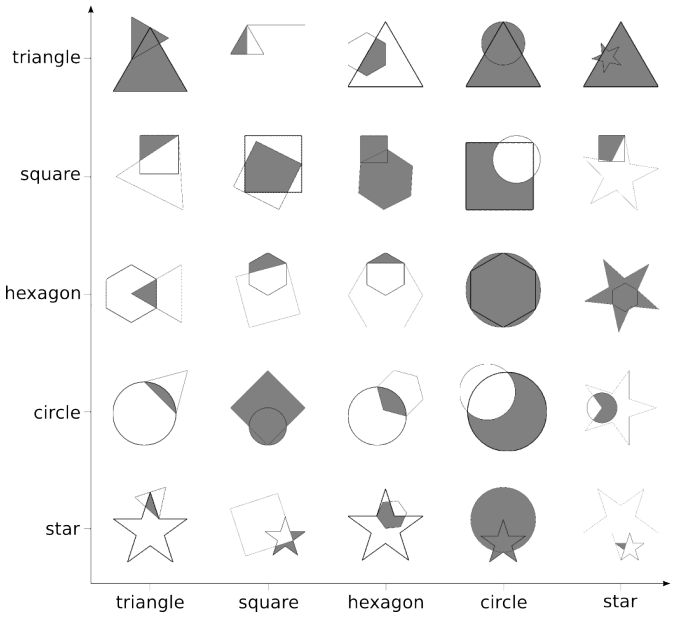


FIGURE 4.21: A result of client's collected compositional shapes

TABLE 4.16: Success rates

Index	SelectThre.	Prefer	Sim1	Sim2	Sim3	Sim4	Sim5	Mean
1	0.7	0	0.99	0.925	0.94	0.947	0.976	0.956
2	0.7	0.2	0.994	0.993	0.983	1.0	1.0	0.994
3	0.7	0.4	0.807	0.824	0.807	0.756	0.688	0.776
4	0.7	0.6	0.762	0.608	0.58	0.563	0.59	0.621
5	0.7	0.8	0.461	0.68	0.516	0.506	0.56	0.545
6	0.7	1	0.53	0.648	0.442	0.676	0.532	0.566
7	0.9	0	0.852	0.923	0.93	0.804	0.913	0.884
8	0.9	0.2	0.901	0.885	0.835	0.914	0.771	0.861
9	0.9	0.4	0.681	0.69	0.774	0.616	0.461	0.644
10	0.9	0.6	0.483	0.367	0.538	0.482	0.405	0.455
11	0.9	0.8	0.486	0.404	0.342	0.505	0.6	0.467
12	0.9	1	0.365	0.526	0.617	0.29	0.309	0.421

other new designs will be collected. On the other hand, if it requires a design that is too novel and therefore surpasses the designer-agents' capability and the difference between each collected design work is too great, this will produce more designs that may be very similar to each other. Both of these factors (i.e., the client-agent's novelty requirement being very low or seriously high) may lead to fewer collections. Further, when the client-agent's selection threshold is lower, e.g. 0.7, it can collect more works with the same level of difference preference except for the couple of 0.9 selection threshold and 0.2 difference preference. The exception means that comparable higher selection threshold with considerably lower difference preference could result in more acceptable works from the designers in these generation games.

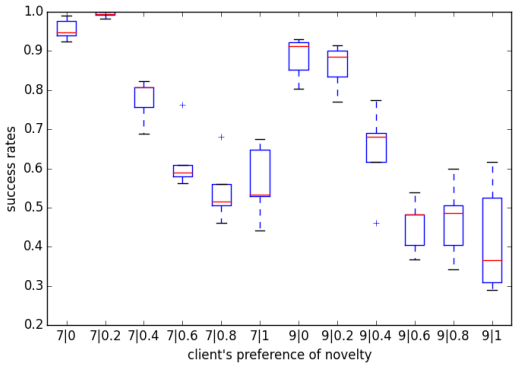


FIGURE 4.22: Success rates of generation games (x-axis e.g. 7/0 means selectionThreshold = 0.7, differencePreference = 0.0)

TABLE 4.17: The total number of compositional shapes in client's association memory

Index	SelectThre.	Prefer	Sim1	Sim2	Sim3	Sim4	Sim5	Mean
1	0.7	0	39	50	45	41	47	44
2	0.7	0.2	65	52	50	55	52	55
3	0.7	0.4	51	50	49	52	51	51
4	0.7	0.6	43	58	56	52	53	52
5	0.7	0.8	47	52	42	42	36	44
6	0.7	1	43	43	48	48	50	46
7	0.9	0	44	42	38	37	42	41
8	0.9	0.2	59	51	60	58	53	56
9	0.9	0.4	51	48	49	52	50	50
10	0.9	0.6	47	43	46	52	54	48
11	0.9	0.8	46	42	37	41	39	41
12	0.9	1	41	40	41	39	39	40

4.3.3.3 Different Evaluations of Collecting Types of Shapes

In this experiment, a total of 25 combinations of two types of shapes from among triangle, square, circle, hexagon and star are provided. As the results show (see Table 4.18 and Fig. 4.24), the client-agent covers an average 24 compositional shapes when using 0.9 selection threshold with 0.2 difference preference or 0.7 threshold with 0.6 preference. Although 0.7 threshold with 0.8 preference gets all the compositional shapes (totally 25) in a simulation, this couple of strategies is not stable. Only an average of 22 compositional shapes are collected when the client-agent uses 0.7 threshold with the preference of no difference or 0.9 threshold with 0.8 difference preference.

4.3.4 Conclusion

New designs of combining two shapes, i.e., the combinations in the same dimension, are successfully generated in the evolution of compositional languages. The results suggest

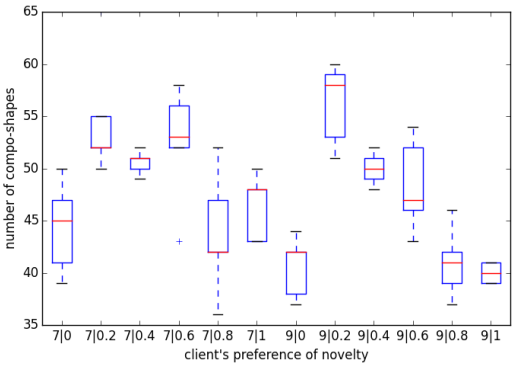


FIGURE 4.23: Client's collected compositional shapes through generation games (x-axis e.g. 7/0 means selectionThreshold = 0.7, differencePreference = 0.0)

TABLE 4.18: The total number of compo-shape-types in client's association memory

Index	SelectThre.	Prefer	Sim1	Sim2	Sim3	Sim4	Sim5	Mean
1	0.7	0	22	22	22	23	23	22
2	0.7	0.2	23	23	23	23	23	23
3	0.7	0.4	23	23	23	22	23	23
4	0.7	0.6	23	23	22	25	25	24
5	0.7	0.8	25	23	23	24	22	23
6	0.7	1	22	23	24	23	24	23
7	0.9	0	23	22	23	22	23	23
8	0.9	0.2	24	23	24	24	23	24
9	0.9	0.4	23	23	23	23	22	23
10	0.9	0.6	23	23	23	22	24	23
11	0.9	0.8	23	23	21	22	21	22
12	0.9	1	23	22	23	24	22	23

that the associations of items in the same dimension can produce creative designs and expand the design space when recombining the utterances representing the same or different attributes. In addition, the novelty principle, i.e., the production of similar-yet-different artefacts (Saunders, 2002), is realised in this experiment under the condition of the designer-agents generating designs only matching their previous experiences, i.e., the client-agent may collect more designs when its selection threshold is around medium rather than too low or too high.

4.4 Conclusion

The mechanism of composition is tested and discussed in this chapter. Results show that compositional languages can be more creative than holistic languages because the former is far more efficient and a little more ambiguous than the latter. Mapping hierarchical meanings to compositional utterances has been realised by using the combination of SOM and HRRs. The completion of an entire simulation of evolving compositional languages with guessing games and generation games to produce creative combinations of

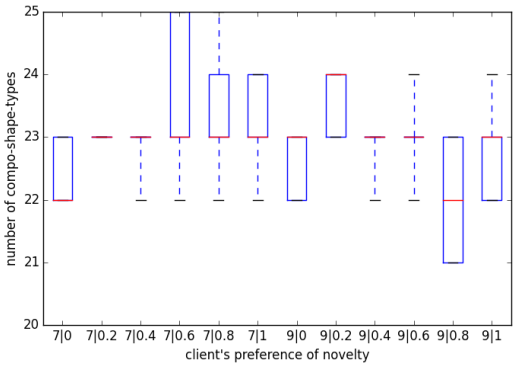


FIGURE 4.24: Client's collected compo-shape types through generation games (x-axis e.g. 7/0 means selectionThreshold = 0.7, differencePreference = 0.0)

shapes suggests the novelty principle, i.e., the production of similar-yet-different artefacts (Saunders, 2002), since a client-agent collects more interesting design works when its selection threshold is moderate rather than too low or too high.

Chapter 5

Exploring Creative Features of Language

To evaluate the utility and effectiveness of language and to find mechanisms for applying the creative features of language in designing, four main topics—ambiguity, scalability, incongruity and extensibility—are explored. At the completion of these experiments, the core features of grounded language for design are clarified and artificial compositional languages have evolved to facilitate both routine design such as parametric design and non-routine design in which design rules and principles are developed via the communications among curious agents.

5.1 Ambiguity

To test the ambiguity of compositional language for creative design, a many-to-many network of associations between topics and utterances, i.e., polysemy and synonymy, is evolved by multi-agents playing language games to generate interesting designs and interesting utterances.

5.1.1 Subjects

The combinations of 11 colours and 11 shapes represented with [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] in each respect are used in this experiment.

5.1.2 Implementation

This experiment is implemented by recombining existing utterances to generate new utterances representing interesting meanings through the communication between agents

who play the roles of speaker and listener in guessing games as well as client and designer in generation games (see Fig. 5.1).

The language games used in this experiment produce compositional utterances, which can be used to denote new concepts. For example, given the previous utterances "red triangle", "red square" and "blue triangle" on a four-quadrant diagram (x-axis: colour including red and blue, y-axis: shapes including triangle and square), new utterances such as "blue square" on the top-right quadrant can be generated by recombining the sub-utterances "blue" and "square".

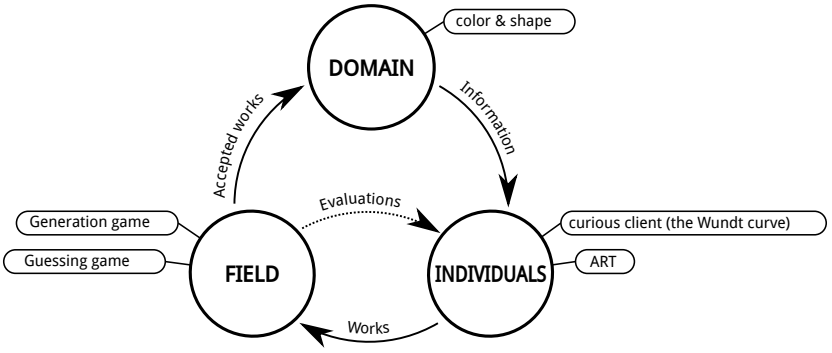


FIGURE 5.1: The Domain-Individual-Field-Interaction (DIFI) framework of exploring ambiguity

The agents in the simulation use the networks based on Adaptive Resonance Theory (ART) to categorise utterances and concepts. ART networks are both stable and dynamic; they can not only retain existing categories but also add new categories for unfamiliar inputs which exceed the threshold of existing neurons (Saunders, 2002). In language games, both rules and instances are stored in agents' memory. A rule is a dictionary that contains feature, utterances, category and weight, e.g. {feature="shape", utterance="a", category=0, weight=0.01}, whilst an instance is also a dictionary that contains prototype, utterance and frequency, e.g. {prototype=[0.1,0.3], utterance="ab", frequency=1}. A prototype is a combination of colour and shape, while frequency is the number of times that the same relation between a prototype and utterance is used and accepted.

The interest of design requirements (i.e., utterances) and design works are evaluated using the Wundt curve. This hedonic function can be used to explore the questions, "How confident is an agent of its prediction of the experience that is described by an utterance or provided with a design/art work?" and "How novel is the most interesting topic?"

5.1.3 Experiment Settings

5.1.3.1 Initial Settings

The first set of experiments is initialised with 50 samples randomly selected from 121 objects, which are generated by combining 11 colours and 11 shapes. Each object is represented by a pair of decimals belonging to [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]. The language game uses one combination rule combining two features including colour and shape. Each feature is represented by one character as its name. So the length of an utterance is limited to 2. For example, {colour:0.2, shape:0.3}'s utterance may be "ha". The population of agents is set to 6.

5.1.3.2 Settings of Guessing Game

In each guessing game, 2 agents are randomly selected from 6 agents to play the role of speaker and listener respectively. The context is composed of 8 topics randomly selected from the initialised 50 samples. When the success rate of guessing games is above 60%[1], guessing games are completed; and agents start to play generation games.

5.1.3.3 Settings of Generation Game

In generation games, four types of procedures by the client-agent are implemented with or without evaluation of the interestingness of requirements (i.e., utterances) or works using the Wundt curve (see Fig. 3.8). The default ratio of difference over similarity, i.e., the most interesting distance between novel design and normal design, is set to 0.364, which could be adjusted to be more conservative or progressive in the Wundt curve. Each cycle is repeated 1000 times. The last agent always plays the role of client while others play the role of designer.

5.1.4 Experiment Procedures

The experiment procedures consist of the procedure of guessing games for evolving a grounded language representing the compositions of colours and shapes and the procedures of generation games for generating interesting or normal requirements and designs.

[1]It is difficult to reach a higher success rate such as 70% possibly owing to the settings of the ART network. So the success threshold is set to 60% in this experiment.

5.1.4.1 Procedure of Guessing Games (Procedure 1)

Guessing games (see Fig. 5.2) are implemented repeatedly till the success rate reaches 60%. Each guessing game consists of 4 steps described as below.

Step 1: Speaker selects a topic randomly from randomly generated context.

Step 2: Speaker generates an utterance representing the selected topic and "tells" listener the utterance.

Step 3: Listener guesses the topic by exploring its stored associations between utterances and the ART categories. If an appropriate association cannot be found, a new association between the utterance and a topic in current context is generated. Then the listener "tells" the speaker its guess.

Step 4: If guessing succeeded, both speaker and listener increase the weight of their associations between the topic's ART category and the utterance, and increase the frequency of each related instance or generate a new association connecting the selected topic with the utterance. If guessing failed, the listener decreases the weight of the related association and generates a new association between the topic's ART category and the utterance, then increases the weight of the newly generated association and generates a new instance connecting the correct topic and the utterance.

After completing guessing games, the agents (Group A) are cloned three times to get three new groups of agents (Group B, Group C and Group D) to implement different procedures for generation games.

5.1.4.2 Procedures of Generation Games (Procedure 2, 3, 4, and 5)

The generation games (see Fig. 5.3) are simulated in four different situations: (1) without evaluating the interest of requirements (i.e., client-generated utterances) and designers' works by the client; (2) only evaluating the interest of designers' works by the client; (3) only evaluating the interest of requirements (i.e., client-generated utterances) by the client itself; and (4) evaluating the interest of both requirements and works by the client. These four situations are represented in the horizontal axis (see Fig. 5.5 and Fig. 5.6). So, there are four results from both the client and designers. Their average instances in the four situations are shown in Fig. 5.5, and the average max degrees of connections of their instances are shown in Fig. 5.6. The max degree of connections means the largest number of meanings represented by an utterance or the largest number of utterances representing a meaning. An agent's instances consist of all its own associations between utterances and meanings. For example, an agent's instances may be {"da":[colour 0.2, shape 0.4],"ku":[colour 0.7, shape 0.3],"gi":[colour 0.6, shape 0.5],...}. As can be seen from Fig. 5.5, the first column shows that the client's average

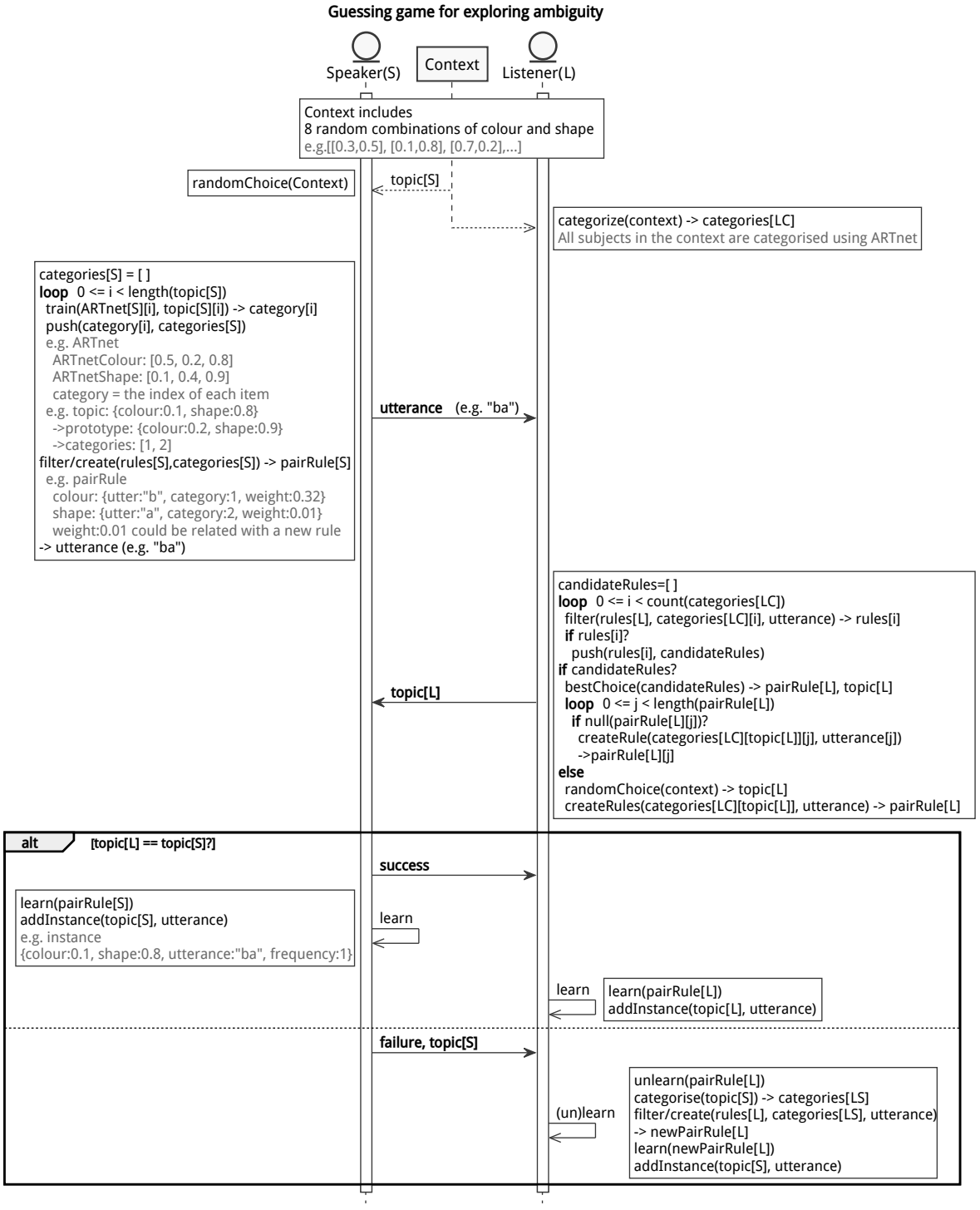


FIGURE 5.2: Guessing game for exploring ambiguity

number of instances is 364 without interest evaluation. The four different procedures of generation games are described below.

Procedure 2: Generation games without evaluation of interest

Generation games are implemented 1000 times by Group A without evaluating the interest of requirements and works.

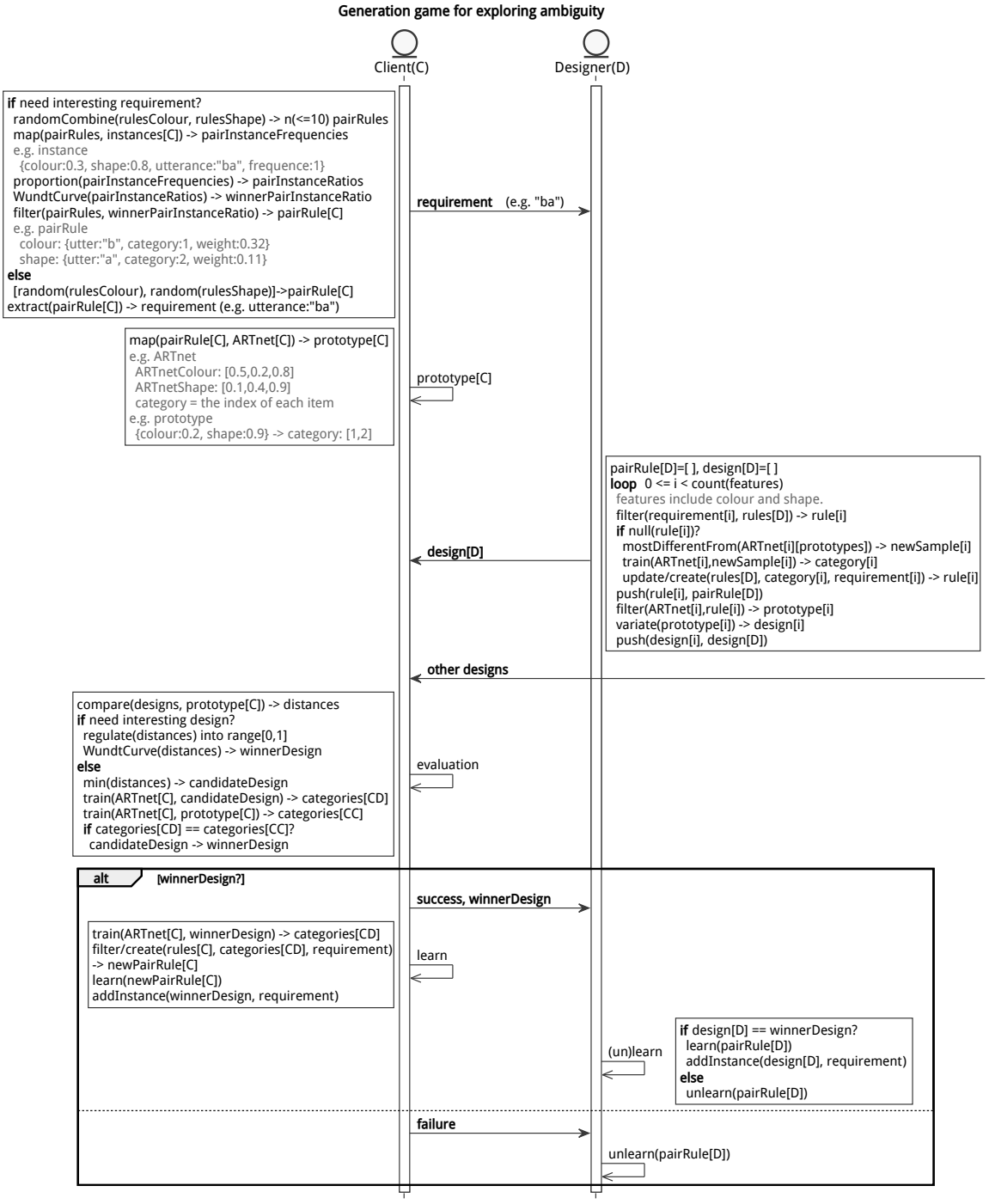


FIGURE 5.3: Generation game for exploring ambiguity

Step 1: The client-agent generates an utterance by combining two randomly selected names of the ART categories (prototypes) related with colour and shape without evaluation.

Step 2: Each designer-agent generates a set of design works by searching existing associations or generating a new association connecting a related ART category with the client-agent's requirement (i.e., utterance).

Step 3: The client-agent selects the most similar design compared with its requirement-associated topic. If the most similar design does not belong to the same ART category as the client-agent's associated topic, the game fails and all designer-agents decrease the weights of their own selected associations. Otherwise the game succeeds, the client-agent finds its own relevant association or generates a new association connecting its own ART category of the selected design and the utterance, then increases the weight of the association and increases the frequency of the related instance or generates a new instance connecting the design and the utterance. At the same time, successful designer-agents increase the weight of the related rule and increase the frequency of the related instance or generate a new instance while other designer-agents decrease the weights of the associations related with the designs rejected by client-agent.

Procedure 3: Generation games with evaluation of the interest of works

Generation games are implemented 1000 times by Group B. The procedure is the same as Procedure 2 except that the client-agent selects design works using the Wundt curve. In the process of selecting design works, the distances between the features of each design works and the features of client-agent's original topic are measured. Then their hedonic value is evaluated. The design with the highest positive interest is selected by the client-agent. If all interests are negative, the generation game fails.

Procedure 4: Generation games with evaluation of the interest of requirements

Generation games are implemented 1000 times by Group C. Each time, the procedure is the same as Procedure 2 except that the client-agent generates several requirements (i.e., utterances) and selects the most interesting one using the Wundt curve. Firstly, the weight of each single utterance in every requirement is calculated by summing the frequencies of the utterances used in all instances. Then the interest values of these requirements are calculated by summing the interests of their own utterances. Finally, the requirement with the highest interest is selected.

Procedure 5: Generation games with evaluation of both the interest of requirements and that of works

Generation games are implemented 1000 times by Group D. In each generation game, the procedure is the same as Procedure 2 except for the generation of interesting requirements and the selection of interesting works by the client-agent. The process of generating interesting utterances is the same as in Procedure 4. The process of selecting interesting works is the same as in Procedure 3.

5.1.5 Results

In Figure 5.4, the radius of each circle represents the frequency of an instance used by an agent. If a topic is associated with more than one utterance, several circles will be

drawn at the same place resulting in a darker colour.

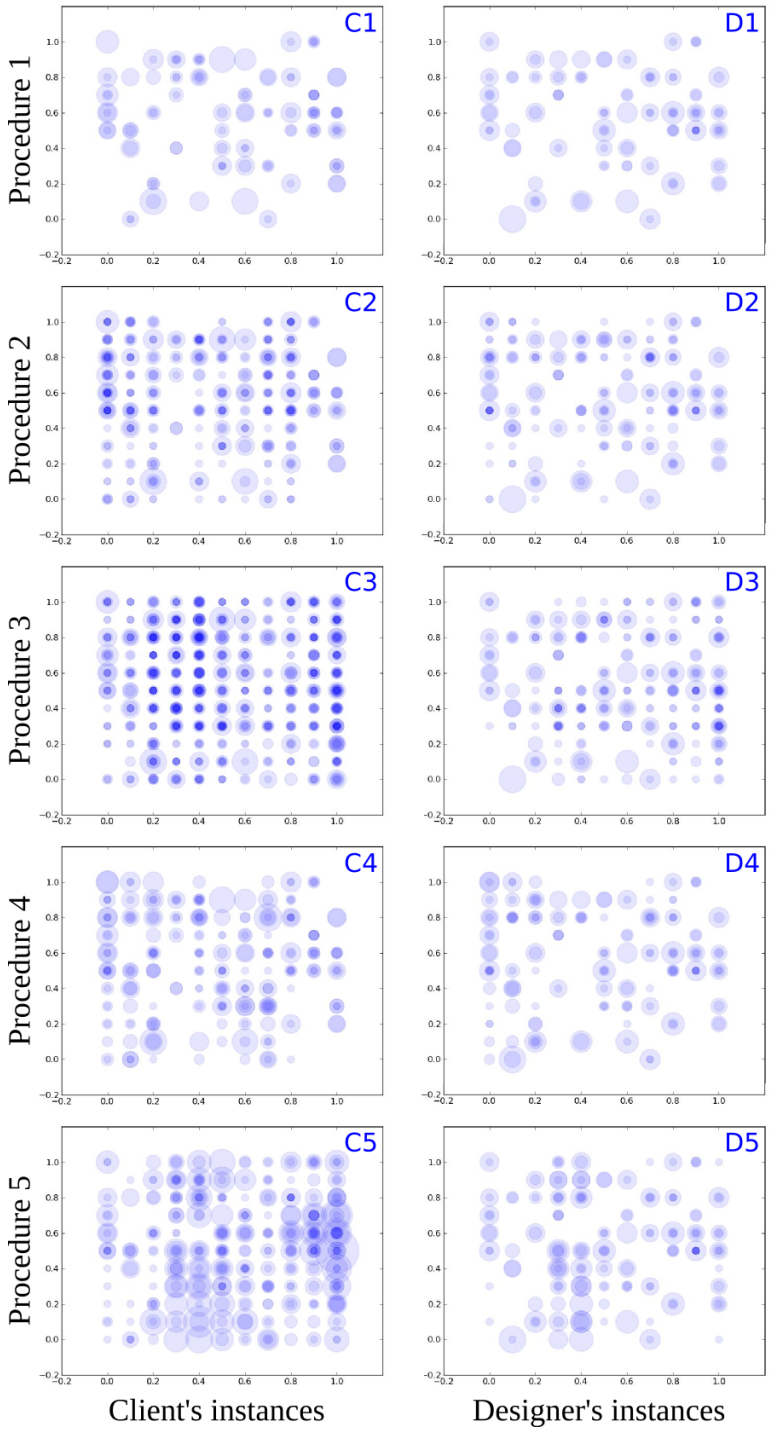


FIGURE 5.4: An example of the distributions of agents' instances

The results of the experiments show that agents explore a greater number of new topics and generate more instances (the associations between topics and utterances) when the client-agent uses the Wundt curve only for selecting interesting design works (see Fig. 5.4 (C3)). But the frequency differences between the instances are not distinctive compared with when the client-agent uses the Wundt curve not only for selecting interesting works,

but also for generating interesting requirements (see Fig. 5.4 (C5)). This suggests that the client-agent prefers using a small set of interesting utterances frequently for generating interesting requirements. So, the frequency-distribution of instances is non-uniform.

The number of designer-agent's instances is less than that of the client-agent's instances (see Fig. 5.4(C2–D5)) except that which is generated in guessing games (see Fig. 5.4(C1,D1)) because only one designer-agent's design could be accepted by the client-agent in a successful generation game. At the same time, other designer-agents have no opportunity to update their instances while the client-agent can update its instances every successful time in generation games. In future experiments, the losers may be able to learn the winning-design, and update their instances in each time when client-agent selects a winning-design.

The average number of instances generated by client-agent and designer-agents in a generation game is shown in Figure 5.5. The number of instances increases sharply especially for the client-agent when only the Wundt curve is used to assess the interest of design works. However, when the Wundt curve is used to evaluate not only the interest of design works but also that of utterances, the number of instances decreases even below that of instances generated without evaluation of interest. The exception is the average number of designer's instances generated in Procedure 5, which is slightly higher than that in Procedure 2 but still lower than that in Procedure 3.

FIGURE 5.5: The average number of agents' instances generated with or without evaluation of interest in generation games

The average max degree of the graph networks of instances generated by client-agent and designer-agents respectively are illustrated in Figure 5.6. As can be seen, the highest average max degree belongs to the instances generated using the Wundt curve to select both interesting requirements and interesting works. The average max degree related to the evaluation of only requirements is higher than that of only works. Therefore, evaluation of the interest of requirements (i.e., utterances) may be more important than that of works.

It is worth mentioning that the agents, especially the client-agent, still explores the vast majority of topics although the guessing game starts with only 10 samples (see

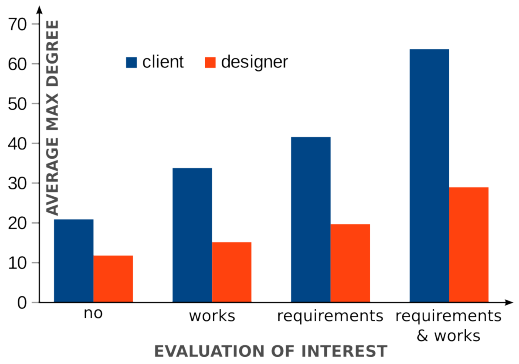


FIGURE 5.6: The average max degree of the graph networks of agents' instances generated with or without evaluation of interest in generation games

Fig. 5.7). This is not only because designers generate design works around suitable prototypes instead of choosing the prototypes directly, but also because the clients select design works using the Wundt curve. However, the distribution of instances becomes non-uniform (see Fig. 5.7 (C3, D3, C5, D5)) when the client generates interesting requirements using the Wundt curve instead of generating requirements randomly.

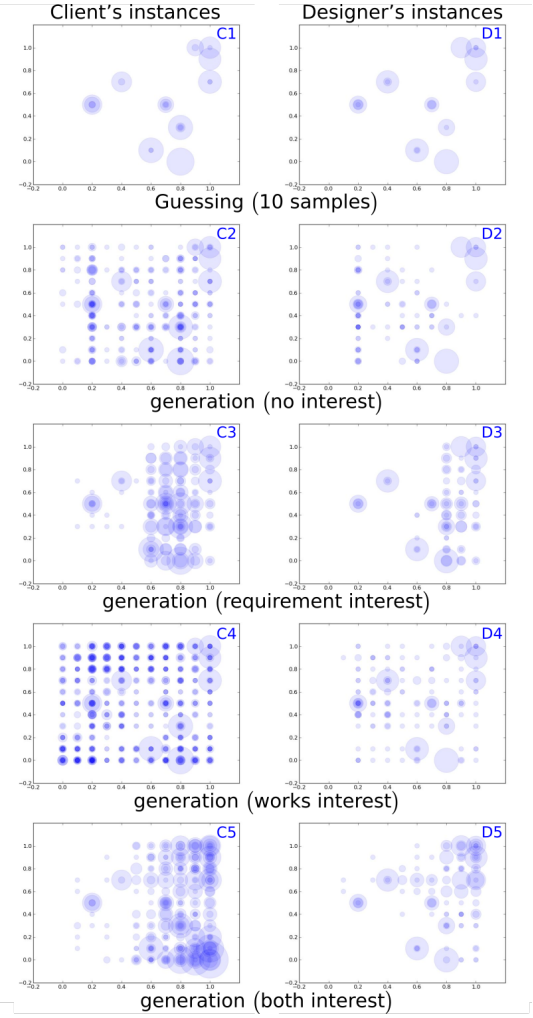


FIGURE 5.7: Another example of the distributions of agents' instances

5.1.6 Discussion

Based on the results of the simulations, client's requirements may be more important than designers' works because the final pattern of the distribution of utterances and design works is primarily determined by the client-agent rather than by the designer-agents (see Fig. 5.8 [2]). In addition, the average max degree related to the evaluation of requirements is higher than that of woks (see Fig. 5.6). Therefore, the evaluation of the interest of requirements may be more important than that of works. Furthermore, interesting requirements may narrow the combination area of utterances initially generated by crossing over two randomly selected utterances, resulting in the selection of interesting artifacts.

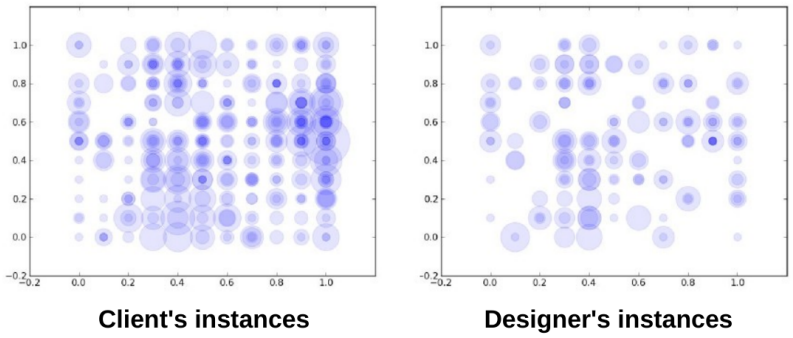


FIGURE 5.8: The instances with evaluation of both requirement & design interest

According to the illustrations of both Figure 5.5 and Figure 5.6, "less is more" is realised as less instances and more connections (see Fig. 5.9). In other words, many more meanings are associated with one utterance while the total number of utterances can be relatively small when using a hedonic function to select randomly combined utterances. Consequently, more connections may lead to discovering more new concepts.

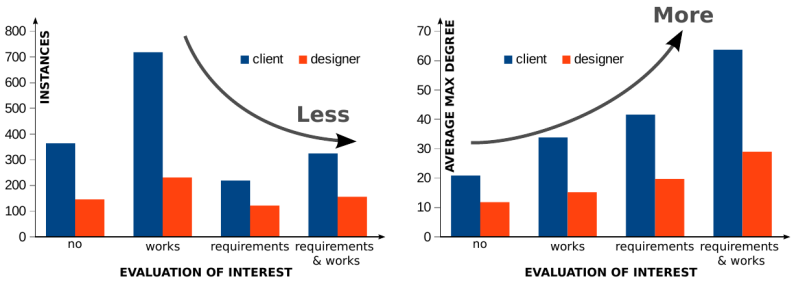


FIGURE 5.9: Less is more

The procedures of language games implemented in this experiment could therefore be adopted in brainstorming by both clients and designers to evolve original requirements

[2]The horizontal axis represents the value of shape while the vertical axis represents the value of colour. Each meaning, which is combined with the two features, shape and colour, is represented by zero or at least one utterance. Each representation (the association between a meaning and an utterance) is an instance. if a meaning is represented by more than one utterances, the utterances illustrated as blue circles will overlap to become darker. If an utterance is used a lot of times, the circle will become bigger.

and novel concepts. The combination of guessing games and generation games can also be utilised in an artificial collaborative system to evolve compositional languages for creative design.

5.1.7 Conclusion

The results of the simulations[3] suggest that the ambiguity of language, especially the ambiguity resulting from polysemy, may play an important role in creative communication by using compositional languages. The number of instances is reduced when using one utterance representing more meanings; at the same time more connections between the instances (i.e., max degree) are generated that leads to connecting more new concepts (see Fig. 5.9). In addition, the direction of exploring conceptual space is affected by using hedonic functions with different settings to evaluate the interest of utterances. As can be seen from Fig. 5.7, the evaluation of the interest of requirements (i.e., utterances) "narrows" the design area (see C3 of Fig. 5.7) compared with others (see C2 and C4 of Fig. 5.7). Further, client-demand driven design may be more important than content driven design in social creative systems due to the distribution of instances mainly determined by the client-agent (see Fig. 5.8) and more connections of the instances resulting from the evaluation of the interest of requirements (see Fig. 5.6).

5.2 Scalability

The aim of this experiment is to expand the abundance of design concepts using the scalability of language. In language games, the meaning of an utterance can be exaggerated without an example having been seen, such as by extending the size of a design to get new spatial functions through the use of modifiers (e.g. "very", "so").

Initially, the utterances representing different sizes or angles (e.g. small, medium, large) or various soft and hard materials could be generated and shared among agents through guessing games. Then modifier-utterances (e.g. "very" → "vo") and their related context free grammar rules (e.g. "very-size" → "very" "size" or "size" "very") can be added into agents' association memory. The relevant prototypes will be generated using conditional rules to distinguish similar meanings such as "very small", "small small", "small" and "small medium".

[3] In this experiment, the evaluation tool based on Graph Theory is used to evaluate the results of language games such as the average max degree of the graph networks of agents' utterance instances.

5.2.1 Subjects

Simulation type 1: Size-samples are randomly generated between 0.25 and 0.75 in guessing games. The sizes smaller than 0.25 and larger than 0.75 are explored using the exaggeration method by combining the modifier, "very", with existing utterances representing "small" and "large" in generation games.

Simulation type 2: Size-samples are randomly generated between 0 and 1 in guessing games. These samples are categorised into three sets (small, medium and large). Then they are subdivided into smaller segments such as "very small" and "very large" in generation games.

5.2.2 Implementation

This experiment is implemented using both guessing games evolving a size-domain language and generation games producing exaggerated sizes via weighted context free grammar (WCFG) with graph network (GN) based on the DIFI framework (see Fig. 5.10).

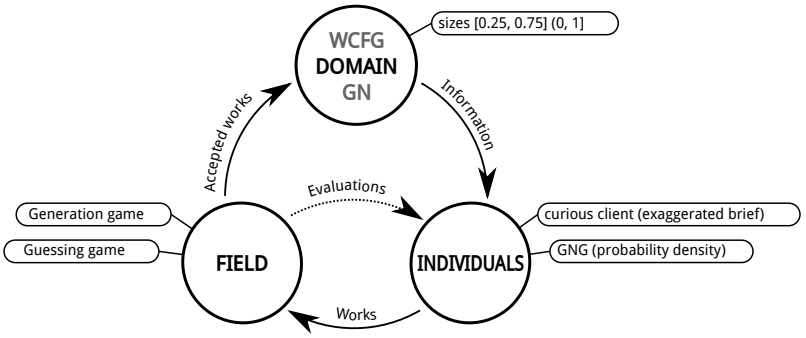


FIGURE 5.10: The Domain-Individual-Field-Interaction (DIFI) framework of exploring scalability

5.2.2.1 Implementation for Simulation Type 1

The agents without language modifiers may be limited to categories without relationship between them. By contrast, the agents with language modifiers can learn how categories being changed via the influence of other categories. For example, a new category involving a modifier such as "very small" is not only determined by "very" and "small" but also influenced by "medium" and "large" when Equation 21 is used. This means the categories of "very small", "very medium" and "very large" build up relationship between "small", "medium" and "large".

$$s' = s + \sum_{i=1}^{n} (s - S_i)/d \tag{21}$$

S is a list of sizes consisting of small, medium and large etc. n is the length of the list. s is one item in the list. d is a factor adjusting the impact of other sizes on the current size, s. s' is the exaggerated size of s. An example of exaggerating prototypes is shown below (see Equations 22, 23, 24).

$$VS = S + ((S - S) + (S - M) + (S - L))/d \tag{22}$$

$$VM = M + ((M - S) + (M - M) + (M - L))/d \tag{23}$$

$$VL = L + ((L - S) + (L - M) + (L - L))/d \tag{24}$$

The procedure of modifying size is described in Algorithm 10. Given $size = 0.257$, $sizes = [0.257, 0.454, 0.633]$ and $d = 4$, the exaggerated size namely "very small" can be calculated using this algorithm resulting in 0.114 (see Table 5.1). Not only the prototype but also min, max, mean and variance of the exaggerated size can be calculated by changing the type of items of S to min, max, mean and variance of the original sizes.

Algorithm 10 Size Modification

```
function MODIFYSIZE(size, sizes, d)
    A ← 0
    for i = 1 → length(sizes) do
        A ← A + (size − sizes_i)
    end for
    return size + A/d
end function
```

As can be seen from Table 5.1, new categories are generated using the equations 22, 23 and 24 with d = 4 and variance factor = 50%. If d is greater than 4, the impact of other categories on VS ("Very Small") may be smaller, and VS will be closer to S ("Small") whereas if d is less than 4, VS will be far away from S, even less than 0 (here, "small" is certainly small, at most containing only part of the range of "very small"). In an extreme example, if d = 0.01, the result will be VS = -57.043, VM = 2.254, and VL = 56.133. This can be very interesting for extending the range of size to generate novel spatial concepts. In the experiment, the min of "very small" is set to be greater than 0 and the max of "very large" is normally set to be less than 1. This can be adjusted by decreasing d or increasing variance, or adding more modifiers such as "very very small" reaching 0 and "very very large" reaching or even over 1; or just by extending "very small" and "very large" to 0 and 1 respectively.

The impact of the existing categories on new categories adjusted by modifiers might be dynamically diffuse. For example, the range of very small is affected by that of small, medium and large with different affecting weights (W.small > W.medium > W.large for "very small"). In addition, the modifier itself will evolve and develop new categories

TABLE 5.1: Categories with language modifier

Size	**Mean** (prototype)	**Variance**	**Min**	**Max**
Small (S)	0.257	0.150	0.107	0.407
Medium (M)	0.454	0.150	0.304	0.604
Large (L)	0.633	0.150	0.483	0.783
Very Small (VS)	0.114	0.075	0.039	0.189
Very Medium (VM)	0.458	0.075	0.383	0.533
Very Large (VL)	0.772	0.075	0.697	0.847

dynamically by giving agents a range of numbers to select for categorisation instead of setting a certain range in advance.

In this experiment, exaggerational languages evolve by generating compositional utterances using a weighted context free grammar (WCFG). For example, "za vo" representing the meaning of "very small" is collected during the following process (see Table 5.3), "compo-size" → "very-size" → "size & very" (weight: 0.871) rather than "very & size" (weight: 0.152) → "size0.257" & "vo" (weight: 0.892) rather than "so" (weight: 0.010) → "za" (weight: 0.995) & "vo" → "za vo". In generation games, a client-agent may select rules via direct weighted selection or roulette selection to generate appropriate compositional utterances.

An agent's association rules include basic rules and compositional rules. The former rules are primarily generated in guessing games while the latter rules are developed in generation games. In guessing games, new categories related to basic rules are generated when a new topic is out of the area of existing categories. Each category is defined with prototype and tolerance; for example, the category with prototype, 0.2 and tolerance, 0.1 is [0.2-0.1, 0.2+0.1] → [0.1,0.3]. Another category could be [0.25,0.55] from prototype, 0.4 and tolerance 0.15. 0.27 is more likely to belong to category [0.1,0.3] than to category [0.25,0.55] because the distance between 0.27 and prototype 0.2 is less than that of prototype 0.4. The overlap of categories could be a source of creativity. In generation games, new categories associated with compositional rules are generated temporally based on the previous generated categories related to the basic rules. In this experiment, the compositional categories are generated according to the following conditional rules (see Equations 25, 26, 27, 28, 29). The related function, getting the prototype and tolerance of an exaggerated size, is presented in Algorithm 11.

$$t' = \begin{cases} 0.5 * t & \text{if } VS, VL, VM \\ 0.7 * t & \text{if } SS, MM, LL \\ (t_1 + t_2)/2 & \text{if } SM, SL, ML \text{ e.g. } SM\text{: } (t_S + t_M)/2 \end{cases} \tag{25}$$

$$p' = \begin{cases} p + \sum_{i=1}^{n}(p - P_i)/4 & \text{if } VS, VL, VM, MM \\ p + \sum_{i=1}^{n}(p - P_i)/7 & \text{if } SS, LL \\ (p_1 + p_2)/2 & \text{if } SM, SL, ML \text{ e.g. } SM\text{: } (p_S + p_M)/2 \end{cases} \tag{26}$$

$$p' = \begin{cases} max(0, p') & \text{if } VS, SS \\ min(1, p') & \text{if } VL, LL \end{cases} \tag{27}$$

$$p'' = \begin{cases} (p' + t')/2 & \text{if } VS \wedge (p' - t' \neq 0) \\ (1 + (p' - t'))/2 & \text{if } VL \wedge (p' + t' \neq 1) \end{cases} \tag{28}$$

$$t'' = \begin{cases} (p' + t')/2 & \text{if } VS \wedge (p' - t' \neq 0) \\ (1 - (p' - t'))/2 & \text{if } VL \wedge (p' + t' \neq 1) \end{cases} \tag{29}$$

S: small, M: medium, L: large, V: very $\rightarrow$ e.g. VS: very small
t: tolerance, t': the tolerance of exaggerated size, t'': the regulated tolerance of exaggerated size
p: a size-prototype, p': the prototype of exaggerated size, p'': the regulated prototype of exaggerated size, P: a list of size-prototypes

The process of the guessing game for Simulation Type 1 is illustrated in Figure 5.11. First, the speaker-agent selects a random-topic and maps it to a relevant rule or creates a new rule in which the associated utterance can be obtained. This utterance will be sent to the listener-agent who compares the utterance with all samples in the context, finds a relevant rule and then maps it to a topic. This guessed topic will be categorised in the normal way by speaker-agent and compared with the categories of the speaker's selected topic. If the two sets of categories share at least one category, the guessing game succeeds. Both speaker and listener learn their respective rules. Otherwise, the listener unlearn its rule, then generate a new rule related to the speaker's selected topic and utterance, and learn the new generated rule.

The process of the generation game (see Fig. 5.12) for Simulation Type 1 is described as follows. First, the client-agent traces forward the rule, "compositional size", to obtain a compositional utterance (i.e., requirement) and several rules related to the tracing-path. A pair of terminal rules are extracted from these rules and categorised to find relevant prototype and tolerance by the client-agent. Then the designer-agent traces backward the client's requirement to get a number of rules, extracts a pair of terminal rules from them, and then categorises the pair of rules to get another couple of prototype and tolerance which is used to generate a specific design. This design and other designers' design works are compared with the client's prototype. If the distance between the

Algorithm 11 Getting The Prototype and Tolerance of An Exaggerated Size

function GETEXAGGERPROTOL($agent, exaggerRule$)
  if $exaggerRule \in [vs, vm, vl, sv, mv, lv]$ **then**
    $size0 \leftarrow remove(v, exaggerRule)$
    $tolerance \leftarrow agent['nodes][size0]['tolerance] \times 0.5$
  else if $exaggerRule \in [ss, mm, ll]$ **then**
    $size0 \leftarrow exaggerRule[0]$
    $tolerance \leftarrow agent['nodes][size0]['tolerance] \times 0.7$
  else if $exaggerRule \in [sm, sl, ml, ms, ls, lm]$ **then**
    $size0 \leftarrow exaggerRule[0]$
    $size1 \leftarrow exaggerRule[1]$
    $tolerance0 \leftarrow agent['nodes][size0]['tolerance]$
    $tolerance1 \leftarrow agent['nodes][size1]['tolerance]$
    $tolerance \leftarrow (tolerance0 + tolerance1)/2$
  end if
  if $exaggerRule \in [vs, vm, vl, sv, mv, lv, mm]$ **then**
    $size0 \leftarrow remove(v, exaggerRule)$
    $p \leftarrow agent['nodes][size0]['prototype]$
    $A \leftarrow sum(map(x \rightarrow (p - agent['nodes][x]['prototype]), [s, m, l]))$
    $prototype \leftarrow p + A/4$
  else if $exaggerRule \in [ss, ll]$ **then**
    $size0 \leftarrow exaggerRule[0]$
    $p \leftarrow agent['nodes][size0]['prototype]$
    $A \leftarrow sum(map(x \rightarrow (p - agent['nodes][x]['prototype]), [s, m, l]))$
    $prototype \leftarrow p + A/7$
  else if $exaggerRule \in [sm, sl, ml, ms, ls, lm]$ **then**
    $size0 \leftarrow exaggerRule[0]$
    $size1 \leftarrow exaggerRule[1]$
    $p0 \leftarrow agent['nodes][size0]['prototype]$
    $p1 \leftarrow agent['nodes][size1]['prototype]$
    $prototype \leftarrow (p0 + p1)/2$
  end if
  if $exaggerRule \in [vs, sv, ss]$ **then**
    $prototype \leftarrow max(0, prototype)$
  else if $exaggerRule \in [vl, lv, ll]$ **then**
    $prototype \leftarrow min(1, prototype)$
  end if
  if $(exaggerRule \in [vs, sv]) \wedge (prototype - tolerance) \neq 0$ **then**
    $prototype \leftarrow (prototype + tolerance)/2$
    $tolerance \leftarrow prototype$
  else if $(exaggerRule \in [vl, lv]) \wedge (prototype + tolerance) \neq 1$ **then**
    $prototype1 \leftarrow prototype$
    $tolerance1 \leftarrow tolerance$
    $prototype \leftarrow (1 + (prototype1 - tolerance1))/2$
    $tolerance \leftarrow (1 - (prototype1 - tolerance1))/2$
  end if
  return $\{'prototype : prototype, 'tolerance : tolerance\}$
end function

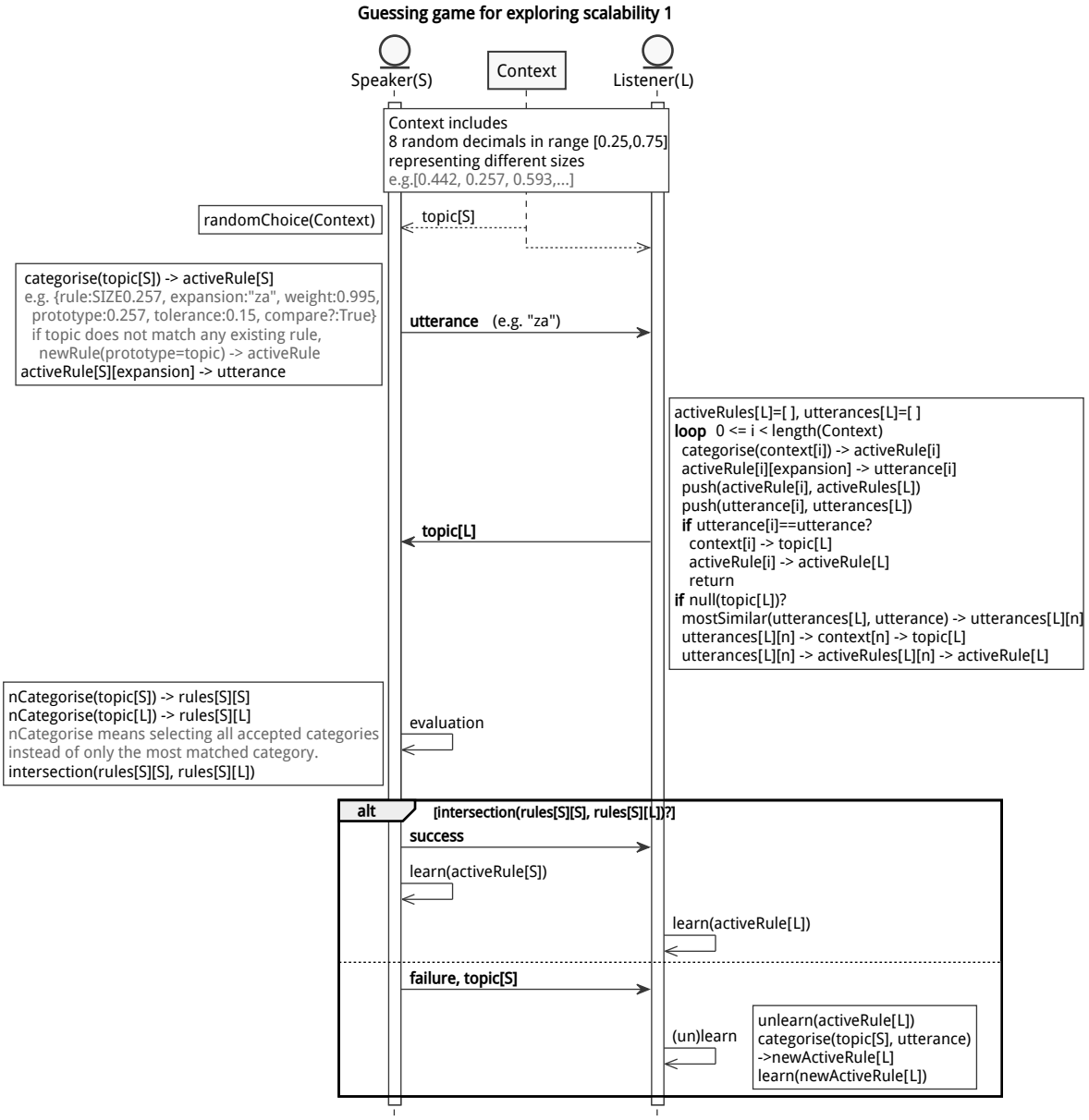


FIGURE 5.11: Guessing game (type-1) for exploring scalability

prototype and a design is in the range of client's tolerance, the design is accepted. After all the designs have been compared with the client's prototype, if there is at least one accepted design, the game succeeds and the client learns its own rules. At the same time the designer also learns its own rules if its design is a member of the accepted designs. Otherwise it unlearns the rules, generates new terminal rules based on the best design and the requirement, and then learns the new terminal rules. If there is no accepted design, the game fails and the designer-agents unlearn their respective rules.

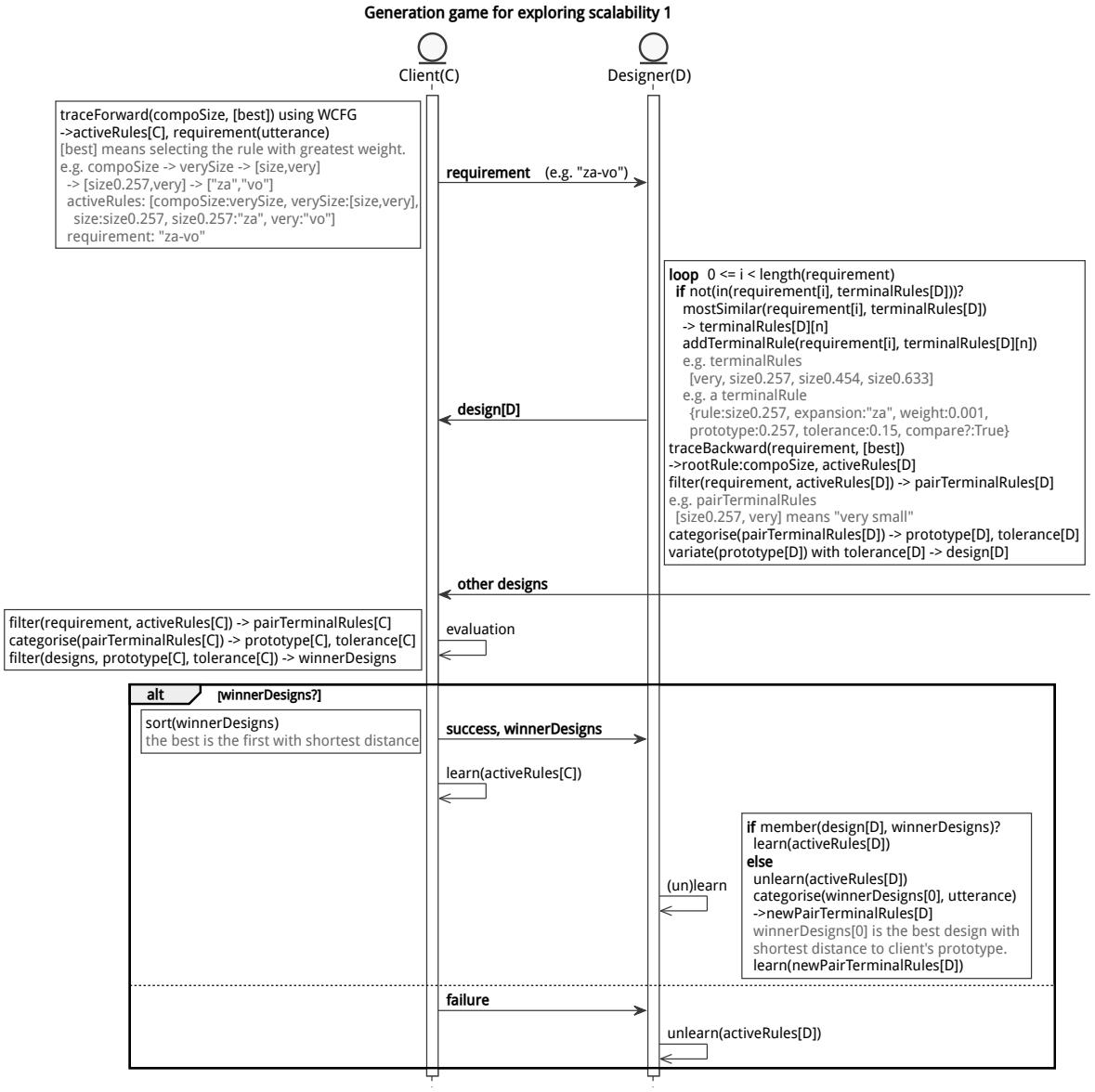


FIGURE 5.12: Generation game (type-1) for exploring scalability

5.2.2.2 Implementation for Simulation Type 2

Another way of representing sizes and exaggerated sizes is to use range, which is the combination of prototype (i.e., mean) and tolerance (i.e., variance). "very" is the result of increasing the difference of a category from other categories. New exaggerated feature categories can be generated by adding modifiers to existing categories. For example, the range of "very medium" can be generated by calculating the mean of medium size to maximise the distance to other categories such as small and large (If the range of medium was [0.304, 0.604], the mean will be 0.454) and reducing the variance of medium size by a factor (50%). If the variance of medium was 0.15, the variance of "very medium" will be 0.075. Here, the original size is equal to the general size, which contains the exaggerated size. For instance, very large is a part of large. The range of "very large" could be

[0.8, 1] derived from "large", [0.6, 1] via $1 - 0.5 \times (1 - 0.6)$ while the range of "very small" would be (0, 0.2] derived from "small", (0, 0.4] via 0.5×0.4. The equations for calculating the range of exaggerated sizes with $\text{small}_{\text{min}}$=0 and $\text{large}_{\text{max}}$=1 are described below (see Equations 30, 31; R' is the range of exaggerated size while R is the range of original size. a is the factor for reducing the variance of original size. $0 < a < 1$).

$$R'_{min} = \begin{cases} 0 & \text{if } R \text{ is } small \\ R_{min} + (1-a)(R_{max} - R_{min})/2 & \text{if } R \text{ is } medium \\ 1 - a * (1 - R_{min}) & \text{if } R \text{ is } large \end{cases} \tag{30}$$

$$R'_{max} = \begin{cases} a * R_{max} & \text{if } R \text{ is } small \\ R_{max} - (1-a)(R_{max} - R_{min})/2 & \text{if } R \text{ is } medium \\ 1 & \text{if } R \text{ is } large \end{cases} \tag{31}$$

To calculate the exaggerated sizes more dynamically, the probability density function of the truncated normal distribution between 0 and 1 is utilised. The probability density function of the truncated normal distribution (probability-t-pdf) is given by Equation 32 (also see Algorithm 12, default settings: $u = 0.0$, $sigma = 1.0$, $a = 0.0$, $b = 1.0$).

$$f\left(x; \mu, \sigma, a, b\right) = \frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})} \tag{32}$$

Algorithm 12 Probability Density Function of the Truncated Normal Distribution

```
function tPdf(x, u, sigma, a, b)
    return (sPdf((x − u)/sigma)/sigma)/(sCdf((b − u)/sigma) − sCdf((a − u)/sigma))
end function
```

Where $\phi(\xi)$ is the standard normal distribution, which is defined as Equation 33 (also see Algorithm 13).

$$\phi(\xi) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}\xi^2) \tag{33}$$

Algorithm 13 Probability Density Function of the Standard Normal Distribution

```
function sPdf(x)
    return exp(−0.5 × x × x)/√(2 × π)
end function
```

And $\Phi(x)$ is the cumulative distribution function of the standard normal definition, which is defined as Equation 34 (also see Algorithm 14 [4]).

[4]The algorithm is translated from https://www.johndcook.com/blog/python_phi/

$$\Phi(x) = 0.5 + \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2} \left[x + \frac{x^3}{3} + \frac{x^5}{3 \cdot 5} + \cdots + \frac{x^{2n+1}}{(2n+1)!!} + \cdots \right] \tag{34}$$

Algorithm 14 Cumulative Distribution Function of the Standard Normal Distribution

```
1: function sCdf(x)
2:     a1 ← 0.254829592
3:     a2 ← −0.284496736
4:     a3 ← 1.421413741
5:     a4 ← −1.453152027
6:     a5 ← 1.061405429
7:     p ← 0.3275911
8:     sign ← 1
9:     if x < 0 then
10:        sign ← −1
11:    end if
12:    x ← abs(x)/sqrt(2)
13:    z ← 1/(1 + p × x)
14:    y ← 1 − (((((a5 × z + a4) × z + a3) × z + a2) × z + a1) × z × exp(−x × x))
15:    return 0.5 × (1 + sign × y)
16: end function
```

The artificial neural network Growing Neural Gas (GNG) is utilised to categorise the samples with different sizes. As can be seen from Table 5.2, nine nodes (n0, n1,. . . , n8) of a GNG network are utilised to categorise basic sizes including "small" (S), "medium" (M) and "large" (L), and exaggerated sizes such as "very small" (VS) and "small small" (SS). The categories of these sizes will be updated in time after training the GNG network with each sample. Therefore, both prototypes and tolerances of the categories change dynamically instead of being fixed.

Table 5.2: Categorizing sizes using GNG & probability density

Size	Prototype	Variation(sigma)
S	n1	(n1 + n4)/4
M	n4	(n7 - n1)/4
L	n7	(2 - n4 - n7)/4
VS	0.0	n0
VM	n4	(n5 - n3)/8
VL	1.0	1 - n8
SS	n0	(n0 + n1)/4
MS	n1	(n2 - n0)/4
LS	n2	(n3 - n1)/4
SM	n3	(n4 - n2)/4
MM	n4	(n5 - n3)/4
LM	n5	(n6 - n4)/4
SL	n6	(n7 - n5)/4
ML	n7	(n8 - n6)/4
LL	n8	(2 - n7 - n8)/4

As can be seen from Figure 5.13, the guessing game for Simulation Type 2 also starts from random topic selection by the speaker-agent, who categorises this topic to a relevant category-rule using GNG. Then an utterance relating to the rule is generated and sent to the listener-agent, who generates its own rule based on the utterance and the samples in the context, then match the rule to a relevant topic. The listener's rule is compared with the speaker's rule. If they are the same, the game succeeds, and both speaker and listener learn their respective associations between the rules and utterance. Otherwise, the listener unlearns the association between its own rule and utterance, then generates a new rule based on the speaker's selected topic and utterance, and learns the association between the new rule and the utterance.

The generation game for Simulation Type 2 (see Fig. 5.14) also starts from tracing forward the rule, "compositional size", to obtain a compositional utterance (i.e., requirement), a list of associations between rules and their expansions (i.e., active-rule-expansions) related to the tracing path as well as the category-rule by a client-agent. Then a designer-agent traces backward the client-agent's requirement to get its own active-rule-expansions and category-rule. This category-rule is used to generate a specific design. Then the client categorises all designs to relevant category-rules and compare these rules with its original category-rule. If they are the same, the related design will be accepted. After evaluating all the designs, if there is at least one accepted design, the game succeeds, and the client learns its own active-rule-expansions. The designer also learns its own active-rule-expansions if its design is one of the accepted designs; otherwise, the designer unlearns the active-rule-expansions and generates new active-rule-expansions based on the best design and the requirement, then learns the new active-rule-expansions. If there is no accepted design, all designers unlearn their respective active-rule-expansions.

5.2.3 Results of Simulation Type 1

During simulation, speaker and listener are randomly selected from 6 agents before each guessing game while one client and three designers are randomly selected from 6 agents before each generation game. At the completion of a simulation, 23 basic utterances are generated. Among these, "za" and "ge" are successfully associated with the meaning, "small"; "wu" and "hoxo" mainly represent the meaning, "medium"; "xufa" represents "large"; and "vo" represents "very". Each agent generates three basic categories: "small", "medium" and "large", although the range of the same category generated from each agent differs slightly owing to their different experiences on various topics and different sequences of these topics.

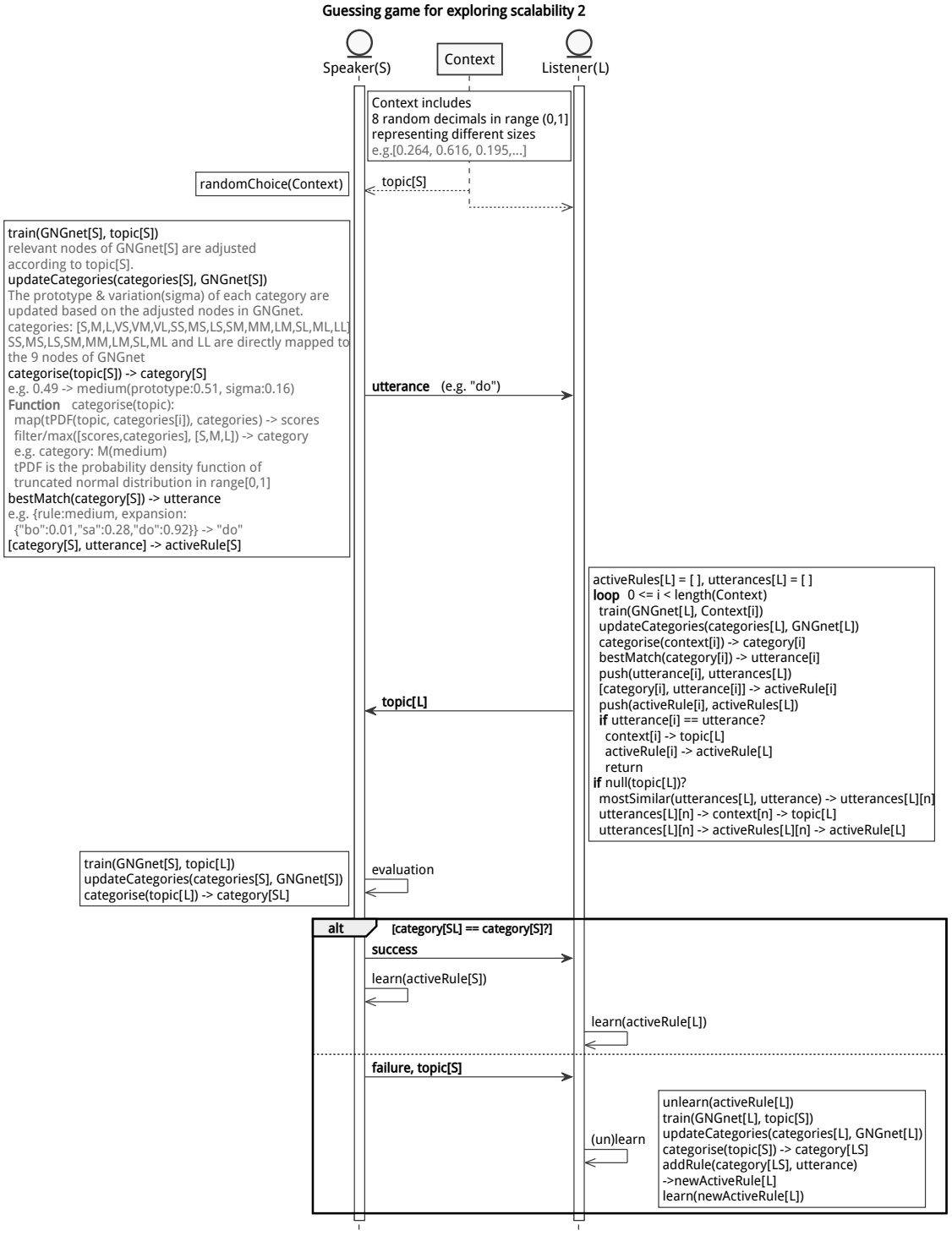


FIGURE 5.13: Guessing game (type-2) for exploring scalability

5.2.3.1 Individual's Rules

After completing the generation games, each individual collects a number of association rules. As can be seen from Table 5.3, an agent collects 30 rules associating compositional rules with terminal rules and utterances. If Compare? is 1, the related rule may be compared with other rules with the same rule name. For example, rule-2 and rule-3

FIGURE 5.14: Generation game (type-2) for exploring scalability

with the same rule name, "very-size", could be compared by the client-agent who prefers to select rule-3, which has higher weight. Three terminal categories—size0.257 (small), size0.454 (medium) and size0.633 (large)—are generated. Each category is associated with several utterances which can be compared and selected according to their different weights. The final result shows that the associations ["very","vo"], ["size0.257","ge"], ["size0.257","za"], ["size0.454","wu"], ["size0.454","hoxo"], ["size0.633", "xufa"] could be selected more often than other rules due to their high weights.

TABLE 5.3: An agent's association rules

No.	Rule	Expansion	Weight	Prototype	Tolerance	Compare?
0	COMPO-SIZE	VERY-SIZE	0.871	N/A	N/A	0
1	COMPO-SIZE	SIZE-SIZE	0.938	N/A	N/A	0
2	VERY-SIZE	VERY,SIZE	0.152	N/A	N/A	1
3	VERY-SIZE	SIZE,VERY	0.871	N/A	N/A	1
4	SIZE-SIZE	SIZE,SIZE	0.938	N/A	N/A	0
5	SIZE	SIZE0.257	0.995	N/A	N/A	0
6	SIZE	SIZE0.454	0.950	N/A	N/A	0
7	SIZE	SIZE0.633	0.836	N/A	N/A	0
8	VERY	"so"	0.010	N/A	N/A	1
9	VERY	"vo"	0.892	N/A	N/A	1
10	SIZE0.257	"ho"	0.009	0.257	0.15	1
11	SIZE0.257	"la"	0.010	0.257	0.15	1
12	SIZE0.257	"pa"	0.108	0.257	0.15	1
13	SIZE0.257	"hoxo"	0.109	0.257	0.15	1
14	SIZE0.257	"xufa"	0.109	0.257	0.15	1
15	SIZE0.257	"zira"	0.109	0.257	0.15	1
16	SIZE0.257	"ge"	0.970	0.257	0.15	1
17	SIZE0.257	"za"	0.995	0.257	0.15	1
18	SIZE0.454	"ve"	0.010	0.454	0.15	1
19	SIZE0.454	"xufa"	0.109	0.454	0.15	1
20	SIZE0.454	"ge"	0.415	0.454	0.15	1
21	SIZE0.454	"wu"	0.931	0.454	0.15	1
22	SIZE0.454	"hoxo"	0.980	0.454	0.15	1
23	SIZE0.633	"ge"	0.009	0.633	0.15	1
24	SIZE0.633	"vo"	0.009	0.633	0.15	1
25	SIZE0.633	"hoxo"	0.098	0.633	0.15	1
26	SIZE0.633	"ho"	0.109	0.633	0.15	1
27	SIZE0.633	"mewi"	0.109	0.633	0.15	1
28	SIZE0.633	"wu"	0.109	0.633	0.15	1
29	SIZE0.633	"xufa"	0.879	0.633	0.15	1

5.2.3.2 Utterances and Prototypes

The most successful associations between utterances and prototypes for all agents are listed in Table 5.4. Most of these utterances represent the same categories of each agent except "wu", which is related to Agent-5's "small" category and other agents' "medium" categories, probably because the prototype (0.282) of Agent-5's "small" category is very close to other agents' "medium" categories. In terms of various weights of the associations between the utterances and the prototypes, "za" is related to a little smaller size than "ge" while "wu" relates to a smaller size than "hoxo" although both of them represent the meaning of medium size; large size is only represented by "xufa" (see Table 5.4 and Fig. 5.15). Further, regarding the history of generating rules, "xufa" was also related to "very"; "vo" was used to represent medium size (0.492 by Agent-5) then changed to mainly represent "very". Even when a meaning has been mainly represented by certain utterances, it is still possible to evolve new associations between existing or new utterances and this meaning. For instance, a new association between "ziza" and size0.282 occurs after "za" and "wu" primarily representing Agent-5's size0.282. The diversity of dynamic changeable representations related to ambiguity may lead to creative communication and designing.

TABLE 5.4: Main associations between utterances and prototypes
(P: prototype, P0: Agent-0's prototype, W: weight, T: tolerance)

Utter.	P0	W0	P1	W1	P2	W2	P3	W3	P4	W4	P5	W5	T
"za"	0.269	0.907	0.256	0.995	0.261	0.995	0.257	0.995	0.256	0.983	0.282	0.941	0.15
"ge"	0.269	0.978	0.256	0.874	0.261	0.908	0.257	0.970	0.256	0.970	0.282	0.978	0.15
"wu"	0.448	0.995	0.421	0.970	0.413	0.995	0.454	0.931	0.412	0.980	0.282	0.800	0.15
"hoxo"	0.448	0.912	0.421	0.901	0.413	0.942	0.454	0.980	0.412	0.863	0.492	0.995	0.15
"xufa"	0.669	0.911	0.643	0.968	0.684	0.991	0.633	0.879	0.619	0.886	0.734	0.991	0.15

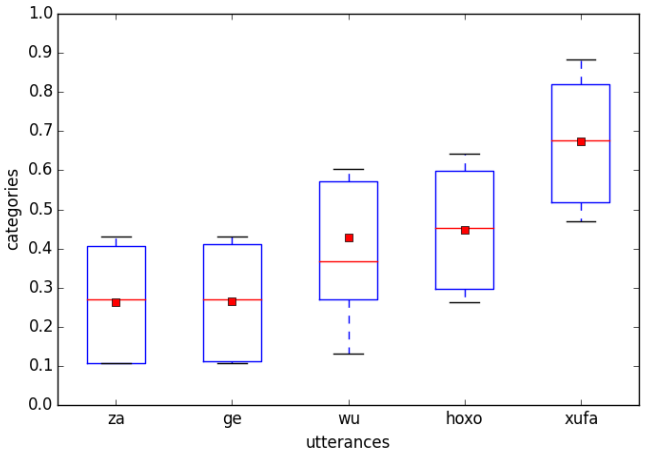


FIGURE 5.15: The associations between utterances & categories

5.2.3.3 Exaggerational Rules

After generation games, each agent's exaggerational rules (see Table 5.5 and Fig. 5.16) are generated based on the basic rules associated with "small", "medium" and "large". The exaggerational rule, "very small", covers smaller size to 0 while "very large" represents larger size to 1. "Very medium" is associated with the same prototype as "medium" but less tolerance while the tolerance of "medium medium" is between that of "very medium" and "medium"; "large large" and "small small" has the same tolerance as "medium medium" but a slightly bigger or smaller prototype-size than "large" and "small" respectively. In addition, the pair-combinations of "small", "medium" and "large" are related to the intermediate areas of each pair, e.g. the prototype of "small medium" is the average of those of "small" and "medium". Further, the exaggerational rule, "size very", evolves to be more preferred than "very size" in this simulation (see Table 5.6).

5.2.3.4 Exaggerational Rules Generated in Another Simulation

Exaggerational rules can be generated efficiently by setting d with an appropriate value in equation 21. As can be seen from Table 5.7 and Figure 5.17, the prototypes of VS, VM, MM and VL are calculated with d = 4 and those of SS and LL are generated with

TABLE 5.5: An agent's exaggerational rules

Rule	ab.	Utterance	Prototype	Tolerance
Very Small	VS	"vo za"("vo ge")	0.110	0.110
Small Small	SS	"za za"("ge ge")	0.163	0.120
Small	S	"za"("ge")	0.257	0.150
Small Medium	SM	"za wu"("za hoxo","ge wu","ge hoxo")	0.356	0.150
Medium	M	"wu"("hoxo")	0.454	0.150
Very Medium	VM	"vo wu"("vo hoxo")	0.454	0.105
Medium Medium	MM	"wu wu"("hoxo hoxo")	0.454	0.120
Small Large	SL	"za xufa"("ge xufa")	0.454	0.150
Medium Large	ML	"wu xufa"("hoxo xufa")	0.544	0.150
Large	L	"xufa"	0.633	0.150
Large Large	LL	"xufa xufa"	0.727	0.120
Very Large	VL	"vo xufa"	0.836	0.164

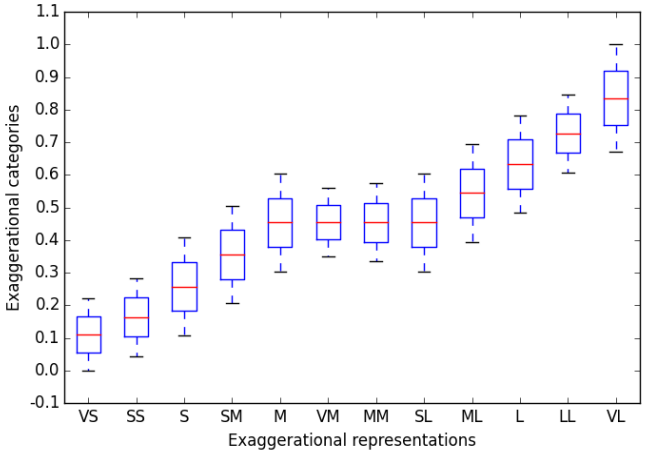


FIGURE 5.16: An agent's exaggerational categories

TABLE 5.6: The weights of "very-size" rules

Agent	"size very"	"very size"
0	0.869	0.177
1	0.944	0.350
2	0.993	0.325
3	0.871	0.152
4	0.834	0.010
5	0.928	0.415

d = 7, which makes the ranges of SS and LL closer to S and L than to VS and VL respectively. "Very small" is extended to 0 by keeping max (0.184 + 0.150) = 0.334 but changing tolerance 0.334/2 = 0.167 and prototype to 0.167, while "very large" is expanded to 1 by keeping min (0.782 - 0.150) = 0.632 but changing tolerance ((1 - 0.632) / 2) = 0.184 and prototype (1 - 0.184) = 0.816. After such adjustments, the tolerances of both VS and VL are greater than those of S and V because the prototypes of S and V are considerably far away from 0 and 1, respectively, in this simulation.

TABLE 5.7: An agent's exaggerational rules in another simulation

Rule	ab.	Utterance	Prototype	Tolerance
Very Small	VS	"se fu"("gi fu")	0.167	0.167
Small Small	SS	"fu fu"	0.237	0.120
Small	S	"fu"	0.309	0.150
Small Medium	SM	"fu wita"("fu saro")	0.388	0.150
Medium	M	"wita"("saro")	0.468	0.150
Very Medium	VM	"se wita"("gi wita", "se saro", "gi saro")	0.462	0.105
Medium Medium	MM	"wita wita" ("saro saro")	0.462	0.120
Small Large	SL	"fu ku"("fu qeva")	0.480	0.150
Medium Large	ML	"wita ku"("saro ku","wita qeva","saro qeva")	0.560	0.150
Large	L	"ku"("qeva")	0.651	0.150
Large Large	LL	"ku ku"("qeva qeva")	0.726	0.120
Very Large	VL	"se ku"("se qeva", "gi ku", "gi qeva")	0.816	0.184

All variances are the same as the settings of the former simulation except that the variance of very medium is changed to (0.7 × variance) instead of (0.5 × variance) because tightening the range hinders the success of language games.

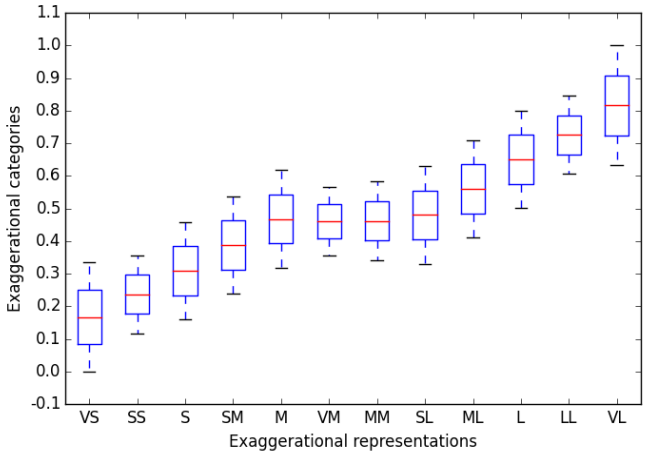


FIGURE 5.17: An agent's exaggerational categories in another simulation

5.2.4 Results of Simulation Type 2

The mean and deviation of each category of an agent are dynamically changed and adjusted when categorising new samples in guessing games and designs in generation games. In detail, the positions of three categorising nodes, including the one that is closest to the sample or design, and its two neighbours are adjusted using the algorithm of Growing Neural Gas (GNG), which is similar to that of SOM but is extensible.

TABLE 5.8: An agent's weighted association rules

Rule	Compo-size	Size	Very		S		M		L	
expan.	[very, size]	S	"bo"	0.86	"doho"	0.98	"sa"	0.96	"ca"	0.82
	[size, size]	M	"sa"	0.28	"ga"	0.42	"ye"	0.71	"ga"	0.42
		L	"ca"	0.20	"sa"	0.28	"ga"	0.47	"lizu"	0.20
			"doho"	0.01	"ha"	0.11	"ca"	0.28	"fale"	0.11
			"ye"	0.01	"fa"	0.01	"bo"	0.27	"bo"	0.01
			"ga"	0.01	"bo"	0.01	"wu"	0.11	"doho"	0.01
			"ve"	0.01	"ca"	0.01	"ha"	0.11	"sa"	0.01
					"fale"	0.01	"lizu"	0.01	"wu"	0.01
					"ye"	0.01	"doho"	0.01	"ye"	0.01

5.2.4.1 Individual's Rules

In simulation type 2, each agent also stores a number of association rules based on context free grammar. These rules include root/branch rules: "compo-size" & "size", and terminal rules: "very", "small", "medium" & "large" related to weighted utterances, e.g. very: {"bo": 0.86, "sa": 0.28, "ca": 0.20,...} (see Table 5.8). After completing the language games, almost all agents share the primary utterances, "bo" representing "very", "doho" related to "small", "sa" relating to "medium", and "ca" denoting "large".

5.2.4.2 Categories with GNG and Probability Density

To match the changing samples during language games, the tolerance (i.e., sigma, variation, threshold, deviation) is dynamically changed according to the changing distances between the nodes (i.e., prototypes) in the GNG network (see Fig. 5.18 and Fig. 5.19) instead of fixed values such as 0.2 for {S,M,L}, 0.03 for {VS,VM,VL}, and 0.1 for {SS,MS,LS,SM,MM,LM,SL,ML,LL}.

Initially, nine nodes with random-uniform between 0.0 and 1.0 are generated and sorted as the means of "small-small", "medium-small", "large-small", "small-medium", "medium-medium", "large-medium", "small-large", "medium-large" and "large-large" for each agent. The means of "small", "medium" and "large" are the same as those of "medium-small", "medium-medium" and "medium-large". For the modifier, "very", the mean of "very-small" is set to 0.0 while that of "very-large" is set as 1.0; the mean of "very-medium" is the same as that of "medium" as well as "medium-medium" but their deviations are different: $d_{VM} < d_{MM} < d_M$ (VM: very-medium, MM: medium-medium, M: medium).

In guessing games, only the categories of "small", "medium" and "large" are used to evolve basic utterances which will be utilised as the units of compositional utterances generated to represent exaggerated-meanings such as "very-small" in generation games.

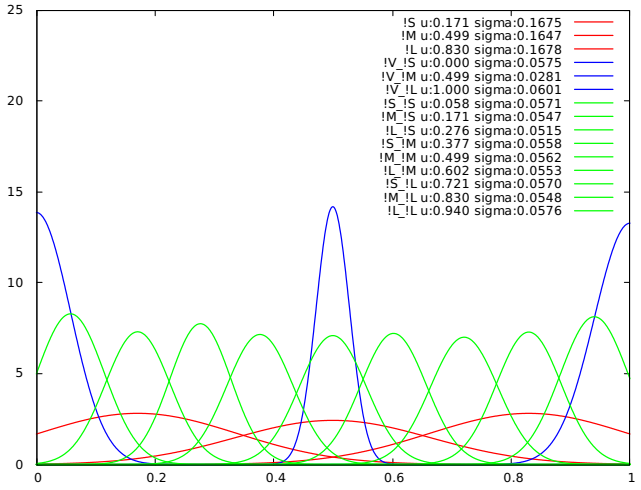


FIGURE 5.18: An agent's categories

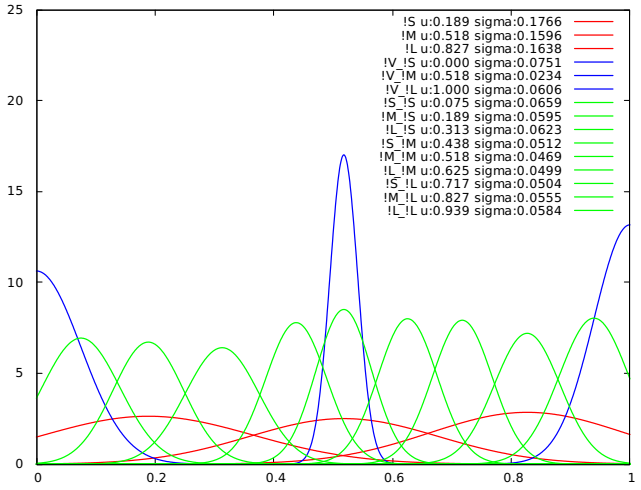


FIGURE 5.19: Another agent's categories

As can be seen from Fig. 5.18 [5] and Fig. 5.19, after the generation games are completed, the results of two agents' categories, which are generated with the probability density function of truncated normal distribution between 0 and 1, are a little different due to their different experiences with samples and designs, and different initial settings of the positions of the nine categorising nodes. During language games, these nodes are continually adjusted to become more and more similar to other agents' categorising nodes.

It is worth considering that size can be subdivided into small parts (see Fig. 5.20) according to the categorising rules based on the GNG network described above.

[5] Red curves: !S(small), !M(medium) and !L(large); Blue curves: !V_!S(very-small), !V_!M(very-medium) and !V_!L(very-large); Green curves: !S_!S(small-small), !M_!S(medium-small), !L_!S(large-small), !S_!M(small-medium), !M_!M(medium-medium), !L_!M(large-medium), !S_!L(small-large), !M_!L(medium-large) and !L_!L(large-large).

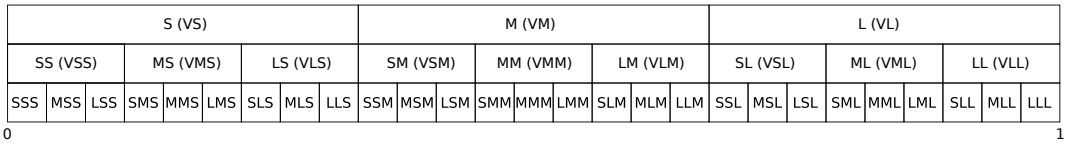


FIGURE 5.20: The subdivision of sizes

5.2.5 Conclusion

The exaggeration feature of language can be utilised not only to expand design space and generate new dynamic categories but also to differentiate existing categories into more specific categories or combine them to obtain some new intermediate categories. For the former, "very large" and "very small" extend the concepts of "large" and "small"; for the latter, "very medium" narrows the concept of "medium"; and "small medium" is associated with a new bridging area between "small" and "medium". Therefore, new interesting exaggerated concepts such as "so large duck in a very small harbour" would be generated using context free grammar. Other dimensions such as colour and shape could also be exaggerated using this method.

5.3 Incongruity

The aim of this experiment is to explore design creativity using the feature of compositional language, incongruity. Incongruent sequences could occur by breaking self-similarity and continuous repetition, e.g. "balabalabala" → "balabalaqila". Incongruent topics can also be generated by breaking normal combinations of utterances, which may represent the relationship between objects and background, and recombining them based on inverse weighted random choice.

5.3.1 Subjects

A simplified landscape is taken as the reference for generating incongruent compositions based on the relationships of objects (circle, square and triangle) and contexts (sky and ground) (see Fig. 5.21).

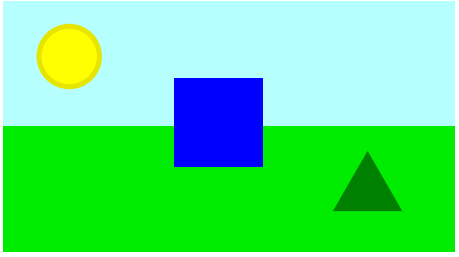


FIGURE 5.21: A simplified scene

TABLE 5.9: The data of scene generated using normal random variables (h: height of ground, c: circle, s: square, t: triangle)

Feature	Min	Max	Mean	Deviation	Sample 1	Sample 2
h	0.01	1	1/2	1/24	0.465	0.517
x_c	0.01	1	1/4	1/24	0.316	0.253
y_c	0.01	1	h+(1-h)/2	(1-h)/24	0.735	0.764
r_c	0.01	1/2	(1-h)/4	(1-h)/24	0.131	0.076
x_s	0.01	1	1/2	1/24	0.484	0.511
y_s	0.01	1	h	min((1-h)/24, h/24)	0.450	0.500
r_s	0.01	1/2	min((1-h)/2, h/2)	min((1-h)/24, h/24)	0.110	0.206
x_t	0.01	1	3/4	1/24	0.752	0.742
y_t	0.01	1	h/2	h/24	0.240	0.257
r_t	0.01	1/2	h/3.5	h/24	0.108	0.163

The original scene is a unit square with corners at the four points (0, 0), (1, 0), (0, 1), and (1, 1). The scene can be scaled unevenly to facilitate visualisation, but the proportion of each object in the scene retains the same. A number of features consisting of the height(h) of green "ground", the coordinates of circle (x_c, y_c), the coordinates of square (x_s, y_s), the coordinates of triangle (x_t, y_t), and the "r" of these three shapes (r_c: the radius of the circle, r_s: the radius of the inscribed circle of the square, r_t: the radius of the circumcircle of the triangle) are generated as data samples which are a little different for each guessing game since the data are obtained via normal random generation (see Table 5.9, sample 1 is a little different from sample 2). This is used to simulate real-like data.

5.3.2 Implementation

This experiment is implemented using both guessing games evolving domain language (i.e., the representation of relations between shapes and backgrounds) and generation games producing incongruent scenes via inverse weighted random choice based on the DIFI framework (see Fig. 5.22).

A scene can be represented via hierarchical representations of the relations between a number of features using context free grammar (see Table 5.10) based on weighted graph networks. For example, a hierarchical representation could be (and ($>$ y_c r_c) ($<$ r_c ($-$ h y_c))), which denotes an incongruent scene: "sun (i.e., circle) in the ground".

5.3.2.1 Representation of Incongruity

The incongruence of the relationship between objects and context is tested using language games —guessing games for recognition and generation games for incongruent

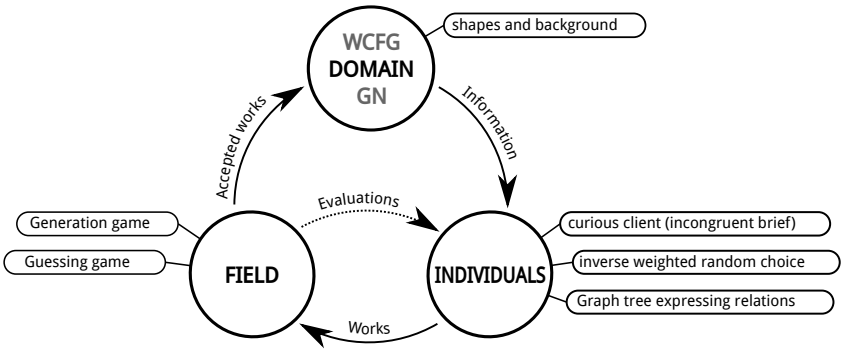


FIGURE 5.22: The Domain-Individual-Field-Interaction (DIFI) framework of exploring incongruity

TABLE 5.10: Basic rules of representing a scene

No.	Rule	Expansion
0	operation	(operator var var)
1	operator	<
2	operator	and
3	operator	+
4	operator	-
5	var	y
6	var	r
7	var	h
8	var	operation
9	object	circle
10	object	square
11	object	triangle
12	relation	(object operation)

generation. Incongruence relies on expectations from context with particular shapes related to low probability. Building the relationships between shapes and context using Equation 35 based on probability theory can be used to generate incongruence.

$$P(S \mid C) < I_{th} \tag{35}$$

S denotes subject such as shape or colour; C represents context such as a scene or a group of harmony shapes or colours; and I_{th} is the threshold of incongruence. A relation is incongruent when its probability is less than I_{th}. The procedure of finding incongruent subject(s) is described in Algorithm 15. Given $subjects = [circle, triangle, square]$, $relation(subject, context) = \{coordY_{subject} < height_{ground}\}$, $threshold_{incongruence} = 0.2$, the incongruent subject could be *circle* (see No. 2 in Table 5.15).

The meaning of incongruence is relative. It is not enough that $P(S \mid C)$ is low; there must also exist some S' such that $P(S' \mid C)$ is high (i.e., with high confidence of congruence) (see Equation 36) to compare with incongruence.

Algorithm 15 Finding Incongruity

```
1: function FINDINCONGRUITY(subjects, context, threshold_incongruence)
2:     incongruentSubjects ← emptyList
3:     for i = 1 → length(subjects) do
4:         if probability(relation(subjects_i, context)) < threshold_incongruence then
5:             incongruentSubjects ← append(incongruentSubjects, subjects_i)
6:         end if
7:     end for
8:     return incongruentSubjects
9: end function
```

$$P(S' \mid C) \geq I_{th} \tag{36}$$

An incongruous composition could be generated by crossing over several consistent compositions. For example, the incongruent composition, "dark triangle and circle + light background", is produced by recombining "dark", "triangle", "circle", "light" and "background" from two consistent compositions—"light triangle and square + dark background" and "dark eclipse and circle + light background".

Incongruity can also be represented by different contexts based on the same subject (see Equation 37). For example, "sea supporting a house" instead of "land supporting a house" keeps the object unchanged but replaces the context, "land", with "sea" to become incongruent.

$$P(C \mid S) < I_{th} \tag{37}$$

Some examples of representing incongruent scenes and incongruent shapes are described as follows. As can be seen from Fig. 5.23, incongruent scenes occur by replacing circle with triangle or yellow cloud (see Equation 38). The scenery could be interpreted by describing the colours and locations of both shapes and background (see Table 5.11) and their relations (see Table 5.12).

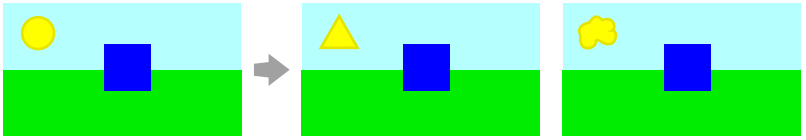


FIGURE 5.23: Example of incongruent scenes 1

$$incongruence = \begin{cases} F & \text{if } P(circle_{yellow} \mid C) \geq I_{th} \\ T & \text{if } P(triangle_{yellow} \mid C) < I_{th} \\ T & \text{if } P(cloud_{yellow} \mid C) < I_{th} \end{cases} \tag{38}$$

TABLE 5.11: The features of scenes 1

Colour	**Shape**	**Background**	**Location**
yellow/orange	circle		top-left
black/blue	square		center
light blue/white...		sky	top
green/red		ground	bottom

TABLE 5.12: The relations of items in scenes 1

Relations	***Circle***	***Square***	***Sky***	***Ground***
Circle		separate	in	separate
Square	separate		intersect	intersect
Sky	contain	intersect		touch
Ground	separate	intersect	touch	

Incongruent scenery may also be generated by replacing green triangle with circle (see Fig. 5.24, Equation 39). The scenery can be represented using the colours and locations of both shapes and background (see Table 5.13) and their relations (see Table 5.14).

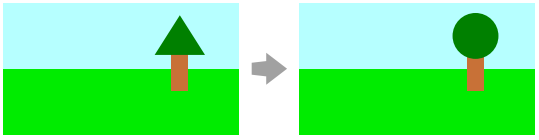


FIGURE 5.24: Example of incongruent scenes 2

$$incongruence = \begin{cases} F & \text{if } P(triangle \mid C) \geq I_{th} \\ T & \text{if } P(circle \mid C) < I_{th} \end{cases} \tag{39}$$

TABLE 5.13: The features of scenes 2

Colour	**Shape**	**Background**	**Location**
green	circle		top-right
brown	rectangle		middle-right
light blue/white...		sky	top
green/red		ground	bottom

TABLE 5.14: The relations of items in scenes 2

Relations	***Circle***	***Rectangle***	***Sky***	***Ground***
Circle		touch	in	separate
Rectangle	touch		intersect	intersect
Sky	contain	intersect		touch
Ground	separate	intersect	touch	

Incongruent shapes may be generated by replacing one shape in a group of cornered shapes with a rounded shape (see Fig. 5.25, Equations 40, 41). The difference between two shapes can be compared without the effect of scale and rotation via Scale-Invariant

Feature Transform (SIFT) and Harris Corner Detection (HCD) in OpenCV (Bradski et al., 2000).

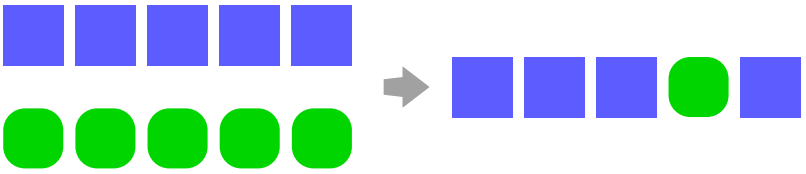


FIGURE 5.25: The incongruity of shapes

$$\{A, B\} \otimes \{C, D\} \rightarrow \{C, B\} \tag{40}$$

$$\{S_{cornered}, C_{cornered}\} \otimes \{S_{rounded}, C_{rounded}\} \rightarrow \{S_{rounded}, C_{cornered}\} \tag{41}$$

5.3.2.2 Procedure of Language Games

The procedure of language games is from guessing games evolving congruent expressions to generation games developing incongruent artworks.

- Guessing Games

  The process of a guessing game is illustrated in Figure 5.26. Initially, a few basic rules (see Table 5.10) are predefined for generating random expressions denoting the relations between object (circle, square and triangle) and context ("sky" and "ground"). In each guessing game, the speaker-agent generates a random expression using these rules via context free grammar to match the objects. For example, an object, square, is ($x_s = 0.484$, $y_s = 0.450$, $r_s = 0.110$) with $h = 0.465$; and the generated s-expression is (and (< y h) (< r y)). The result, which is obtained by substituting values in this expression, is true if the expression matches the object. After all objects are tested, if the expression matches at least one of these objects but not all of them, an object will be selected from the matched object(s) randomly, e.g. [square, triangle] → square → appropriate topic: "(square (and (< y h) (< r y)))". Otherwise, the speaker-agent will generate another expression to match the criteria. Then the appropriate topic will be mapped to a list of utterances such as ("fi" "pa" "tepa" "pa" "mu" "kini" "hagi" "aq" "pa" "mu" "rola" "ya" "aq" "aq") via weighted random choice of existing utterances or generating new utterances. In the list of utterances, "pa" represents "(" while "aq" represents ")".

  After finishing the production, the speaker-agent tells the listener-agent the utterances. The listener-agent will start to parse the utterances to the combination of object and the corresponding expression via weighted random choice of existing relations between these elements and utterances. If some parts of the utterances

do not exist, the listener will match these parts to random elements and regulate them to satisfy the format of expression.

Then the guessed topic is sent to the speaker-agent who will compare it with its own topic. If the two topics are the same, both speaker and listener learn to strengthen the weights of the associations between each utterance and element. Otherwise, the listener unlearns the relations between guessed topic and utterances by weakening the related weights and learns the relations between the speaker's topic and utterances.

After a number of guessing games, both the instances (see Table 5.15) of Rule 12 in Table 5.10 and the utterances representing terminal-elements including <, "and", "+", "-", "y", "r", "h", "circle", "square" and "triangle" are generated and accumulated. For example, an agent could represent the operator "+" with three weighted utterances including "bi: 0.783", "poxi: 0.511" and "di: 0.198" that means "bi" is likely to be selected more than "poxi" and "di" to represent "+". Some of these weights will continue to be adjusted after each guessing game, leading to more successful communication next time. When the success rate is above the threshold (0.65) of the guessing game, agents will start to play generation games by changing their roles to client and designer.

- Generation Games

  In each generation game (see Fig. 5.27), a client-agent and one or more designer-agent(s) are randomly selected from the agents. First, the client-agent selects an instance of Rule "operation → (operator var var)" (see the relations in Table 5.15) such as "(< y h)". Then the selected instance is combined with the meaning, "incongruent", to become "incongruent (< y h)", which will be mapped to utterances such as ("inco" "pa" "tepa" "mu" "kini" "aq"). After completing the generation of utterances, the client-agent sends the requirement (utterances) to the designer-agent(s).

  Then a designer-agent starts to parse the requirement into two parts. The first part is the interesting type[6] while the remainder is the relevant expression obtained via weighted random choice of the relations between the utterances and elements. If the expression is a part of the instances of Rule (see Table 5.15), "relation → (object operation) → (circle/square/triangle operation)", the object will be selected via inverse weighted random choice of the relevant instances. For example, "circle" would have much more chance of being selected than "square" and "triangle" from the instance-rule, "(object (< y h)) → (((0.0 circle) (0.714 square) (0.758 triangle)) (< y h))", to satisfy the client's requirement of incongruity. If the expression is not a member of the existing instances, a new instance-rule will be generated by calculating the results of this expression with the related variables

[6]In this experiment, only incongruence is considered. In the nex experiment, some other interesting types including extensible, exaggerational, unusual and strange are added.

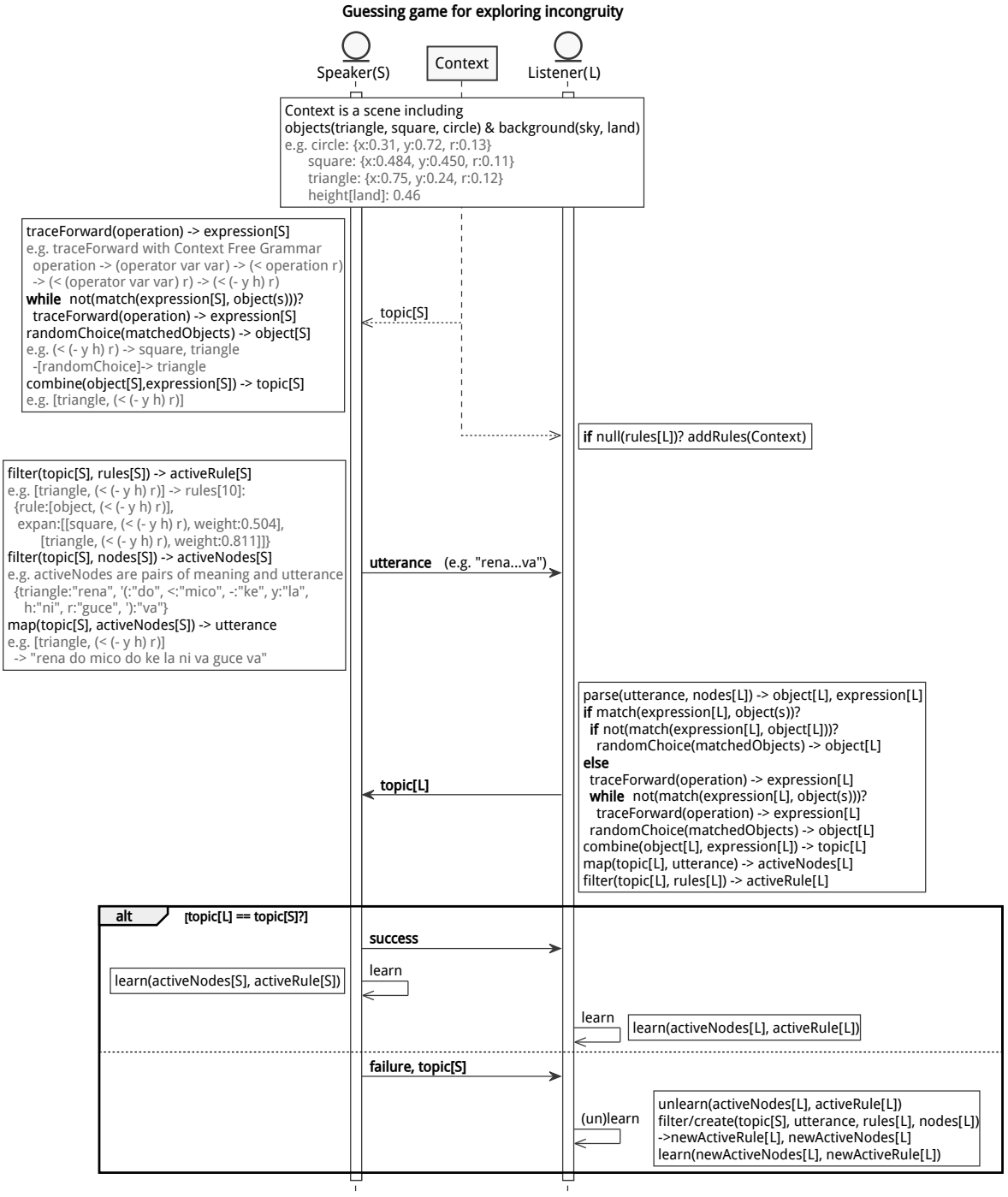


FIGURE 5.26: Guessing game for exploring incongruity

extracted from a scene-sample; and an incongruent object will be selected from the calculated results.

After the designer-agent has finished finding the incongruent topic, it will send the topic to the client-agent who will check whether the topic is incongruent or not by comparing its own result of inverse weighted random choice of its instance-rule such as "(< y h)".

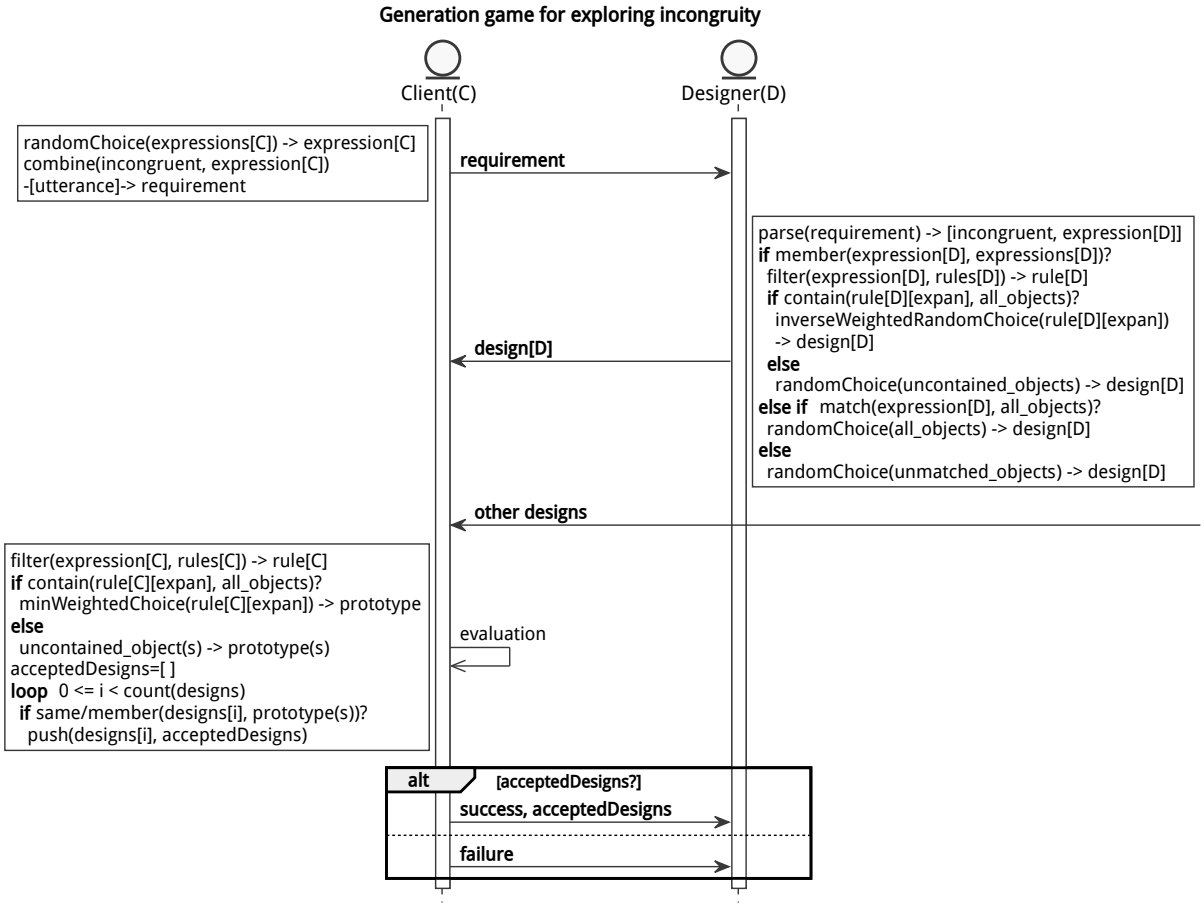


FIGURE 5.27: Generation game for exploring incongruity

5.3.3 Results

Results include the results of guessing games for evolving languages via context free grammar, and that of generation games for generating incongruent compositions.

5.3.3.1 The Results of Guessing Games

Compositional languages representing the relations between objects and contexts have evolved based on graph networks through guessing games. The structure of a graph network for developing the rules of context free grammar is combined with a number of nodes and their relations. Each node has a property of "utterance" containing a few of utterances with different weights which have been changed and evolved in guessing games; for example, {node: <, utterances: ((0.741 "zafe") (0.848 "vi") (0.77 "tijo") (0.825 "wafe") (0.864 "rewa"))}. Each relation includes a list of rule-nodes and its expansion containing a few expanded-nodes with weights, which are also generated and evolved, e.g. {rule: operator, expansions: ((0.01 <) (0.01 and) (0.01 +) (0.01 -))}. Another more complicated example is that the rule, (object (< (- y h) r)), is expanded

TABLE 5.15: An agent's relation rules
(Cir.: circle, Sq.: square, Tri.: triangle, int: intersect, gr.: ground)

No.	Relation	Meaning	Cir.	Sq.	Tri.
0	(< (- Y H) R)	int/in gr.		.504	.811
1	(< H Y)	over/int gr.	.892	.518	
2	(< Y H)	int/in gr.		.714	.758
3	(< R (- Y R))	over bottom	.795	.547	.108
4	(< (+ R H) Y)	over gr.	.913		
5	(< R (- Y H))	over gr.	.900		
6	(AND (< H Y) (< R (- Y H)))	over gr.	.875		
7	(AND (< H Y) (< R (- Y R)))	over/int gr.	.870	.732	
8	(< (- Y (- (+ R H) H)) H)	in/int gr.		.477	.657
9	(< H (+ Y Y))	int/over m-gr.	.592	.629	
10	(AND (< (- Y H) R) (< H Y))	int gr.		.789	
11	(AND (< R (- Y R)) (< R (- Y H)))	over gr.	.968		
12	(AND (< R (- Y R)) (< Y H))	int/in gr.		.855	.526
13	(< H (+ Y R))	over/int gr.	.697	.696	
14	(AND (< R (- Y H)) (< H Y))	over gr.	.845		
15	(AND (< H Y) (< R (+ Y R)))	over/int gr.	.706	.467	
16	(AND (< R (+ Y R)) (< Y H))	in/int gr.		.415	.752
17	(AND (< H Y) (< R (+ Y H)))	over/int gr.	.714	.277	
18	(< H (- Y R))	over gr.	.690		
19	(AND (< R (- Y R)) (< R (+ Y H)))	over bottom	.716	.328	.109
20	(AND (< R (+ Y H)) (< H Y))	over/int gr.	.773	.414	
21	(AND (< R (+ Y R)) (< R (- Y H)))	over gr.	.764		
22	(< (+ Y (- (+ R H) H)) H)	in gr.			.568

to ((0.504 square (< (- y h) r)) (0.811 triangle (< (- y h) r))), which is similar to context sensitivity.

The context free grammar of each agent is initialised with 13 basic rules (see Table 5.10). Some complicated rules such as "(and (< (- y r) h) (> (+ y r) h))" can be generated using Rule 0 and Rule 8 (see Table 5.10) to represent the intersection of "square" and "ground" (see Fig. 5.21). This meaning has been represented by a similar relation rule, "(and (< (- y h) r) (< h y))", which is generated (see Table 5.15) in a guessing game.

As can be seen from Table 5.15, only square matches Rule 10, which means only square is across "sky" and "ground". All objects match Rule 3, which means the radius is less than the subtraction of y-coord (object-height) with radius. But their weights are different, $W_{circle} > W_{square} > W_{triangle}$. It means that triangle is probably lower than square, which is probably lower than circle regarding their relative similar radius. Here, weight is analogised with probability. So, the result of different probabilities provide a basis for evolving incongruent language in the next stage running generation games.

5.3.3.2 The Results of Generation Games

The success rate of generation games is very high (> 0.9) from the early stage partially because only one type of interesting requirement, incongruence, needs to be satisfied; and relevant relations may be captured more easily via the representations with long sequences that can provide abundant information for filtering topics. In this experiment, the process of learning/unlearning is not implemented in generation games because the client's evaluation is fixed so that it always compares the results evolved in the previous stage with guessing games. In a future experiment, the evaluation would become dynamic by changing the incongruent combinations accumulated in generation games.

Incongruent scenes are generated by inverse weighted random choice. In each generation game, the client-agent randomly selects one of existing relation rules (see Table 5.15), i.e., expressions, to generate compositional meaning, and map it to a relevant requirement, i.e., a sequence of utterances. A designer-agent then parses this requirement to related CFG rule, selects one of the connections of the rule via inverse weighted random choice, and generates a new design opposite to the congruent rule. For example, when $P(circle \mid sky) > I_{th}$ and $P(rectangle \mid sky) << I_{th}$, normally, the combination of circle and sky is likely to be selected. On the contrary, in order to obtain an incongruent composition, the combination of rectangle and sky is selected due to its low probability (see Fig. 5.28).

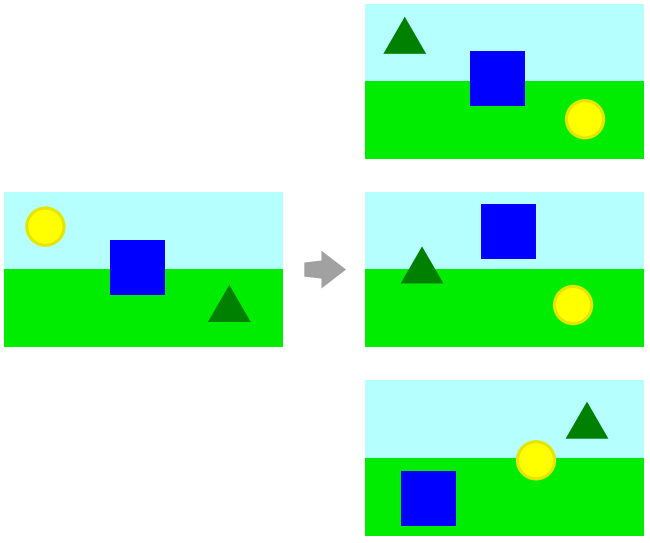


FIGURE 5.28: The illustration of some incongruent results of generation games

5.3.4 Conclusion

Incongruent composition can be produced by inverse weighted random choice of rules generated via context free grammar. To begin with, the normal rules associated with objects and context are collected through guessing games. Then, in generation games,

incongruent compositions are generated by using inverse weighted random choice of the accumulated relation rules, i.e., incongruent artworks are generated by selecting the combinations of objects and contexts with low probability.

5.4 Extensibility and Other Features

The aim of this experiment is to explore design creativity using the feature of compositional language, extensibility. Representations can be expanded using context free grammar to obtain interesting topics such as elaboration, unusualness and strangeness.

To explore interesting replacement and combination in a scene (see Fig. 5.29), designer-agents will learn the meanings of elaboration (extensibility, or describing an object with numbers of other objects), unusualness (the replacement of objects), strangeness (the replacement of relations), incongruence (the mismatch between object and background) and exaggeration (scaling up/down size). These meanings are provided by client-agents using relevant utterances in generation games.

5.4.1 Subjects

A simplified nature scene (see Fig. 5.29) is used as the subject of this experiment. To simplify the operation of the items on the scene and their relations, only regular triangle, rectangle, and circle are used to represent house, tree, boat, fish, sun and land. Most of the shapes touch each other except that the boat intersects with the sea.

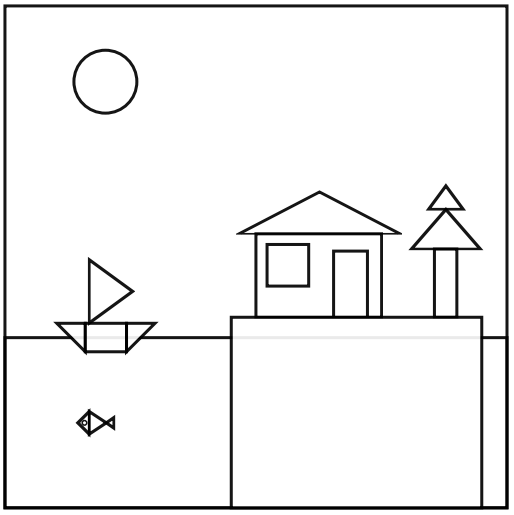


FIGURE 5.29: A sample of scene

5.4.2 Implementation

Interesting topics such as elaborated objects are produced by tracing and generating rules with related utterances and compositional items based on context free grammar and graph networks. A domain language evolves in guessing games; and interesting novel scenes are generated via elaboration, composition and replacement of original topological relations in generation games (see Fig. 5.30).

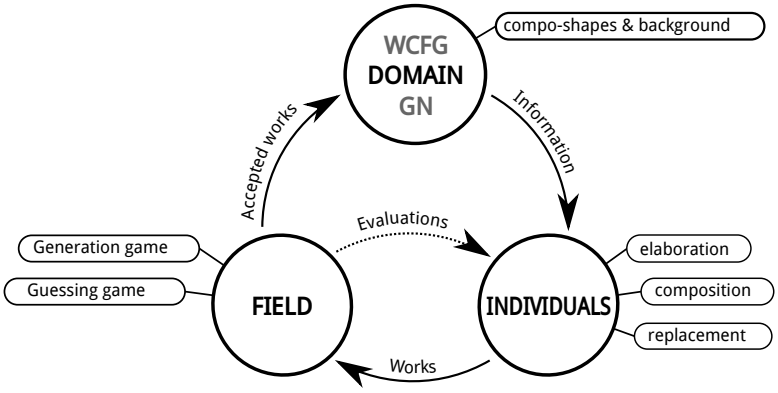


FIGURE 5.30: The Domain-Individual-Field-Interaction (DIFI) framework of exploring extensibility & other features

5.4.2.1 Initialisation

Before running language games, the features of the sample-scene are extracted in four steps. First, basic objects such as roof, wall, window and door are associated with appropriate qualified-features and represented by randomly generated utterances. For example, the features of a roof are composed in a dictionary {shape: triangle, direction: up, size: medium} and its utterance could be "guzu" with initial weight 0.01. Then, related pairs of the objects will be collected if their topological relationship is "touch" or "in/contain", e.g. the pair of roof and wall. Thirdly, each connection of these pair-objects will be integrated into a composed object such as "house" which is obtained from the chain, "(touch roof (contain wall (window door)))". Finally, the relations between these compositional objects and related context will be developed. For example, the relation between house and land is "(on house land)" and that between sun and sky is "(in sun sky)".

5.4.2.2 Guessing Game

In each guessing game (see Fig. 5.31), both speaker-agent and listener-agent are randomly selected from 6 agents. First, the speaker-agent selects a topic such as fish, roof or sky randomly and selects a corresponding utterance via weighted random choice. When

the utterance is selected, the speaker-agent sends this utterance to the listener-agent who will parse it to a guessed topic.

Then, the listener-agent sends the guessed topic back to the speaker-agent. If it is the same as the speaker's topic, both speaker-agent and listener-agent strengthen the weight of the relation between the utterance and the topic. Otherwise, the listener firstly weakens the weight of the relation between the utterance and the guessed topic and then strengthens that of the utterance and the speaker's selected topic.

After a number of guessing games, when the success rate is above the threshold of guessing game (0.8), the game will be changed to a generation game for generating novel utterances to find interesting designs. Due to the application of compositional languages, the topics such as house are represented with two kinds of utterances—compositional utterances generated from its components and their relations, and a holistic word representing house.

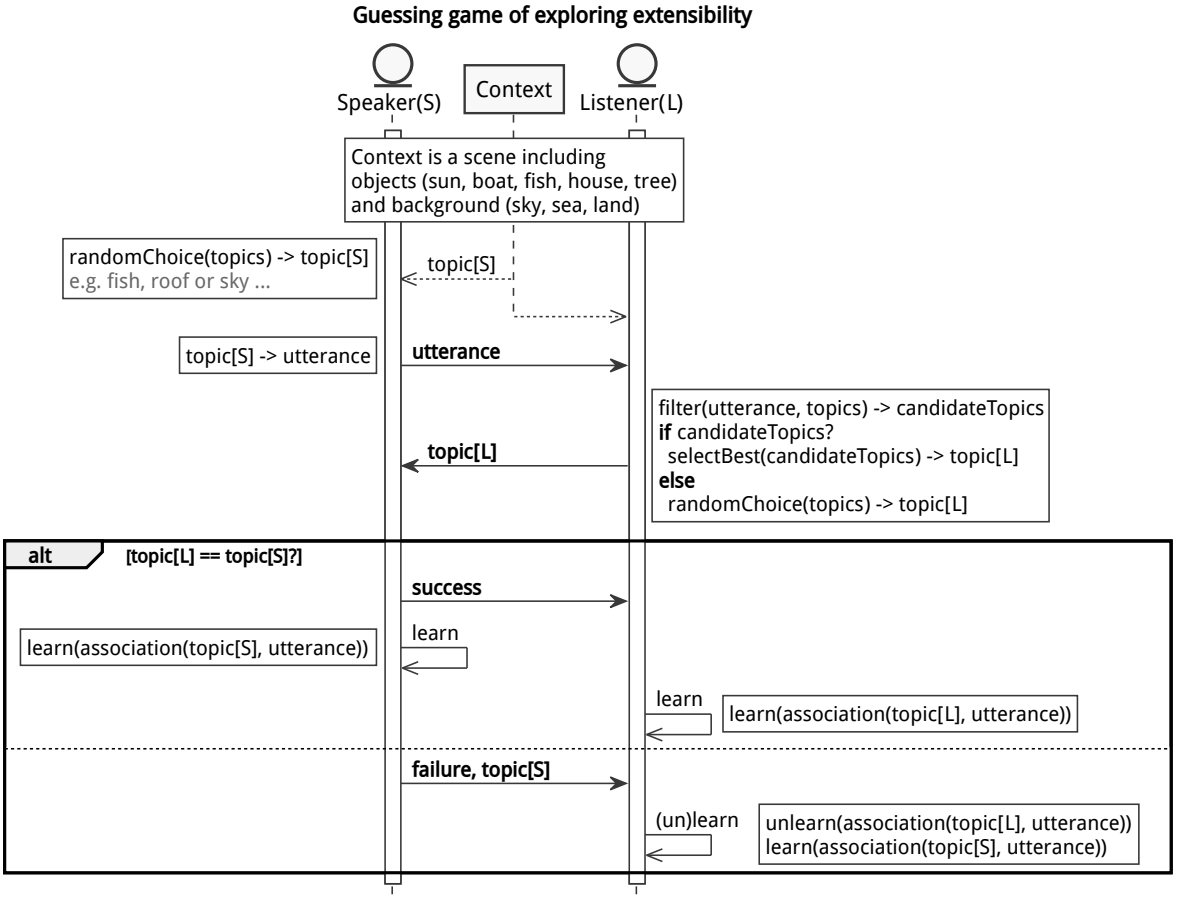


FIGURE 5.31: Guessing game for exploring extensibility & other features

5.4.2.3 Generation Game

In each generation game (see Fig. 5.32 [7]), the first agent always plays the role of client while 3 or 4 designers will be selected randomly from the other agents. The client-agent requests a new scene with an interesting type such as incongruence by providing designers an utterance, which is generated randomly in advance, then converged to certain utterance during generation games.

In the early stage of generation games, designers randomly select one of the following topics: object (e.g. bow), compositional object (e.g. boat), the relations between two objects (e.g. "(on up) roof wall"), and the relationship between object and background (e.g. "(in sun sky)"). Then an instance is obtained by tracing forward the topic (e.g. object → door → (rectangle up medium)). The instance could be changed partially or wholly according to a randomly generated number (-1 <= number < the length of expanded topic). If the number is -1, the whole instance will be changed to a new instance with the same or more complicated details. For instance, "rectangle up medium" is converted to "triangle right small" via Rule 0 (see Table 5.16); and it can be expanded to "(on left) (rectangle up medium) (triangle right small)" via Rule 1 (see Table 5.16). Otherwise, the number is taken as the index of an item of the instance. This item will be replaced with a new item via relevant rules including Rule 3, 4, 5, 6, 7 and 9 (see Table 5.16). For example, "rectangle" could be replaced with "triangle" by tracing backward "rectangle" to "shape" then tracing forward "shape" to "triangle". When the change is finished, the designer-agent provides the design, i.e., the modified instance, to the client-agent.

If the design is accepted by the client-agent, the designer-agent memorises the association (see Table 5.17) between the selected topic and the changed position and the client's utterance, and strengthens the weight of the relations. Otherwise the weight of the relations will be weakened. After at least one relation is memorised, a designer will first search the memorised relation(s) to match the client's requirements in next generation games.

5.4.2.4 Operations on Topics

In this experiment, three main operations on topics are used. They are replacement, elaboration and composition. Replacement includes replacing operators and operands (e.g. "on tree land" → "in tree land" involves changing the operator while "on tree land" → "on tree sea" involves the replacement of sea with land). Elaboration enhances the abundance of the original topics by expanding the topic to a new relation of new object

[7] A path from refer-instance to design-instance instead of design-instance is stored in an agent's memory when game succeeds. It is for convenience in generating a new design via mutating a path using context free grammar in next generation game.

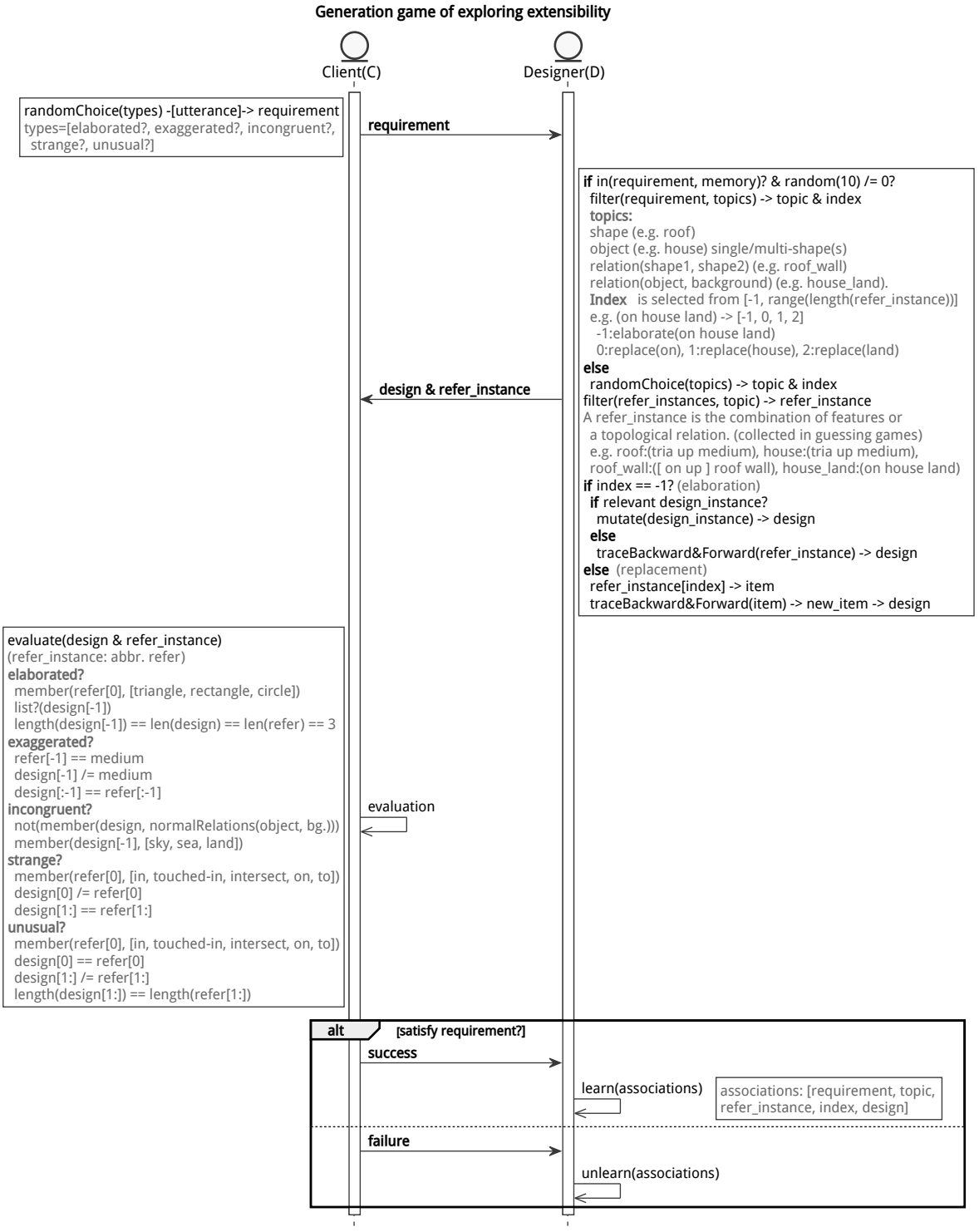


FIGURE 5.32: Generation game for exploring extensibility & other features

with the topic (e.g. "on roof wall" → "on (in (round window) roof) wall"). Composition is used to generate a compositional object such as a house with roof, wall, window and door, to represent a long phrase describing a scene (e.g. (on house land) → (on (and (on roof wall) (in window wall) (in door wall)) land)), and to combine modifier and size in a exaggerated form. Consequently, a hierarchical structure of representing the sample

TABLE 5.16: The rules of representing a scene
(tin: touched-in, int: intersect)

No.	Rule	Expansion	Example
0	object	(shape direction size)	bow → (triangle left medium)
1	object	(location object object)	bow-hull → ((on left) bow hull)
2	object	(and object object . . .)	boat → (and bow-hull stern-hull sail-hull)
3	shape	triangle / rectangle / ellipse	
4	direction	up / down / right / left	
5	direction	(direction direction)	(up right)
6	size	small / medium / large	
7	size	(very size)	(very (very small))
8	location	(relation direction)	(in (up right))
9	relation	in / tin / int / on / to	

TABLE 5.17: An example of associations for (un)learning after each generation game

Association	Example
Requirement	elaborated? → utterance: "po"
Topic	shape
Refer-instance	(tria up medium)
Index	-1
Design	((on (right up)) (tria up medium) (tria left medium))

scene and newly generated novel scenes are developed.

5.4.3 Results

Before guessing games, each agent generates a number of instance-rules (see Table 5.18) consisting of all single objects (such as roof and window) with their features (e.g. roof → {shape: triangle, direction: up, size: medium}), all touched pair-objects such as "wall & roof" and their relations (e.g. "wall & roof" → ([on down] wall roof)), four components (fish, boat, tree and house) and their features (e.g. fish → {shape: triangle, direction: left, size: medium}) and their relationship with the background (e.g. "fish & sea" → (in fish sea)). Although each agent's initialised instance-rules are the same, the associated utterances are different.

5.4.3.1 Results of Guessing Games

In one of the simulations, the success rate reaches the threshold of the guessing game (0.8) after nearly 3921 iterations (see Fig. 5.33). As can be seen from Fig. 5.34, the points outside the diagonal are very sparse, which means that most of the utterances representing elements are distinguished from each other very well.

TABLE 5.18: An agent's initial rules and instance-rules
(obj: object, wal: wall, rof: roof, win: window, dor: door, trk: trunk, cro: crown, crw: crown2, hul: hull, pop: poop, sal: sail, fad: fish-head, fdy: fish-body, fil: fish-tail, fye: fish-eye, tria: triangle, compo-obj: compositional object, int: intersect, lan: land, obj-bg: object-background, bg: background, sce: scene)

No.	Rule	Expansion
0	size	(very size) max-depth:2
1	direction	(direction direction) max-depth:1
2	obj	(relation obj obj) max-depth:1
3	obj	sun/wal/rof/win/dor/trk/cro/crw/hul/bow/pop/sal/fad/fdy/fil/fye
4	rof	(tria up medium)
...	...	...
5	pair-obj	(relation obj obj)
6	pair-obj	wal.rof/rof.wal/win.wal/dor.wal/trk.cro/cro.trk/cro.crw/crw.cro
7	pair-obj	hul.bow/hul.pop/hul.sal/bow.hul/pop.hul/sal.hul
8	pair-obj	fad.fdy/fdy.fad/fdy.fil/fil.fdy/fye.fad
9	wal.rof	([on down] wal rof)
...	...	...
10	(components fish)	(and fad.fdy fdy.fil fye.fad)
11	(components boat)	(and sal.hul bow.hul hul.pop)
12	(components tree)	(and cro.trk crw.cro)
13	(components house)	(and rof.wal dor.wal win.wal)
14	compo-obj	fish/boat/tree/house
15	fish	(tria left medium)
...	...	...
16	fish.sea	(in fish sea)
17	boat.sea	(int boat sea)
18	house.lan	(on house lan)
19	sun.sky	(in sun sky)
20	tree.lan	(on tree lan)
21	obj-bg	sun.sky/house.lan/tree.lan/fish.sea/boat.sea
22	bg	sce/sky/sea/lan

5.4.3.2 Results of Generation Games

When generation games are played by a client and 3 or 4 designers about 1000 times, the success rate surpasses 0.9 (see Fig. 5.35).

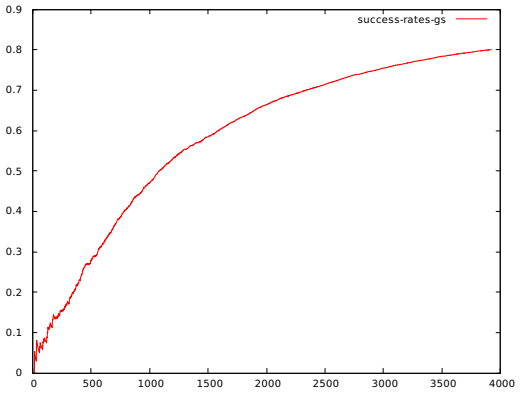


FIGURE 5.33: the success rates of guessing games

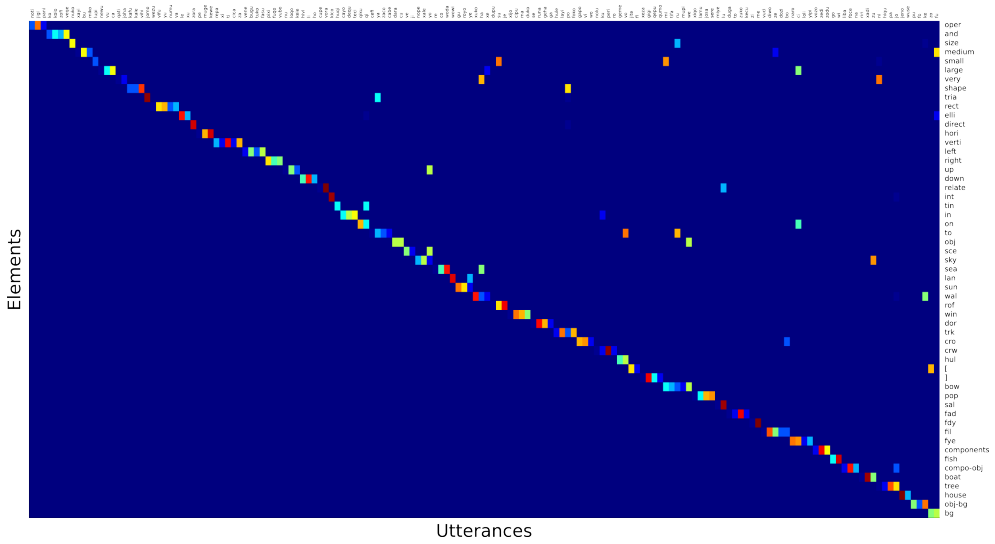


FIGURE 5.34: The relations between elements and utterances from one agent

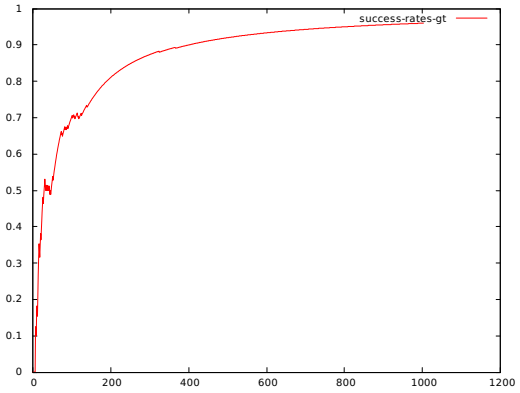


FIGURE 5.35: the success rates of generation games

Some results for novel designs related to extensibility and other interesting topics are illustrated in Figures 5.36, 5.37, 5.38, 5.39 and 5.40. For example, the original shape of the roof is an up triangle with medium size. After elaboration at one level with Rule "object → (location object object)", the roof is changed to more abundant compositional shapes such as "((on (right up)) (triangle up medium) (triangle left medium))" (see Table 5.19).

To obtain other novel types, several different criteria are established. For instance, a new generated design could be "unusual" when one of its operands differs from the original topic; for example, "(in fish sea) → (in fish sky)" via changing "sea" to "sky" (see Table 5.20). It could also be "strange" when its operator is not the same as the original one; for example, "((on up) roof wall) → ((in up) roof wall)" via replacing "on" with "in" (see Table 5.21). The design might also be an exaggerated topic if its size is different from the original one (see Table 5.22). It is likely to become incongruent when the object or the context or the relation between object and context is replaced with other elements; for example, "(intersect boat sea) → (intersect boat land)" (see Table 5.23).

Therefore, different interesting topics can be differentiated via various criteria although some of them are somewhat similar to each other or even the same; for example, the result of unusualness, "(in fish sea) → (in fish sky)", is also incongruent.

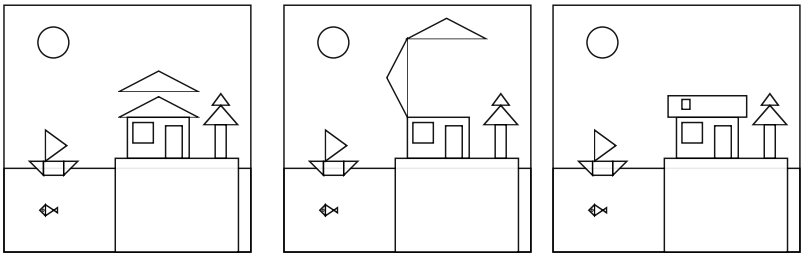


FIGURE 5.36: Elaborated samples

TABLE 5.19: The results of elaboration

No.	Rule		Expansion
0	(tria up medium)	→	((on (right up)) (tria up medium) (tria left medium))
1	(tria up medium)	→	((in right) (tria right medium) (tria (up left) medium))
2	(tria up medium)	→	((in (up left)) (rect up medium) (rect right medium))
3	(tria up medium)	→	((to up) (tria up medium) (tria up medium))
4	(tria up medium)	→	((int (left right)) (tria right medium) (tria left medium))

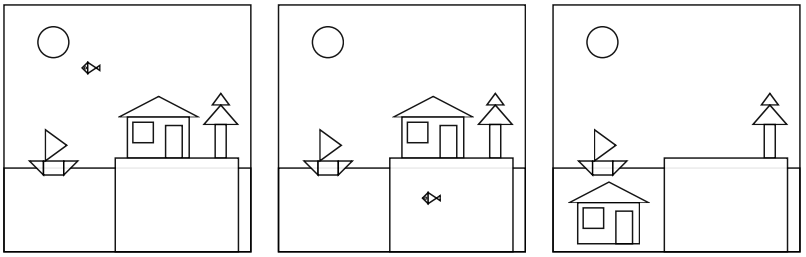


FIGURE 5.37: Unusual samples

TABLE 5.20: The results of unusualness

No.	Rule		Expansion
0	(in fish sea)	→	(in fish sky)
1	(in fish sea)	→	(in boat sea)
2	(in fish sea)	→	(in fish lan)
3	(in fish sea)	→	(in house sea)
4	(in fish sea)	→	(in fish sce)
5	(in fish sea)	→	(in tree sea)

5.4.4 Discussion

Although the experiment is completed successfully, some aspects including the satisfaction of different requirements, the collection of the information of sample scenes, and the evaluation of generated products related with different topics should be discussed and enhanced in future experiments.

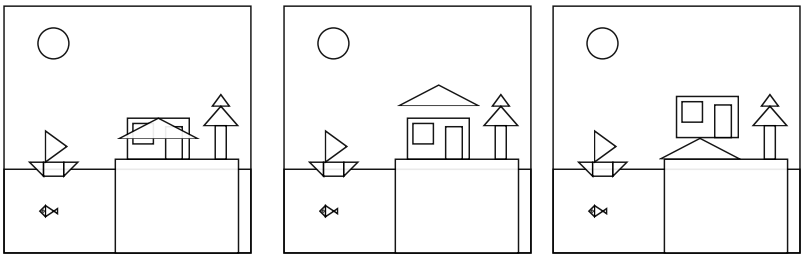


FIGURE 5.38: Strange samples

TABLE 5.21: The results of strangeness

No.	Rule		Expansion
0	((on up) rof wal)	→	((in up) rof wal)
1	((on up) rof wal)	→	((on (down right)) rof wal)
2	((on up) rof wal)	→	((on (up right)) rof wal)
3	((on up) rof wal)	→	((on (down left)) rof wal)
4	((on up) rof wal)	→	((int up) rof wal)
5	((on up) rof wal)	→	((tin up) rof wal)
6	((on up) rof wal)	→	((on right) rof wal)
7	((on up) rof wal)	→	((to up) rof wal)

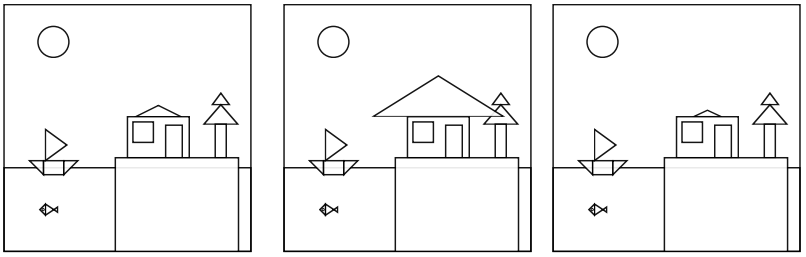


FIGURE 5.39: Exaggerational samples

TABLE 5.22: The results of exaggeration

No.	Rule		Expansion
0	(tria up medium)	→	(tria up (very (very large)))
1	(tria up medium)	→	(tria up (very (very small)))
2	(tria up medium)	→	(tria up (very (very medium)))
3	(tria up medium)	→	(tria up (very large))
4	(tria up medium)	→	(tria up (very medium))
5	(tria up medium)	→	(tria up large)
6	(tria up medium)	→	(tria up small)
7	(tria up medium)	→	(tria up (very small))

5.4.4.1 Requirements

Among the interesting requirements, "elaboration" is particularly difficult to be satisfied by only memorising the relation of selected topic and changed position (which is learned to be -1) due to the multiple rules of "object" (see Rule 0, 1 and 2 of Table 5.16), in

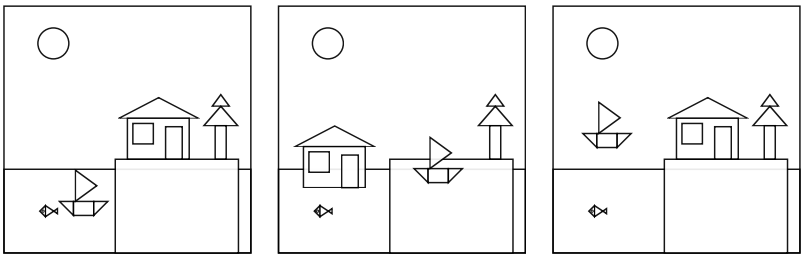


FIGURE 5.40: Incongruent samples

TABLE 5.23: The results of incongruence

No.	Rule		Expansion
0	(int boat sea)	→	(int tree sea)
1	(int boat sea)	→	(int fish sea)
2	(int boat sea)	→	(to boat sea)
3	(int boat sea)	→	(int boat lan)
4	(int boat sea)	→	(int house sea)
5	(int boat sea)	→	(on boat sea)
6	(int boat sea)	→	(tin boat sea)
7	(int boat sea)	→	(int boat sky)
8	(int boat sea)	→	(in boat sea)

which Rule 0 only generates the features of an object without expanding the object to more combinations.

This issue is addressed by using additional memory to store the traced paths of successful modified instances; for example, the path from "triangle up medium" to "(in left) (triangle (up right) medium) (ellipse right medium)" is [-34, -114, 23, 20, 8, 10, 17, 108, 117, 28, 37]. In the path, a negative number means tracing backward. In the following generation games, this path (particularly the last part) would be mutated a little via gradient mutation (i.e., the mutation rate of the next item is higher than that of the previous one) and used for a new design if the instance is the same as the memorised instance-reference of this path. In future generation games, the paths would be analysed to find their shared part matching a certain rule (such as Rule 1 rather than Rule 0) to satisfy the requirement, "elaboration", more easily and flexibly.

This problem can also be resolved by measuring both max-depth and max-breadth of a design (i.e., a nested list of s-expressions) because the complexity of the elaborated instance should be higher than that of the original instance.

On the other hand, the requirements, "strangeness" and "unusualness" are comparably easy to be satisfied due to the utility of s-expressions representing relation rules. The replacement of the first part of an s-expression may match the meaning of "strangeness" while changing the rest of the s-expression could satisfy the brief, "unusualness". Further, "incongruence" is realised by replacing the object with a particular background

or replacing their relations with others. For example, "in boat sky" is generated by replacing "sun" of "in sun sky" with "boat".

5.4.4.2 Sample Scene

In the current experiment, a sample scene (see Fig. 5.29) consists of a number of records. Each record represents the features of an object. For example, "(bow (shape triangle) (x 13) (y 34) (r_x 4) (r_y 4) (angle 1/4) (type object))" consists of the attributes of the bow. Here x and y are the coordinates of the object while r_x and r_y are related to its length and width. The range of angle is from -1 to 1 (i.e., degrees [-180, 180]). So 1/4 means "up right". Here, rx and ry have the same size 4, which means "bow" is simplified to an isosceles right triangle. Based on the data of the objects, the geometrical relations between each are calculated. Then compositional objects are generated in terms of their relations; for example, "house" is combined with "[in left] window wall", "[touched-in right] door wall" and "[on up] roof wall". Finally, the relations between these compositional objects and the background (sky, sea and land) are evaluated, e.g. "on house land".

All the information is embedded in each agent's memory before guessing games and generation games are played. In future experiments, it would be preferable for agents themselves to collect the relevant information piece by piece during language games. In addition, the scene is fixed in each guessing game. The compositional objects could be animated to obtain more real-like geometric data in future experiments. For example, a "boat" can move from the left side of "sea" to the right side and even "sink" to the bottom of "sea", which may change the result of evaluating incongruence.

5.4.4.3 Evaluation

In future experiments, the differences among these requirements would be measured by using cosine-similarity with Holographic Reduced Representations. The terminal measurable elements consist of topological relations (see Table 5.24), directions (see Table 5.25) and sizes (see Table 5.26). If n=5, the result of representing sizes of the objects in the scene is shown in Table 5.27. The meanings of strangeness, unusualness, incongruence, exaggeration and elaboration could be learned by measuring the differences between original instances and the modified or expanded instances based on the dot-product of two sequences obtained from expanded rules as well as their different complexity, which may be evaluated according to the number of levels and the amount of items at each level of a nested list of expanded rules. Therefore, the interesting requirements could be learned rather than being defined in advance.

TABLE 5.24: The rules of representing topological relations

No.	Rule	Expansion	Expansion2
0	start	intersection	
1	operator	increment(inc.) / decrement(dec.)	
2	touched-in	(dec. intersection)	
3	in	(dec. touched-in)	(dec. (dec. intersection))
4	on	(inc. intersection)	
5	to	(inc. on)	(inc. (inc. intersection))

TABLE 5.25: The categorizations of directions

No.	Meaning	Angle(a) (range:[0,1] → degree[0,180])
0	right	(<= 1/8 a 1/8)
1	(up right)	(<= 1/8 a 3/8)
2	up	(<= 3/8 a 5/8)
3	(up left)	(<= 5/8 a 7/8)
4	left	(or (<= 7/8 a 8/8) (<= 8/8 a 7/8))
5	(down left)	(<= 7/8 a 5/8)
6	down	(<= 5/8 a 3/8)
7	(down right)	(<= 3/8 a 1/8)

TABLE 5.26: The rules of representing sizes

No.	Rule	Expansion
0	start	default-size
1	medium	(* default-size 1)
2	small	(/ default-size n)
3	large	(* default-size n)
4	(very (very small))	(/ (/ (/ default-size n) n) n)

TABLE 5.27: The rules of representing sizes for the scene (n=5)

Size-name	Scale	Size	Example
(very (very large))	1	10000	scene
(very large)	1/5	2000	
large	1/25	400	wall
medium	1/125	80	
small	1/625	16	
(very small)	1/3125	16/5	
(very (very small))	1/15625	16/25	fish-eye

The evaluation of creativity may be implemented by measuring the distance (or similarity) between the reference and new designs and the complexity of the designs. The difference related to novelty could be measured from quality to quantity. The real combined features and "imagined" features that are generated via replacement and elaboration might be compared. For example, roof:real (triangle up medium) → roof:imagine (circle up medium) is via replacement while roof:real (triangle up medium) → roof:imagine ((in up) (circle up small) (triangle up medium)) is via elaboration. Their distance can be

calculated by comparing the relevant features between real sequence and "imagined" sequence and measuring the complexity of these sequences. The distance can be measured based on the regulated ranges of directions (see Equation 42 [8]), topological relations (see Equation 43 [9]) and sizes (see Equation 44 [10]). All distances are regulated in the range [-1, 1].

$$\{L, DL, D, DR, R, UR, U, UL, L\} = \{-1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1\} \quad (42)$$

$$\{IN, TIN, INT, ON, TO\} = \{-1, -1/2, 0, 1/2, 1\} \quad (43)$$

$$\{VVS, VS, S, M, L, VL, VVL\} = \{-1, -2/3, -1/3, 0, 1/3, 2/3, 1\} \quad (44)$$

As can be seen from Table 5.28, the distance between an object and its elaboration can be calculated using these equations and the function named "distance-elaboration" as shown below (code in Common Lisp).

```
(defun distance-elaboration (&key (obj '(rect right medium))
    (elaboration '((on up)(tria up small)(rect right medium))))
  ;;'tin: 'touched-in, (second obj): direction
  (setf obj (list (list 'tin (second obj)) obj obj))
  (average (mapcar (lambda (x y) (abs (- x y)))
    (flatten (quantify obj))
    (flatten (quantify elaboration)))))
```

An example of using this function is "obj (rect right medium)" → "((tin right) (rect right medium) (rect right medium))". the direction of relations is the same as the direction of obj (i.e., object). the topological relation is always "tin"(i.e., touched-in), "obj0" is the same as "obj1" which is the original object. Then this expanded object could be used to compare with the elaborated object. The function, "quantify", in the function "distance-elaboration" uses Equations 42, 43 and 44.

Besides measuring the distance-related novelty, the complexity of a nested list (hierarchical structure) of s-expressions can be measured using max-depth and max-breadth. For example, the max-depth of "((IN U) ((INT L) (S1 R S) (S2 U M)) (S3 U L))" is 2 while its max-breadth is 3. Entropy could also be used to measure complexity. Further, predefined interesting criteria in quality could be transformed by generating relevant expressions representing these definitions in quantity that is similar as the relation rules (see Table 5.15) generated in the experiment, Incongruity.

[8] U: up, D: down, R: right, L: left
[9] TIN: touched-in, INT: intersect
[10] V: very, S: small, M: medium, L: large

TABLE 5.28: The distances between objects and their elaborations

No.	Rule	Expansion	Distance
0	hull	(rect right small)	
1	elaborate hull	((in left) (elli (up left) (very (very small))) (rect right small))	.415
2	elaborate hull	((tin down) (rect up small) (rect right small))	.125
3	door	(rect up medium)	
4	elaborate door	((int left) (rect down medium) (rect up medium))	.250
5	crown	(tria up small)	
6	elaborate crown	((on up) (tria up small) (tria up small))	.125

5.4.5 Conclusion

The extensibility of language in terms of its compositional and recursive nature is utilised to combine sub-rules and elaborate these rules to generate abundant compositions that illustrate the original meanings in more detail. The rules generated using Context Free Grammar (CFG) are easily extended and spread in graph networks. CFG is also used to produce incongruent and exaggerated works as well as some other interesting scenes by replacing operators or operands. Conditional functions can also be embedded into graph networks to select different tracing paths according to different situations; and this could be extended to Context Sensitive Grammar (CSG).

5.5 Conclusion

The creative features of language—ambiguity, scalability, incongruity and extensibility—are tested and discussed in this chapter. Ambiguity makes it possible to connect seemingly unrelated concepts via multiple mapping of utterances and meanings. As can be seen from the experiment, Ambiguity (see Section 5.1), with the initial input of only 8.3% of samples, the client-agent still collects the vast majority of topics at the end of the simulation by benefiting from ambiguous connections. Scalability is utilised to generate exaggerated representations with the aid of modifiers in combination with existing sizes such as “small” → “very small” and the technique of range extension (see Section 5.2). Incongruity is applied to produce interesting new scenes by combining shapes with certain contexts with low possibilities based on the collected expressions representing topological relations using weighted context free grammar (see Section 5.3). Extensibility is tested and realised by recursive combination of CFG rules to produce elaborated objects based on the original scene samples. Some other interesting topics such as unusualness and strangeness are also explored in the last experiment (see Section 5.4).

Chapter 6

Exploring Social Creativity

Social creativity could be strengthened by increasing the scale of population and using different types of communication. Some parameters are tested to find appropriate strategies of increasing population and selecting efficient communication methods. The evolution of languages at a multi-culture level may result into diversity and emergence. Each domain evolves its own knowledge and culture carried by a domain language. The information from different cultures can be exchanged by the communication of individuals from different domains to improve the diversity of new artworks and designs.

6.1 Growing Population

To enhance the efficiency of the evolution of artificial languages in a large population, different strategies are tested, and two questions are proposed to be answered: Is a step-by-step evolution more efficient than a "direct" evolution? What are the roles of mature agents and naive agents in a growing population?

6.1.1 Subjects

The subjects of this experiment are RGBA colours with four dimensions—red, green, blue and alpha.

6.1.2 Implementation

Guessing games are used in the simulations, which are the same as those in the experiment, Ambiguity (see Fig. 5.2). Two different strategies — a fixed population with initial large scale and a growing population with initial small population that grows to

the same large scale as the former one — are adopted. Growing population is implemented by adding the same number of naive agents with mature agents at each stage (see Fig. 6.1). For example, 4 naive + 4 mature → 8 naive + 8 mature → 16 naive + 16 naive ...

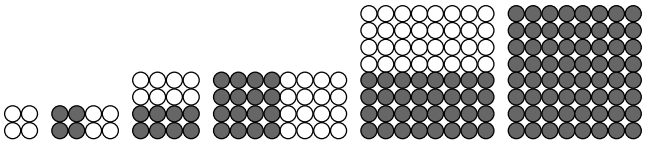


FIGURE 6.1: Growing population (gray: mature, white: naive)

The experiment is implemented 36 times[1] for each number of populations (4, 8, 16, 32 and 64)[2] based on the DIFI framework (see Fig. 6.2). The growing population starts from 4 naive agents. Then the number of agents are doubled (see Fig. 6.1) each time when the success rate of guessing games surpasses 70%[3].

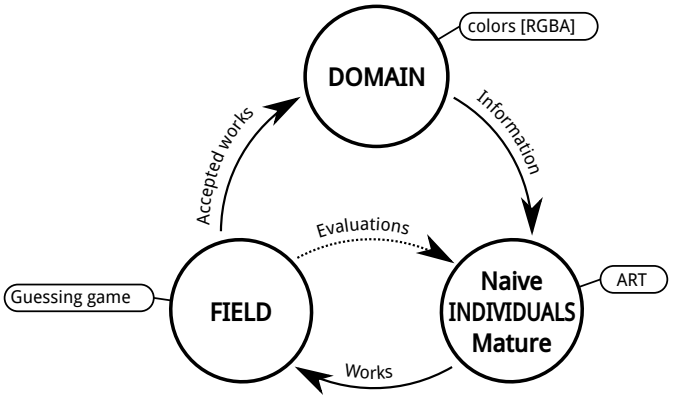


FIGURE 6.2: The Domain-Individual-Field-Interaction (DIFI) framework of growing population

6.1.3 Results

Results show that a growing population is more efficient than the fixed population on a large scale in the evolution of artificial languages. The language games played by a mixed population of mature and naive agents that grows larger in each generation

[1] 36 times of implementation is to make sure that the calculated average gaming times are credible.

[2] The population starting from 4 is because each guessing game needs 2 agents, and the remaining 2 agents can be candidates for next games. Double population is for the convenience of comparison of growing population and fixed population. Some other growth ratios such as triple population can also be used, but double population is simple enough to implement the experiment. When doubling to 64, it costs about 19000 times to complete the language games based on fixed population. If the population was continually doubled to 128 and 356 etc, it could cost too much time but contribute little to clarifying the experiment results. Five populations stages from 4 to 64 is enough to compare the two strategies, growing population and fixed population.

[3] There is no need to set the threshold of success rate greater than 70%. A common language has been shared by agents when the success rate reaches 70%.

TABLE 6.1: The average gaming times of evolving language with different population strategies

Fixed population (FP)	Growing population (GP)	Gaming times of FP	Gaming times of GP
4 (naive)	4 (naive)	137	133
8 (naive)	4 (mature) + 4 (naive)	499	395
16 (naive)	8 (mature) + 8 (naive)	1596	1043
32 (naive)	16 (mature) + 16 (naive)	5739	2563
64 (naive)	32 (mature) + 32 (naive)	19500	5742

succeed more quickly than those played by a fixed population of naive agents on the same scale (see Table 6.1 and Fig. 6.3).

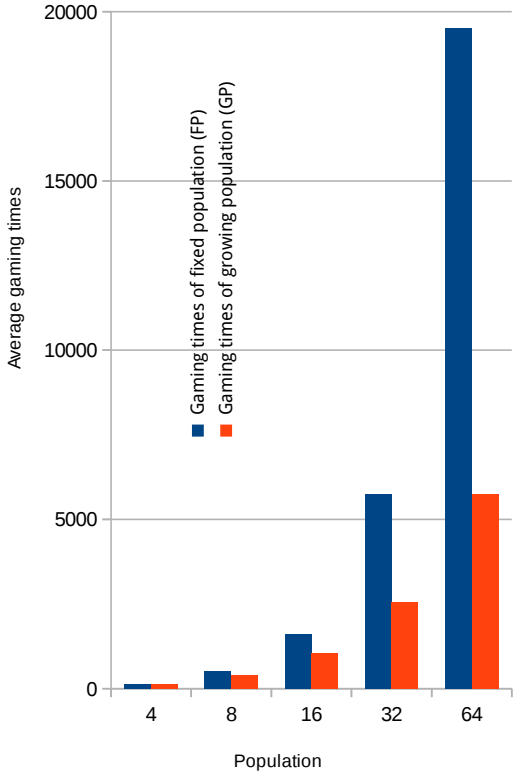


FIGURE 6.3: The average gaming times of evolving language with fixed population and growing population

6.1.4 Conclusion

The strategy of mixing mature and naive agents in a growing population is more efficient than that of using only naive agents in a fixed population especially on large scale for evolving artificial languages. It is probably because, after the first generation, there are already half of the "growing" population (i.e., mature agents) sharing a common language before implementing language games in a new generation compared with the "fixed" population full of naive agents. And when more generations are simulated in the "growing" population, a more common language may evolve and be shared among

mature agents that results into more efficient evolution of languages in next generation. But whether the increase in efficiency of the "growing" population would continue or not after a great number of generations needs to be tested for future work.

6.2 Education in Guessing Game

To explore the efficiency and creativity of education particularly in the conversation between teachers and students, a number of ratios of playing initiator between teacher-agents (i.e., mature agents) and student-agents (i.e., naive agents) with different student–teacher ratios are tested.

6.2.1 Subjects

The subjects of this experiment are RGBA colours with four dimensions, i.e., red, green, blue and alpha.

6.2.2 Implementation

Teacher-agents (TAs) are mature agents developed in initial guessing games while student-agents (SAs) are naive agents. The guessing games used in these simulations are similar to those in the experiment, Ambiguity (see Fig. 5.2). A specific aspect of the implementation is that speakers (i.e., initiators) are selected from both teacher-agents and student-agents by weighted random choice. The weight is based on a predefined ratio (see Table 6.2). Different ratios of playing initiator between teacher-agents and student-agents with several student–teacher ratios are tested based on the DIFI framework (see Fig. 6.4).

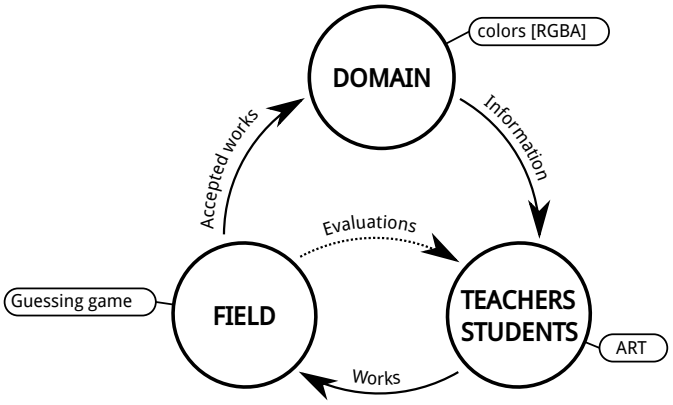


FIGURE 6.4: The Domain-Individual-Field-Interaction (DIFI) framework of education in guessing games

TABLE 6.2: Average number of games being played with different ratios of playing initiator between teacher-agents (TAs) and student-agents (SAs)

Ratio of initiators (TA:SA)	10:0	9:1	8:2	7:3	6:4	5:5	4:6	3:7	2:8	1:9
Population: 20 SAs + 1 TA	589	877	1355	1268	2934	2621	4737	6818	11002	12448
Population: 20 SAs + 2 TAs	1082	1308	1997	1808	2461	3464	6185	9158	10321	8878
Population: 20 SAs + 4 TAs	2035	2778	2647	3315	3218	5026	7290	11107	19934	23197
Population: 20 SAs + 10 TAs	2928	3207	3592	4419	4621	6447	10565	17383	24910	33563
Population: 20 SAs + 20 TAs	4545	4281	4868	5047	5919	9428	15239	23611	42763	65885

The experiment is implemented 20 times for each ratio of playing the role of initiator (i.e., speaker) between teacher-agents and student-agents from 10:0 to 1:9 with different student–teacher ratios including 20:1, 10:1, 5:1, 2:1 and 1:1 (see Table 6.2).

6.2.3 Results

The smallest average number of games played to success is 589, which is completed when the ratio of playing initiator between TA and SA is 10:0 and the student–teacher ratio is 20:1 with the smallest population compared with other populations (see Table 6.2 and Fig. 6.5, 6.7). This means that the guessing games in education could be efficient when the role of initiator is played by only one teacher-agent. It is because all the student-agents learn from the same teacher-agent in high consistency without other teacher-agents' "disturbance", and the student-agents do not play the initiators to generate new utterances which could "disturb" the teacher-agent with its own language. However, there is few creative behaviours occur in such one-direction teaching by only one teacher-agent that can be seen from Figure 6.6. A teacher-agent's original language is the same as its language after education games when the initiator is played only by the teacher-agent, i.e., the ratio of initiator (TA:SA) is 10:0 (see Fig. 6.6).

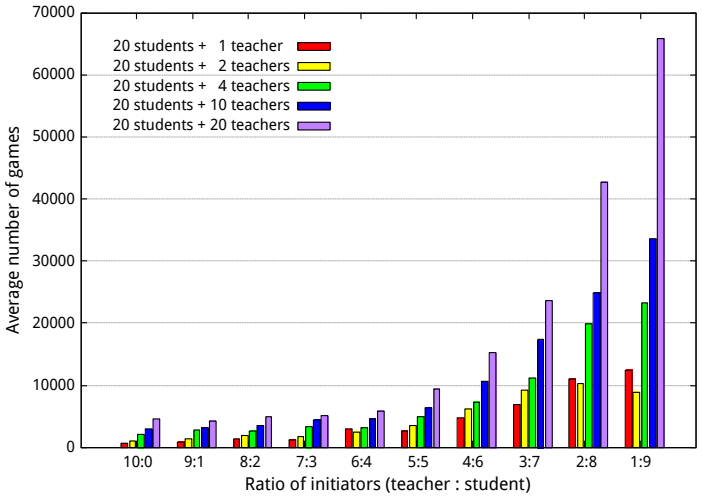


FIGURE 6.5: Average number of games being played in education model

A teacher-agent's language becomes evolve when the ratio of playing initiator between a teacher-agent and a student-agent is changed from 10:0 to 9:1. In addition, when the ratio becomes smaller, the evolved language in education games is more different from the original language, even totally different when the ratio reaches 2:8 and 1:9 with some high student–teacher ratios (see Fig. 6.6). It means that more new utterances could be generated when a student-agent plays the role of initiator more. Such new utterances are learned by a teacher-agent. It also costs more time on assimilating and sharing the new utterances (see Fig. 6.5).

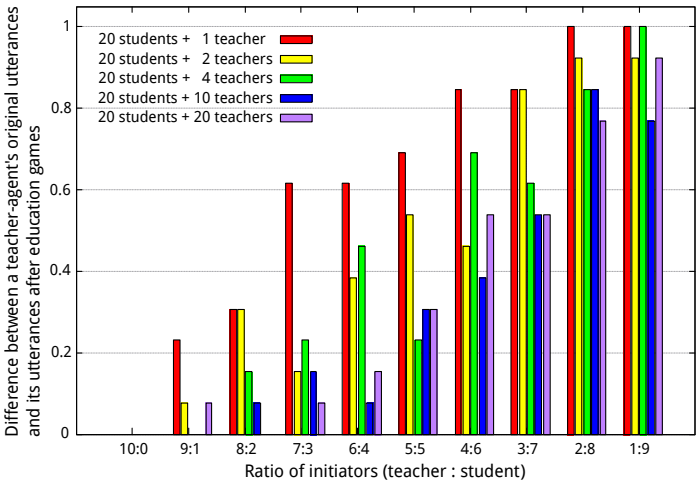


FIGURE 6.6: Difference between a teacher-agent's original utterances and its utterances after education games

As can be seen from Figure 6.7, the number of games that need to be played becomes greater and more unstable when the ratio of playing the initiator between a teacher-agent and a student-agent becomes smaller, i.e., when a student-agent has more opportunities of playing the role of speaker. Such change becomes faster when the ratio is less than 6:4 especially with lower student–teacher ratio such as 2:1 (e.g. 20 students and 10 teachers) and 1:1 (e.g. 20 students and 20 teachers).

6.2.4 Conclusion

Double-direction education is more creative than single-direction teaching due to student-agents' novel behaviours, i.e., the potential of associating existing context with new meanings. In terms of the efficiency of education, a teacher should play the main role of initiator with appropriate ratios such as 7:3 or 6:4 while a student takes part in the conversations positively rather than passively only as recipient. A suitable ratio can not only drive creative behaviours but also prevent from extremely generating too many new utterances which could cost too much time on assimilation.

The student–teacher ratio also affects creative behaviours. New languages could evolve easily with higher student–teacher ratio (see Fig. 6.6) because a comparable great

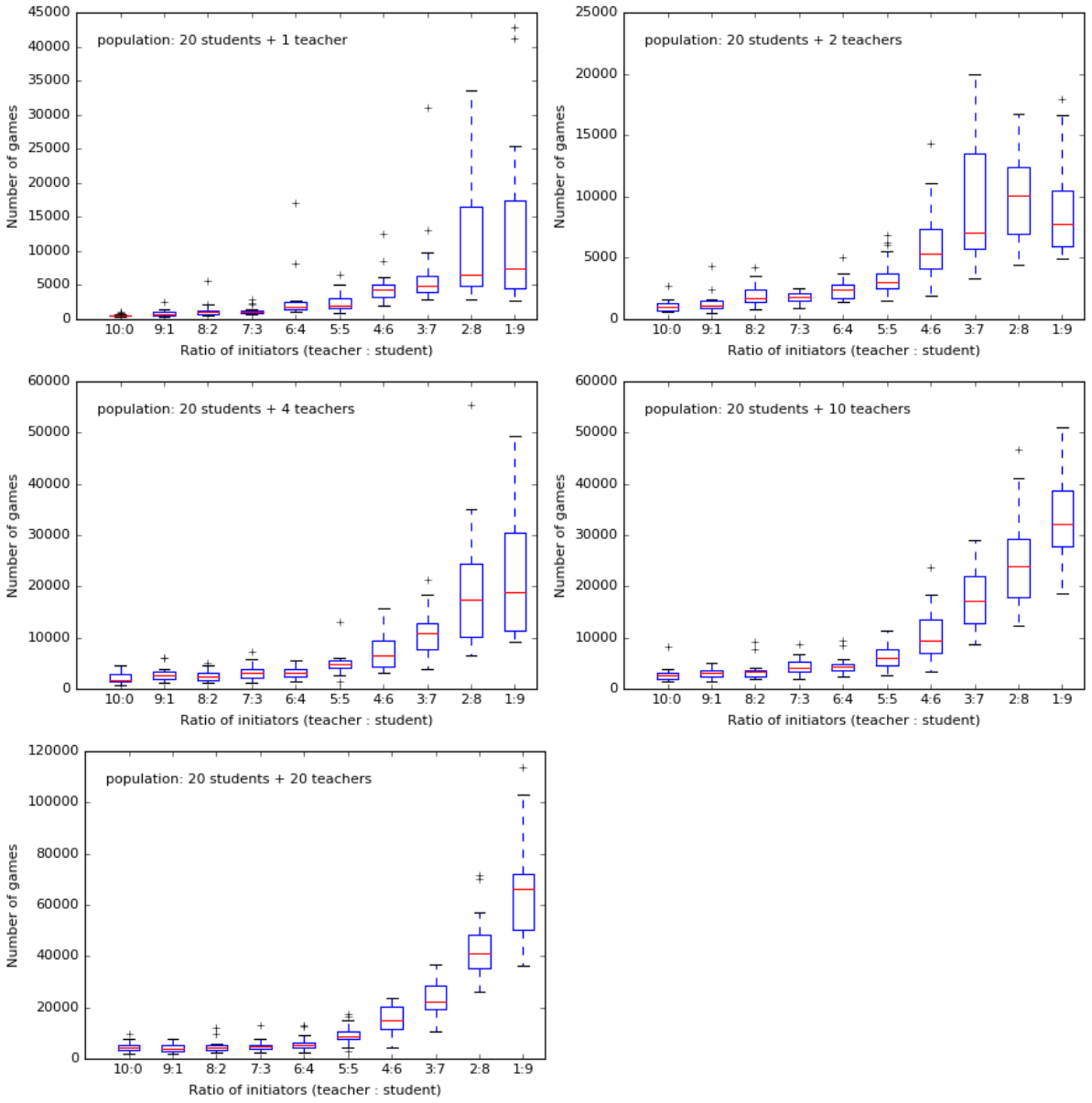


FIGURE 6.7: The number of games being played in education model with different student–teacher ratios

number of students may have more influences on generating novel utterances associating new meanings rather than using the existing utterances repeatedly by teacher-agents if the teacher-agents share a same language with little diversity in advance.

6.3 Clique Formation

To improve design diversity and creativity at the sociocultural level, the evolution of multiple domain languages resulting into clique formation are simulated. This experiment aims to simulate the evolution of artificial languages between different cultures. Via

the simulation, the grounded language could be expanded to different local languages which are helpful for investigating specific features impacting on different designs or the convergence of multiple languages to one language with more complicated patterns resulting into new design styles.

Creativity may increase when an individual adapts to an unexpected situation during a "creative activity" (Candy and Bilda, 2009). Exchanging agents between different environments and cultures could increase creativity.

6.3.1 Subjects

The subjects of this experiment are parametric designs of vases and simple buildings with 4 to 5 parameters which can be adjusted to alter their shapes. Four sectional radius can be changed to alter the shape of a vase (see Fig. 6.14), and five variables including length, width, height, scale, and the angle of twist can be modified to adjust the shape of a simple building (see Fig. 6.16).

6.3.2 Implementation

Generally, two groups of agents communicate with each other after they have evolved their own languages (see Fig. 6.8). Initiated with random choice in the interactions of multi-agents, some associations would be strengthened while others might be weakened. Agents firstly communicate randomly with other agents. Then they may prefer to communicate with certain agents according to the familiarity based on their communication frequency or the success rate of their communications. In terms of different communication strategies, some agents may prefer to interact with similar and familiar agents while others prefer to interact with agents that are very different from themselves.

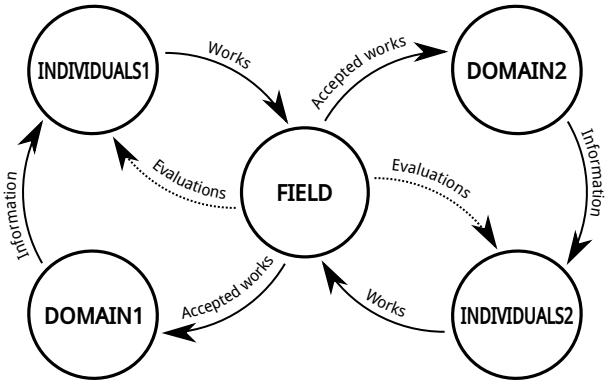


FIGURE 6.8: The Domain-Individual-Field-Interaction (DIFI) framework with multi-domains

The experiment is implemented using the combination of guessing games and generation games adjusted via the bias of agents' selection identified by their previous communication experiences. First, the basic languages evolve via guessing games between speaker-agents and listener-agents. Each time, a speaker is randomly selected from a group of agents, then the speaker randomly selects a listener, or chooses a listener who might be the best listener in their previous communications when the bias of communication is set to be true.

When the success rate of the guessing games reaches 70%, generation games start running for 1000 times. Each time, a randomly selected client selects designers randomly, or prefers to choose the designers with whom it previously had more successful communication when the bias of communication is set to be true. The designers with different preferences (different styles) parse the client's requirement, generate various designs and submit them to the client, who will make the assessment. Two types of simulations are implemented. They are described below.

6.3.2.1 Implementation of Simulation 1

The processes of guessing games and generation games in Simulation 1 are similar to those of the experiment, Ambiguity (see Fig. 5.2 & 5.3). The network based on Adaptive Resonance Theory (ART) is used to learn 4-dimensional samples mapping to 4 sectional radius of a vase through language games among 10 agents.

6.3.2.2 Implementation of Simulation 2

The processes of guessing games (see Fig. 6.9) and generation games (see Fig. 6.11) in Simulation 2 are different from those in Simulation 1. There is no ART machine learning; rather, the combination of value-range and tolerance is used to categorise 5-dimensional samples mapping to simple buildings.

A number of dictionaries are used to store the weighted relations (i.e., rules) between categories (i.e., value-range) and utterances. Each dictionary contains value-range, utterance and weight, e.g. {value-range = {3: 0.9, 4: 1, 5: 0.9}, utterance = "s", weight = 0.01}.

The strengthening and weakening equations are used twice in learning (see Equations 45, 46) and unlearning (see Equations 47, 48) respectively. In the equations, w_r is the weight of a rule while w_v is the weight of a value in the value-range of the rule; for example, $w_r = 0.01$, 3: $0.9 \rightarrow value = 3$, $w_v = 0.9$ in the rule: {value-range={3: 0.9, 4: 1, 5: 0.9}, utterance="s", weight=0.01}.

$$w_r \leftarrow \eta + (1 - \eta) \times w_r \tag{45}$$

$$w_v \leftarrow \eta + (1 - \eta) \times w_v \tag{46}$$

$$w_r \leftarrow (1 - \eta) \times w_r \tag{47}$$

$$w_v \leftarrow (1 - \eta) \times w_v \tag{48}$$

In guessing games (see Fig. 6.9), 16 agents are divided into two groups (see Fig. 6.10) to play their own language games respectively, and evolve their own domain languages.

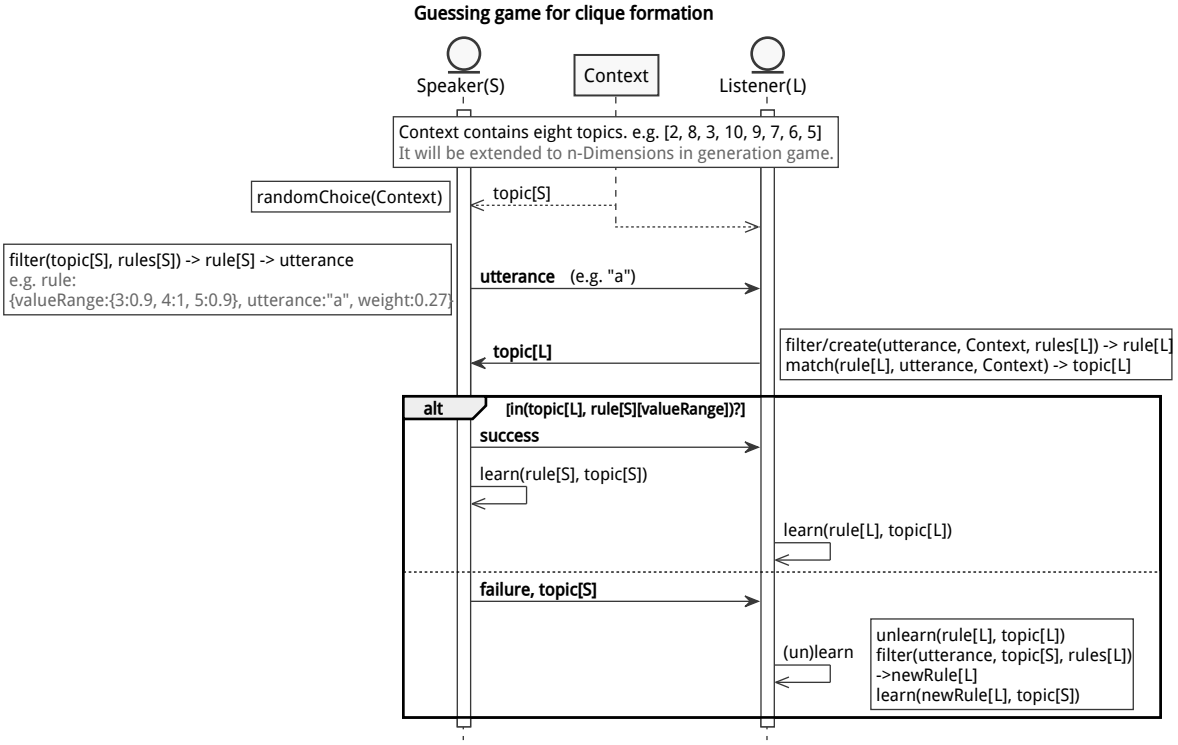


FIGURE 6.9: Guessing game of clique formation

In generation games (see Fig. 6.11), a client-agent is randomly selected from all the 16 agents. Then one or two more designer-agents are selected by the client-agent based on its previous communication success rate with the designer-agents. The designer-agents whose works are not accepted may or may not learn from the winner-designer according to different strategies.

6.3.3 Results

The results of Simulation 1 (see Table 6.3) show that 10 agents evolve into several cliques through generation games according to the differences between their evolved languages (see Fig. 6.12). The differences of languages may impact on the diversity of designs (see Fig. 6.13 and Fig. 6.14). The differences between agents' utterances and designs are measured using cosine similarity and visualised with the distance map of hierarchical

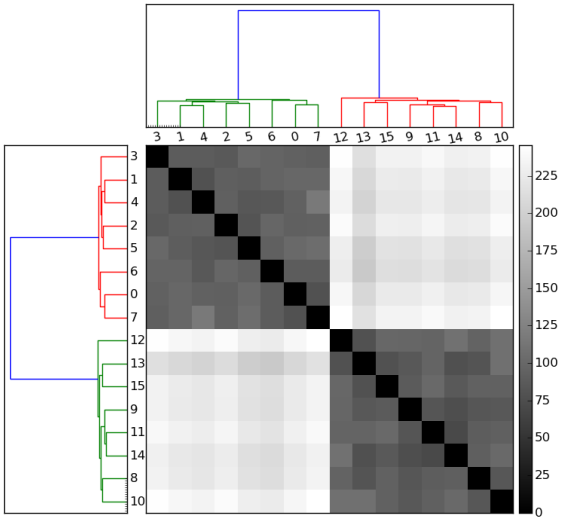


FIGURE 6.10: Two groups evolve languages in guessing games

clustering. As can be seen from Fig. 6.12, Fig. 6.13 and Table 6.3, the result of hierarchical clustering of the evolved languages is very similar to that of the collected designs. It means that communication bias may not only strengthen the differences between languages and that between designs but also reinforce the domain cultures by improving the relationships between their languages and related designs, i.e., languages and designs reflect each other. However, the reflection is not strict due to a few of mismatches between certain languages and designs. For example, agent-2 shares the same language with clique-1 but has the same design taste as that of clique-3 leading to the overlap between clique-1 and clique-3, and agent-6 shares the same design preference with clique-1 but shares the same language with clique-2 that results in the intersection of clique-1 and clique-2 (see Table 6.3).

TABLE 6.3: Clique formation based on the differences of languages and designs

Clique index	1	2	3
Members of each clique (based on their languages)	[0,2,8,9]	[1,3,6,7]	[4,5]
Members of each clique (based on their stored designs)	[0,6,8,9]	[1,3,7]	[2,4,5]

The results of Simulation 2 (see Fig. 6.15) show that agents who are more tolerant are more capable of accepting languages that differ from their own. This leads to mixing two different languages to one new language with a number of sub-languages at different levels related with different cliques (see Setting 1 and 2 of Fig. 6.15). Two languages almost merge to one language when the success rate of generation games, which are played by tolerant agents, reaches 90% (see Setting 2 of Fig. 6.15) compared with 80% (see Setting 1 of Fig. 6.15).

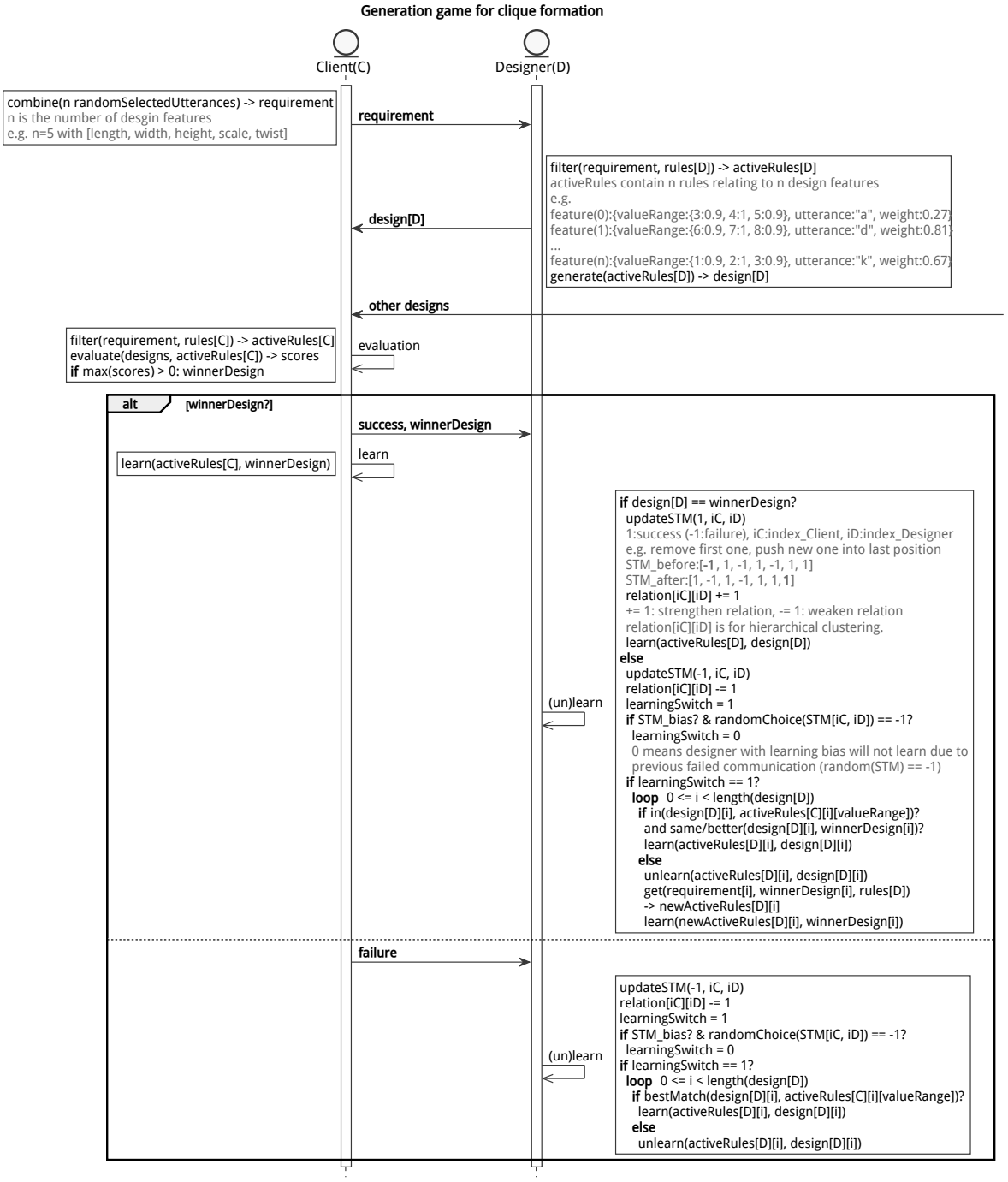


FIGURE 6.11: Generation game of clique formation

By contrast, when agents' tolerances are very low (such as 0.01 compared with 0.1 in this experiment), it is difficult to merge two languages into one (see Setting 3 and 4 of Fig. 6.15). A number of simple buildings related with different utterances are generated in Simulation 2 (see Fig. 6.16).

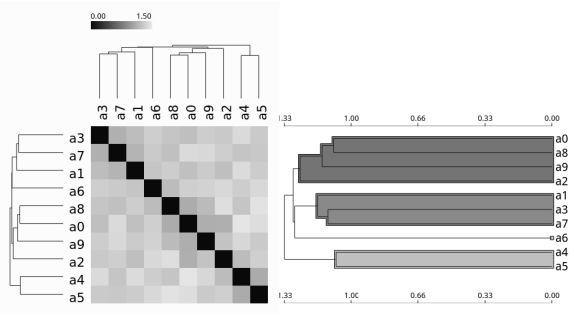


FIGURE 6.12: The hierarchical clustering of agents based on their utterances

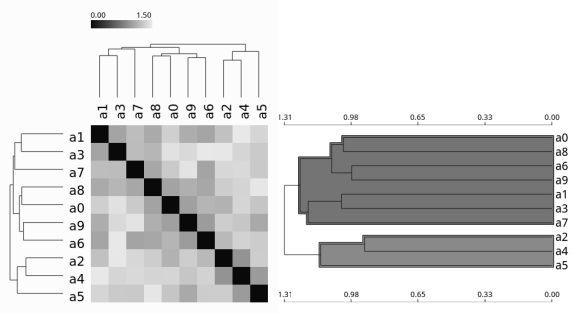


FIGURE 6.13: The hierarchical clustering of agents based on their stored designs

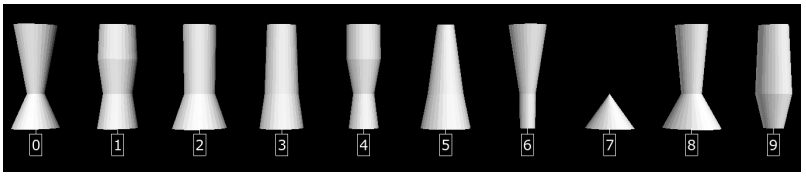


FIGURE 6.14: The designs generated by averaging each agent's stored prototypes

6.3.4 Discussion

Based on the results of Simulation 1, it may be very helpful for building interesting relationships between cliques by the aid of the mismatches between domain languages and designs. Some interesting concepts might be developed by the agents who share their languages and designs with different cliques respectively. In terms of the ambiguity of languages, two individuals, who share the same utterances but associate them with different meanings, may generate some interesting novel ideas when they connect these meanings during their conversations. Hence, the creative relationships between differences (i.e., different cliques) could be strengthened when the differences themselves (i.e., the differences between cliques) are increased via the interactions between the languages and designs belonging to various domains.

According to the results of Simulation 2, clique formation can be affected by three factors: tolerance, learning bias and the success threshold of language games. If an agent's tolerance is great, i.e., the threshold for new categorisation is high, an utterance can represent a broad value range. For example, if tolerance=0.1, each category contains

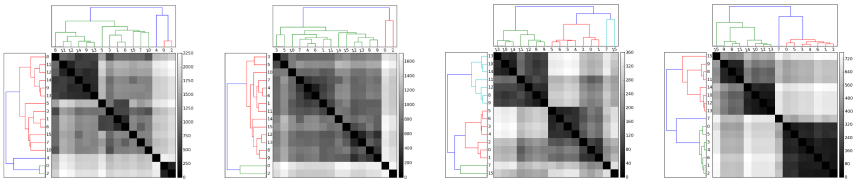
Setting	1	2	3	4
Tolerance (t)	0.1	0.1	0.01	0.01
Threshold (gs)	0.9	0.9	0.8	0.9
Threshold (gt)	0.8	0.9	0.8	0.8
LearningBias (b)	1	1	1	1
Result				

FIGURE 6.15: The cliques evolved with different tolerance and threshold

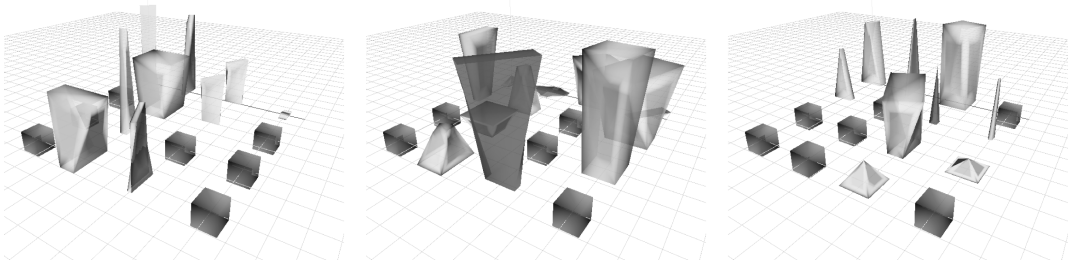


FIGURE 6.16: Some results of generated buildings

three values, the value range could be from 0.2 to 0.4 including 0.2, 0.3 and 0.4. If tolerance=0.01, only one value is included in a category such as "b" only representing 0.3. The function of getting value range is shown in Algorithm 16. It is difficult to mix two groups when their tolerances are very low such as 0.01 (see Setting 3 and 4 of Fig. 6.15). Otherwise the two groups will mix with each other to form more than two sub-groups (cliques) or even integrate into one group when their tolerances are equal to or greater than 0.1 (see Setting 1 and 2 of Fig. 6.15).

Algorithm 16 Get Value Range

```
1: function getValueRange(value, tolerance, range)
2:     valueRange ← {}
3:     base ← range_max − range_min
4:     for v in range do
5:         pos ← (v − range_min)/base
6:         peakPos ← (value − range_min)/base
7:         score ← sineCurve(pos, peakPos)
8:         if score >= 1 − tolerance then
9:             valueRange_v ← score ∗ 0.1
10:        end if
11:    end for
12:    return valueRange
13: end function
```

Learning bias also affects clique formation. A loser-designer may or may not learn from

the winner-designer or the client depending on their previous communication records. If the success rate of their previous communication is high, the loser would learn from the winner; otherwise the loser may refuse to learn from the winner. Therefore, this factor could affect the evolution of shared languages. It may impede change in the number of cliques or evolving sub-cliques, or prefer to retain the current status. The status will change, however, when the members of two different groups communicate for a very long time and increase the success rate of communications from 0.8 to 0.9. More commonly, two groups mix into several sub-groups when the success rate reaches about 0.7 to 0.8.

Last but not least, the success threshold of language games may influence the result of clique formation. Two groups with different languages would merge or mix more easily when their own languages are less mature resulting from lower success threshold of guessing games (see Setting 3 compared with Setting 4 of Fig. 6.15). In addition, the individuals from two different groups would assimilate each other after playing a great number of language games even their own languages are very mature (see Setting 2 compared with Setting 1 of Fig. 6.15)

6.3.5 Conclusion

Cliques are developed by the communication between a number of groups of agents with different cultures, i.e., domain languages. Clique formation results in diverse and creative designs which are more similar in one clique than in others but all differ from each other due to different combinations of utterance-like requirements and various experiences of designers. A few of mismatches between domain languages and designs might increase the social creativity of the interactions between different cliques. Technically, not only individuals' tolerances and communication preferences based on their previous experiences can influence the results of clique formation, but also different success thresholds of language games and the number of language games played by agents could affect the results of the interaction of multiple languages.

6.4 Conclusion

Three social phenomena related to creativity are simulated and discussed in this chapter. First, the process in which the population mixed with naive and mature individuals grows in an orderly way from generation to generation is more efficient than one-generation process for the evolution of artificial languages on a large scale. Secondly, the simulation of educational conversation in guessing games shows that two-way interaction between teachers and students may be more creative than single direction teaching, particularly when the ratio of playing initiator between teachers and students is 7:3 or 6:4. Finally, clique formation occurs during the interaction between multiple languages leading to social diverse creativity.

Chapter 7

Discussion

The main objective of this thesis is to enable and augment the exploration of creative features of language and develop an artificial language system for creative designing at the sociocultural level. This objective has been realised by developing three aspects of multi-agent simulations (see Chapter 4, 5 and 6) based on the computational DIFI framework (see Section 3.1) and the procedure of representation, communication and evaluation (see Section 3.2, 3.3, 3.4 and Fig. 4.20 as an example).

The first aspect, evolving compositional languages (see Chapter 4), shows that compositional languages are more efficient (see Fig. 4.4) and creative (see Fig. 4.6 [1]) than holistic languages; Holographic Reduced Representations can be used to represent geometric relations (see Table 4.10), which can be mapped to compositional utterances via Self Organising Map (see Fig. 4.14); and interesting designs can be generated in the evolution of compositional languages (see Fig. 4.21). The second aspect, exploring language features (see Chapter 5), indicates that design brief may be more influential than design works partially due to the ambiguity of language (see Fig. 5.8); exaggeration can be used to expand a design space, differentiate and overlap design categories (see Fig. 5.18); incongruent combinations can be obtained by inverse weighted random choice of possible combinations (see Section 5.3.4); and elaboration can be used to generate complicated concepts via the recursion of context free grammar (see Table 5.19). The last aspect, exploring social creativity (see Chapter 6), suggests that growing population mixed with mature and naive individuals can be more efficient than fixed population in the evolution of language (see Fig. 6.3); appropriate two-way education is better than unidirectional learning (see Fig. 6.5, 6.6); and the evolution of cliques generating specific designs is affected by individuals' tolerances, communication preferences and the success thresholds of language games (see Fig. 6.15). Based on these results of the three aspects of simulations, the main contributions of this thesis are summarised as follows:

[1] Less discrimination means more ambiguity, which is a creative feature of language.

- A computational process, representation-communication-evaluation (see Section 3.2, 3.3, 3.4 and Fig. 4.20 as an example), has been developed to evolve grounded language for creative design via guessing games and generation games;
- Several language features including ambiguity, scalability, incongruity and extensibility (see Section 5.1, 5.2, 5.3 and 5.4) have been implemented and tested in computational simulations to support reasoning about these features in the context of creative design;
- A computational model of social creative system (see Section 3.1) based on the DIFI framework has been developed to incorporate the evolution of artificial languages for improving sociocultural creativity.

7.1 An Artificial Language System for Creative Design

An artificial language system based on the Domain-Individual-Field-Interaction (DIFI) framework has been developed to evolve compositional languages by combining and transforming diverse associations of utterances and design concepts among multi-agents using language games. In other words, a model of an artificial creative system has been developed via the evolution of compositional language for creative design at a sociocultural level, and the features of language—ambiguity, scalability, incongruity and extensibility—have been investigated and applied to design conceptualisation and communication through guessing games and generation games. Various views and different experiences of curious agents reflect the social diversity that resulted in cooperative design creativity.

7.2 Significance of Experiments and Results

The significance of experiments and results are the evolution of compositional languages, the exploration of creative features of language, and the strategies of evolving languages at sociocultural level for creative design.

7.2.1 Evolution of Compositional Languages for Creative Design

Chapter 4 describes three experiments: the comparison of compositional languages and holistic languages, Holographic Reduced Representations of topological relations of two rectangles and mapping them to compositional utterances, and the evolution of compositional languages for generating interesting combinations of regular shapes. Results show that:

- The evolution of compositional languages is more efficient than that of holistic languages (see Table 4.4). The feature, ambiguity, occurs in both but it is stronger in the former due to its lower discrimination (see Table 4.5).
- Topological relations can be represented by Holographic Reduced Representations (HRRs), and the distances between these relations can be clarified by measuring cosine similarities of their HRRs;
- Mapping between HRR geometric relations and HRR represented utterances can be accomplished using SOM, which establishes the feasibility of transformation between artificial languages and design concepts (see Section 4.2).
- The evolution of compositional languages can be used to generate designs, especially when a client-agent selects similar-yet-different designs rather than highly familiar or highly different designs (see Section 4.3).

7.2.2 Exploration of Language Features for Creative Design

Four experiments related to the features of language—ambiguity, scalability (exaggeration), incongruity and extensibility (elaboration)—have been implemented and applied in conceptual design (Chapter 5). The key results from these experiments are:

- The ambiguity of language caused by polysemy may play an important role in creative designing at the social level aided by hedonic selection. This leads to fewer utterances representing more concepts suggesting that in systems with ambiguous languages design briefs may be more influential than design works (see Section 5.1).
- Exaggerated concepts are generated by using modifiers such as "very" to expand the range of topics, differentiate existing categories such as "very medium" from "medium", and overlap basic categories to get more dynamic meanings, e.g. "small-medium" and "medium-small" (see Section 5.2). A design space can be expanded and more options could be provided to designers for exploring interesting combinations of different sizes to generate novel designs by extending and differentiating existing categories via exaggeration.
- Incongruent combinations can be computationally modelled by inverse weighted random choice of possible combinations in generation games based on the "real" congruent combinations accumulated in guessing games (see Section 5.3). It could be used for the computational modelling of conceptual design to explore the incongruent recombinations of existing products and usage situations.

- Elaborated combinations and other interesting (unusual, strange) relations can be generated via the recursion of context free grammar and the replacement of operators and operands (see Section 5.4). Elaborated combinations can be modelled computationally to provide human designers abundant raw concepts for choosing and refining.

In brief, four creative features of language have been used to evolve compositional languages for generating interesting designs mainly aided by weighted context free grammar with graph networks (see Section 3.2.3).

7.2.3 Exploration of Social Creativity in Designing

Three experiments—growing population, education in guessing games and clique formation—are described in Chapter 6. These experiments explore the evolution of compositional languages in dynamic populations, education model and multiple cultures. The aim of these experiments was to develop a computational model to study the role of compositional languages in the social aspects of creative designing. The key results from these experiments are:

- The results of the experiment, Growing Population (see Section 6.1), demonstrated that gradual staged evolution of languages in a mixed population of mature and naive individuals can be more efficient than the direct evolution of languages in an initially large population of naive individuals (see Section 6.1).
- Positive two-way education can be better than passive unidirectional learning, as demonstrated by simulating Education in Guessing Games with different ratios of playing initiator between teachers and students (see Section 6.2).
- The interactions among individuals from different cultural backgrounds, reflected in different languages, can evolve cliques generating specific design works (see Section 6.3). The number of cliques and the degree of difference between the cliques are affected by individuals' tolerances and communication preferences based on their previous experiences (see Section 6.3.4).

These experiments have expanded the range of social behaviours that can be computationally studied. Given the success of these experiments, the ability of language evolution as a way to expand the range of social and cultural phenomena surrounding design can be computationally modelled.

7.3 Comparison of the Experiments

This dissertation has explored three aspects of language and design. First, the processes supporting the evolution of compositional languages in design has been explored. Then, the features of language including ambiguity, scalability, incongruity and extensibility have been identified and applied to design. Finally, several models of evolving artificial languages for designing at the sociocultural level are explored. Most of the experiments have been based on a computational model based on the Domain-Individual-Field-Interaction (DIFI) framework. This section provides an overview of the outcomes of the experiments related with their general settings, representations and categorisations, evaluation and analysis, and advantages and disadvantages.

7.3.1 General Settings

As can be seen from Table 7.1, both guessing games for evolving domain languages and generation games for generating designs are used in most simulations, although the experiment, Compositional and Holistic Language (see Section 4.1), uses only guessing games to compare the efficiency of compositional languages and holistic languages in evolving languages. The experiment, Compositional Representation of Rectilinear Relation (see Section 4.2), does not use any language games because its goal is to test mapping HRR meanings to HRR utterances; this is achieved using SOM.

Most of these experiments are set with a small population with 6 individuals, except for the experiments that explore social creativity with 10 to 64 agents (see Table 7.1). These settings are suitable for exploring grounded features of language. Large scale with more than 1000 agents may be useful to study the impact of diversity of languages on creative design leading to emergence. A large-scale agent system is highly dynamic due to both the heterogeneity and interoperability of agents and the great number of (inter)actions (Wijngaards et al., 2002). To adapt to large scale simulations for diverse creativity, the personality and flexibility of agents may need to be strengthened.

The success threshold of most simulations with guessing games and generation games is set to 0.7 or higher. There are two exceptions. In the experiment, Ambiguity (see Section 5.1), it is set to 0.6 because it is difficult to reach a high success rate (0.7) possibly due to the settings of the ART network. The vigilance parameter may need to be adjusted to increase the robustness of categorisation (Carpenter and Grossberg, 2016). In the experiment, Incongruity (see Section 5.3), the long sequence of utterances representing a long length of expressions with 21 items makes it difficult to achieve a high success rate in guessing games, so the success threshold is set to 0.65. However, the final success rate of the generation games in the experiment, Incongruity (see Section 5.3), surpasses 0.9 after only about 1000 iterations; this might also be because the long sequences of representations make them easily distinguishable from each other. In other words, it

TABLE 7.1: General settings of the experiments
(abbr.: Compo.&Holistic: Compositional and Holistic Language, Compo-Represen.: Compositional Representation of Rectilinear Relation, Compo-Shape: Compositional Language for Shape Combination, Extensibility: Extensibility and Other Features, Growing Pop: Growing Population, Education Game: Education in Guessing Games, Clique Form.: Clique Formation, gs: guessing game, gt: generation game, T:8 S:16: 8 teachers and 16 students)

Experiment	GameType	Population	Rate(gs)	Rate(gt)	Num(gs)	Num(gt)
Compo.&Holistic	gs	6	0.7	N/A	N/A	N/A
Compo-Represen.	N/A	N/A	N/A	N/A	N/A	N/A
Compo-Shape	gs, gt	6	0.7	0.7	Max:1000	Max:1000
Ambiguity	gs, gt	6	0.6	N/A	N/A	Max:1000
Scalability	gs, gt	6	0.7	0.7	Min:1000	Min:1000
Incongruity	gs, gt	6	0.65	N/A	Min:1000	Max:300000
Extensibility	gs, gt	6	0.8	0.8	Min:1000	Min:1000
Growing Pop.	gs	4,8,16,32,64	0.7	N/A	Min:20	N/A
Education Game	gs	T:8 S:16	0.7	N/A	Min:20	N/A
Clique Form.	gs, gt	10, 16	0.8,0.9	0.8,0.9	Min:20	Min:20

is easy for a client's long brief to be matched to the relevant topological relations only if the designer-agent decoded part of the requirement correctly; then the designer can produce unusual relations by inverse weighted choice. The max number of generation games (see Num(gt) in Table 7.1) in this experiment is set very high (to 300000) only for convenience in testing the trend of the success rate. The setting of min number for language games (see Num(gs) in Table 7.1) is to prevent premature high success rate.

Regular shapes (e.g. triangle, square and circle), colours, sizes and simple 3D objects with 4 to 5 parameters are used as subjects and design materials in the experiments (see Table 7.2). The values of most extracted features of these subjects are regulated into normal range [0,1] (see Table 7.2 and 7.3) for convenience in calculation and comparison.

The size of context, i.e., the number of samples, in each guessing game is set to 8 in most experiments, except for the experiment, Incongruity (see Section 5.3), in which only four samples (circle, square, triangle and the height of ground) are involved. Each shape has three features (coord-x, coord-y and radius), 10 features in total; and the values of these features are changed in a range in each guessing game. So, the context size of this experiment is sufficient for evolving grounded languages. There is no context size in the experiment, Extensibility and Other Features (see Section 5.4), because the features of the sample scene (single shapes, compositional objects and the relations between these objects and backgrounds) are extracted and generated before running guessing games. The guessing games in this experiment are only used to evolve shared languages among agents. On the sample scene, core features and relations are generated before running guessing games to avoid generating too many possible topics related with a speaker's utterances. For future work in this area, it may be better to allow agents to explore features step-by-step during guessing games.

TABLE 7.2: The subjects (samples) and selection methods of the experiments

Experiment	Subjects	Context	Topic Selection
Compo.&Holistic	3sizes*6colours*5shapes=90	8	rand.,(un)common,(un)confident
Compo-Represen.	Rectilinear Areas	N/A	N/A
Compo-Shape	Tri, squ, hex, circ&star	8	rand.
Ambiguity	11colours*11shapes=121(50)	8	rand.
Scalability	Size [0.25, 0.75] or (0,1]	8	rand.
Incongruity	Cir, squ, tri, H_{ground}	4	rand.
Extensibility	Shapes,sky,land,sea	pre-extracted	rand.
Growing Pop.	RGBA colours	8	rand.
Education Game	RGBA colours	8	rand.
Clique Form.	4 to 5 dimensional data	8	rand.

Most of the experiments adopt the random topic-selection method because of its convenience and objectivity. Random topic-selection is also more helpful for testing language games than other methods, such as uncommon, confident and unconfident topic selections, although the common topic selection is shown to be effective in the experiment, Compositional and Holistic Language (see Section 4.1).

7.3.2 Representations and Categorisations

The methods of representation in most experiments are general associations between meanings and compositional utterances (see Table 7.3). For example, "(in bird sky)" is represented as "vi-gu-wa". In addition, more than one artificial language can be represented using a graph network by connecting the same meaning with a number of utterances belonging to various languages (i.e., meanings – language 1 – language 2 – language 3 . . .). This may be very useful for obtaining interesting sequences by crossing over the utterances of different languages, or to get new connections between unfamiliar meanings by tracing the path of these mapped languages or comparing their phonemes and forms.

The experiment, Compositional and Holistic Language (see Section 4.1), is implemented to explore the different effects of compositional languages and holistic languages. The results show that compositional languages are far more efficient and a little more ambiguous than holistic languages. Ambiguity is a creative feature of language. So, compositional languages may be better than holistic languages for clients to generate creative requirements and for designers to describe their designs in a novel way. A compositional language is a mapping of a relational signal space to a structural meaning space (Smith et al., 2003) that could be useful to explore new relations between different meanings connected by associated utterances. Weighted context free grammar (WCFG) based on graph networks (GN) is used to evolve compositional languages to explore the creative features: scalability, incongruity and extensibility because of its support for expansion in multiple ways.

TABLE 7.3: The representations of the experiments

Experiment	Representation
Compo.&Holistic	[size, colour, shape] → holistic/compo-utters
Compo-Represen.	HRR (1024,float0 ∼ 1)
Compo-Shape	[shape1, shape2] → compo-utters e.g. triangle-square → "yawa"
Ambiguity	[colour, shape](range[0,1]) → compo-utters. e.g. {0.2,0.3} → "ha"
Scalability	WCFG&GN bridging sizes and compo-utters
Incongruity	WCFG&GN bridging [object, background] and compo-utters
Extensibility	WCFG&GN bridging compo-object/background and compo-utters
Growing Pop.	[R,G,B,A] → compo-utters e.g. "balagugi"
Education Game	[R,G,B,A] → compo-utters e.g. "balagugi"
Clique Form.	Building[length,width,height,scale,twist]&vase(4 radius) → compo-utters

Table 7.4 summarises the general categorisation method using prototype and tolerance applied in the experiments, Compositional Language for Shape Combination (see Section 4.3) and Scalability (see Section 5.2), aided by the variation of the Wundt curve (sine curve) and GNG respectively. The neural network based on ART is used to categorise topics in the experiments on exploring social creativity and the experiment, Ambiguity (see Section 5.1), due to the extensibility of ART to generate more categorisations when new topics surpass the tolerance of existing categories. SOM is used in the experiment, Compositional Representation of Rectilinear Relation (see Section 4.2), to map different types of rectangular areas to compositional utterances. The categorisations in the experiment, Incongruity (see Section 5.3), are related to topological relations between shapes and backgrounds generated by the evolution of expressions using WCFG, while in the experiment, Extensibility and Other Features (see Section 5.4), hierarchical categorisations are generated with basic shapes and compositional objects as well as the relations between these objects and different contexts. Then new designs are compared with the original topics with changed locations, the complexity of representations and scaled sizes in this experiment.

Interesting design works are produced by the cooperation of clients and designers. Client-agents provide design briefs and filter the products while designer-agents generate products, learn the responses of client-agents and affect client-agents' future requirements. The types of design works consist of the combinations of colours and shapes, the combinations of shapes, the combinations of objects and contexts, deformed shapes, exaggerated shapes, elaborated shapes, and simple 3D buildings and vases (see Table 7.4).

7.3.3 Evaluation and Analysis

The evaluation of designs is completed by a client-agent in each generation game. Fixing the role of some agents as client means that certain agent(s) always play(s) the role of client while other agents only play the role of designer in every generation game (see Table 7.5). This is done for convenience in studying the impact of client behaviours

TABLE 7.4: The categorization methods and designs of the experiments

Experiment	Categorization	Design/art works
Compo.&Holistic	Predefined sizes(3),colours(6),shapes(5)	N/A
Compo-Represen.	SOM	N/A
Compo-Shape	Prototype and preferred similarity	Deformation(Bezier), Boolean
Ambiguity	ART	Combining colour and shape
Scalability	GNG,prototype&tolerance(probability density)	Exaggerated sizes
Incongruity	Evolve expressions → topological relations	New relations(shape,context)
Extensibility	Compare prototype&design[loca.,complex,size]	Elaborated&interesting works
Growing Pop.	ART	N/A
Education Game	ART	N/A
Clique Form.	ART and Tolerance (0.1, 0.01)	3D buildings & vases

on generating works in a consistent way, especially in the experiments, Compositional Language for Shape Combination (see Section 4.3), Ambiguity (see Section 5.1), and Extensibility and Other Features (see Section 5.4).

In guessing games, the criteria of evaluation are whether the topic guessed by the listener is the same as the topic selected by the speaker, or whether the guessed topic and the selected topic belong to the same category. In generation games, the hedonic function based on Wundt curve is utilised in the experiments, Ambiguity (see Section 5.1) and Clique Formation (see Section 6.3), while sine curve is used in the experiment, Compositional Language for Shape Combination (see Section 4.3), to select similar-yet-different artworks. The other experiments use specific evaluation methods. For example, client-agents prefer to select new relations of shapes and context with lower weight to get unusual and incongruent scene works in the experiment, Incongruity (see Section 5.3).

Most experiments focus on evaluating the novelty of artworks (see Table 7.5) rather than the novelty of utterances, except the experiment, Ambiguity (see Section 5.1). The characters and forms of utterances might be analysed with more different views to capture interesting sequences and sentences in terms of their own forms rather than their represented meanings. A similar work has been done by Vogt (2005) in his language games using chunking and similarity of utterances as well as related sound etc., but not for generating creative utterances and designs.

The evaluation of language games in this thesis mainly uses non-linear hedonic functions with fixed peak-position. The functions with changeable peak-position, linear hedonic functions and heuristic functions or some other methods may be used to develop more diverse and flexible mechanisms of evaluation. For example, the peak of the Wundt curve can be positioned anywhere along the novelty axis by changing the thresholds of the reward and punishment (Saunders and Gero, 2001a). As can be seen from Algorithm 4, the peak-point can be adjusted by altering the value of $novelR$ and $novelP$.

TABLE 7.5: The comparison of client-agents and evaluation in the experiments

Experiment	Fixing client	Preferred-difference	Evaluation
Compo.&Holistic	N/A	N/A	Guessed the same topic?
Compo-Represen.	N/A	N/A	Cosine similarity(topological relations)
Compo-Shape	1	Preferred similarity = 0.7, 1	Selection threshold=0.7,0.9(sine curve)
Ambiguity	1	0.364 (Wundt curve)	With/without hedonic function
Scalability	0	$d_{VS,VM,MM,VL} = 4, d_{SS,LL} = 7$	Euclidean distance of prototype&design
Incongruity	0	Relations with low weight	$P(Shape \mid Context)$
Extensibility	1	Predefined criteria	Unusual/strange/exten./exagger./incon.
Growing Pop.	N/A	N/A	Guessed the same topic?
Education Game	N/A	N/A	Guessed the same topic?
Clique Form.	0	0.364 (Wundt curve)	Compare prototype&design (tolerance)

The variation of the Wundt curve named Sine Curve can also be used dynamically by adjusting the value of x_0, i.e., the preferred similarity in range [0,1] (see Algorithm 5).

General statistical methods and some graph-theoretic functions (West et al., 2001) such as degree and distance map have been used to analyse the process of simulations and their results (see Table 7.6). For example, the number of games being played are analysed in the experiment, Compositional and Holistic Language (see Section 4.1), to compare the efficiency of compositional language and holistic language, and in the experiment, Growing Population (see Section 6.1), to compare the efficiency of dynamic population and fixed population. Discrimination, consistency, density and the length of representations are used to analyse the ambiguity, completeness and extensibility of the evolved languages. The weights of relations between objects and expressions are analysed to clarify congruent and incongruent combinations in the experiment, Incongruity (see Section 5.3). The quality and quantity of collected works are analysed to find the impact of client's preference on final products in the experiment, Compositional Language for Shape Combination (see Section 4.3). Average max-degree of instances is analysed to find different effects of using hedonic functions for evaluating utterances and designs. Distance map is utilised to analyse the hierarchical structure of the evolution of multi-languages with different styles of generated design works at the sociocultural level in the experiment, Clique Formation (see Section 6.3).

7.3.4 Advantages and Disadvantages

Several key results are obtained via multi-agent simulations and statistical analysis (see Table 7.7). For example, the conclusion "question may be more important than answer" is analogised to the result "client's creative requirements (design briefs) may have more influence on interesting products than creative designing". This is obtained by comparing the degrees of instances generated in different situations, including interest evaluation of only requirements and interest evaluation of only designs. They clarify the importance of creative requirements. If client requirements remain similar or identical without any

TABLE 7.6: The analysis of the experiments

Experiment	Analysis
Compo.&Holistic	Gaming-times, discrimination, consistency, density of utterances
Compo-Represen.	Topological relations mapping compositional utterances
Compo-Shape	Success rates, the number of collected shapes&types(select-thresholds)
Ambiguity	Ambiguity of utterances, average max-degree of instances
Scalability	The range of exaggerated sizes
Incongruity	The weights of each object related with different expressions
Extensibility	The procedure of elaboration
Growing Pop.	Gaming-times with growing population and fixed large population
Education Game	$Ratio_{initiator}[teachers : students]$
Clique Form.	Hierarchical relations of agents (distance map of languages&designs)

TABLE 7.7: The results/conclusions of the experiments

Experiment	Results/conclusions
Compo.&Holistic	Compositional language is more efficient&ambiguous than holistic lang.
Compo-Represen.	HRRs represent topological relations & clarify their relationship
Compo-Shape	Client collects more types of works when requirements are moderate
Ambiguity	Question(requirement) may be more important than answer(design)
Scalability	Exaggerated sizes are generated and differentiated with modifier
Incongruity	Incongruent scenes are generated via reversed probability density
Extensibility	Objects are elaborated, moved, rotated or scaled
Growing Pop.	Growing population evolves more fast than initial large population
Education Game	Double-direction education is better than single-direction teaching
Clique Form.	Clique formation is affected by agents' tolerance & preference

interesting change, even different new solutions which are developed time after time can not improve creativity because their related requirements are fixed (Blayse and Manley, 2004). Incongruent combinations can be obtained by inverse weighted random choice of existing combinations and non-existent combinations with existent elements because incongruence is the opposite of congruence, as imagination is likely to be the opposite or "mirror" of reality.

Some advantages of these experiments are listed in Table 7.8. In particular, the weights of associations between meanings and utterances can change dynamically by applying effective learning algorithms (see Algorithm 25, 27, 26, 28) in both guessing games and generation games. Short term memory for communication preference has been used in the experiment, Clique Formation (see Section 6.3). And long term memory for dynamic categorisation have been implemented in most of the experiments, particularly Growing Neural Gas, combining the advantages of both SOM and ART, has been utilised to categorise exaggerational topics in the experiment Scalability (see Section 5.2). Weighted context free grammar has been efficiently used to extract design features and expand them to interesting compositional design concepts in the experiments Scalability (see Section 5.2), Incongruity (see Section 5.3) and Extensibility and Other Features (see Section 5.4). The hedonic functions based on the Wundt curve (see Algorithm 4) and a newly developed variation of the Wundt curve, sine curve (see Algorithm 5), have been

TABLE 7.8: The advantages of the experiments

Experiment	Advantages
Compo.&Holistic	Composition, multiple topic-selections
Compo-Represen.	HRR representations of geometric relations
Compo-Shape	Deformation and combination
Ambiguity	Analysis of degree of Graph Network
Scalability	Dynamic categorisation of sizes with probability density
Incongruity	Inverse weighted random choice of WCFG
Extensibility	WCFG & GN, gradient mutation
Growing Pop.	dynamic population, mixing naive and mature agents
Education Game	Different ratio settings for comparison
Clique Form.	STM and LTM for communication preference

utilised to evaluate the creativity of requirements or designs in the experiments, Compositional Language for Shape Combination (see Section 4.3), Ambiguity (see Section 5.1) and Clique Formation (see Section 6.3).

Several limitations of these experiments are listed in Table 7.9. For example, in the experiment of Compositional & Holistic Languages, although compositional languages are more efficient and ambiguous than holistic languages, what is the role of holistic language in the evolution of language and social creativity and what is the relationship between holistic languages and compositional languages? For example, the relation between "orange" and "reddish-yellow" could be interesting. Both of them represent the same basic meaning but may be used in different situations to express specific semantics. Is it possible that evolved languages can be more creative and robust by combining holistic and composition?

A number of topological relations are pre-generated before simulating language games in the experiment Extensibility and Other Features (see Section 5.4) compared with the experiment, Incongruity (see Section 5.3), in which the relations are developed by generating relevant expressions during guessing games because there are only a few of simple subjects (a circle, a square, a triangle and a ground). In future experiments, more various features of relations among objects, and between objects and context might be explored, extracted and generated by individuals themselves in language games that may become more diverse and creative with more possibilities.

In the experiment, Scalability (see Section 5.2), every agent has the same basic setting of the range of new exaggerated sizes in each simulation. For example, "very small" may be only related to the range [0, 0.1]. Future simulations might develop ways of letting agents categorise the same meaning with different ranges as a consequence of experience to evolve more diverse languages to study the relationship between diversity and creativity relating to the extension of forming cliques, i.e., the improvement of heterogeneity (Wijngaards et al., 2002).

TABLE 7.9: The disadvantages of the experiments

Experiment	Disadvantages/future work
Compo.&Holistic	The role of holistic language though composition is more efficient?
Compo-Represen.	Predefined topological relations. HRRs for quantity representations?
Compo-Shape	Only considering the difference of outlines
Ambiguity	Entropy or technology driven design besides demand driven?
Scalability	Great differences between individual categorizations for creative work?
Incongruity	How to grow the length of an expression to get more complex relations?
Extensibility	Predefined interesting criteria, pre-generated topological relations
Growing Pop.	Only guessing games without generation games evolving works
Education Game	Different education stages with dynamic ratio of initiator?
Clique Form.	Cliques for social creativity (diverse? Autopoiesis?)

A very long s-expression can be easily generated using a context free grammar with recursion. It is, however, necessary to explore how to control the length of expression to get meaningful topics and evolve from simple to complex meanings and representations steadily. In the experiment, Education in Guessing Games (see Section 6.2), more situations may need to be considered. For example, different stages of language games might need different strategies rather than a fixed ratio of playing initiator between teachers and students. Clique formation may be an appropriate start for studying autopoiesis and the interactions of multiple autopoietic systems in the evolution of artificial languages for creative designing.

The artificial languages evolved in the experiments are languages with limited size that emerge by a number of communications of multi-agents. Normally, they are the result of distributed conversational processes like human languages without being consciously formed (Steels, 2006a) except the involvement of context free grammar (CFG) in Experiment 5.2, 5.3 and 5.4. However most of the rules based on CFG are evolved through the simulations.

The evolved compositional languages have not been re-used outside the experiments so far. The differences between these languages and human languages are their limited size, "toy" functions and short-lived nature (Steels, 2006a). To address the limitations of these languages, more complicated association rules should be developed such as the mappings between different phrases and sentences. In other words, the model in this dissertation should be extended from generating simple compositional utterances to collecting, evaluating and building up the relationships between these compositional utterances by developing a vector space representing the frequency of combinations of different utterances such as phrases and sentences, and analogical mapping the combinations of utterances with the same or similar structures. Thus new interesting and more complicated design briefs could be generated in language games by using analogies or exploring rare combinations based on the developed vector space storing various relationships among unit-utterances and phrase-utterances. Mikolov et al. (2013a) developed a robust vector space representing the relationships of human words and phrases

by using the word2vec approaches including the continuous bag-of-words (CBOW) architecture (Mikolov et al., 2013a) and the Skip-gram model (Mikolov et al., 2013b). The vector space was enriched by using a huge number of news data for training. It connected numerous words and phrases with probabilities that can be used to find a relevant word/phase with a provided context using CBOW architecture (Mikolov et al., 2013a), find the neighbors of a word/phase, and find the fourth word by providing the first three words such as "Germany", "Berlin", "France" → "Paris" using analogy based on the Skip-gram model (Mikolov et al., 2013b).

In addition, the interaction between artificial languages and human languages might improve the capability of artificial languages for communication between agents and humans. Steels (2001b) put forward three requirements of communications between humans and robots, which are with shared situation in the real world, open-ended and speech-based. Some examples of conversations between humans and robots based on human languages were provided (Steels, 2001b). Mubin et al. (2012) designed an artificial language named Robot Interaction Language (ROILA) using the combinations of consonants and vowels such as "koloke" representing the meaning "forward" to improve the recognition of sound by robots. For example, the sound of "koloke" is clearer than that of "forward". Results showed that the human-students who learned ROILA communicated with robots using ROILA more successfully than using human languages (Mubin et al., 2012).

In terms of the same mechanism of resulting languages via distributed conversational processes, the evolution of artificial languages may be a useful method for reflecting human languages and developing some specific features in computational environment.

7.4 Reflection on the Research

A number of human phenomena could be reflected by the experiments completed in this dissertation. Based on the results of simulations, three aspects including the reflection on the evolution of compositional language, that on the exploration of language features, and that on the exploration of social creativity are discussed as follows.

7.4.1 Reflection on the Evolution of Compositional Languages

A compositional languages is more powerful than a holistic language (see Sec. 4.1) for communicating because the former enables a person to understand a sentence that s/he has never heard before. There is a one-to-one correspondence between holistic rules and meanings but a one-to-many correspondence between compositional rules and meanings that makes compositional language more expressive and creative (Putman, 2006). In addition, compositional languages reflect the structured real world such as the mapping

of compositional utterances to geometric relations in the experiment of compositional representation of rectilinear relation (see Sec. 4.2).

"Random" selection is good at distinguishing topics (see Sec. 4.1). Analogically, No strategy might be an ideal strategy for exploring a topic completely due to its non-bias. Certain strategies could have bias and limitation that result in incomplete exploration. On the contrary, random actions without strategies, i.e., doing whatever can be done could explore the environment more completely and identify the differences between each items. "Common" selection can improve consistency of an evolved language (see Sec. 4.1). It means that always selecting a common topic which shares the most number of features with other topics could enhance the consistency of a conversation.

Design could be more productive when the strategy of evaluating design is to keep a balance between mediocrity and novelty (see Sec. 4.3), i.e., improving design creativity step by step. In addition, clients' requirements may strongly affect design creativity (see Sec. 5.1). If their requirements and evaluation did not focus on creativity, the demand-driven design could not be creative. Even something novel occurs but may probably fail to be selected by the clients. So a good client is a creative client who can enhance the creativity of designers and value creative works. A creative designer can produce interesting designs to affect the client's evaluation standard. Therefore, innovation is a process of creative interactions between clients/customers and designers instead of designers' sole job.

7.4.2 Reflection on the Exploration of Language Features

Ambiguity (see Sec. 5.1) provides a space for imagination and exploring interesting concepts which may not exist in the real world. Scalability (see Sec. 5.2) supports the clarification of meanings and their relationships because they are strengthened by exaggerating them since exaggerated concepts help to distinguish different concepts effectively.

Jokes could be made or a problem could be resolved by using incongruity (see Sec. 5.3). In a number of jokes, incongruity occurs between the set-up and the punch line. For example, {set-up: [Two fish in a tank], punch: [One turns to the other and says: "Do you know how to drive this?"]}[2]. In this example, the meaning of "tank" has changed from "a fish tank" to "a vehicle tank" via punch line that surprises people. In other words, a scene can be changed by swapping the meaning of the same word to another meaning. An incongruent composition might occur when a new meaning of the scene is combined with the same object. When the incongruent meaning is clarified, people start laughing. Therefore, incongruity may cause issues but also can be used to generate interesting concepts. As the incongruity theory states, *humor is perceived at the moment*

[2]The joke is cited from http://www.richardwiseman.com/LaughLab/incon.html

of realization of incongruity between a concept involved in a certain situation and the real objects thought to be in some relation to the concept (Mulder and Nijholt, 2002).

An interesting mechanism of extensibility for language is recursion (see Sec. 5.4), which is distinguished from looping and repetition by embedding a structure within another structure of the same type. For example, [John believes [Mary thinks [Joe is handsome]]] is a result of right embedded recursion using pairs of formal notations including [S → NP, VP], and [VP → V, S]. Another example, "The cat [that the dog [that the man hit] chased] meowed" is a center embedded recursion[3]. This provides a way of understanding the creativity of language—the unbounded number of grammatical sentences being of arbitrary length. Recursion has also been used in artworks such as Escher's graphic work (Escher et al., 1982) and designs.

7.4.3 Reflection on the Exploration of Social Creativity

The result of the experiment, Growing Population (see Sec. 6.1), may reflect a creative strategy of developing a large scale company. A company could grow faster by recruiting new employees step by step and arranging co-workers with experienced employees and new naive workers rather than recruiting a great number of people once that may cost much more time for settling down as a company.

The experiment results of Education in Guessing Game (see Sec. 6.2) reflect creative human education, in which teachers encourage students to learn knowledge positively by questioning, finding new ways of understanding certain knowledge and generating new knowledge by stimulating their curiosity for the potential of developing new insight of the world due to their naive with little burden of existing knowledge. At the same time, teachers are so open minded that they can accept new valuable ideas generated by students and learn form them. However, teachers should still play the main role in education by guiding students' learning for the efficiency of knowledge transmission. In other words, a balance between self-learning and supervised learning should be considered for improving the efficiency of creative education.

Language is a tool of communication for creating, transmitting and sharing knowledge. In terms of different subjects involved in various communications, domain-specified languages for representing certain knowledge evolve (see Sec. 6.3). For example, an artist's language may be different from a scientist's language although they share some daily vocabulary. A physicist's language could also be something different from a biologist's language. When they meet together, some misunderstanding is likely to happen. But at the same time, novel ideas could be generated by comparing and connecting different new meanings exported from various domain languages. Some domain languages may

[3]The right embedded recursion and the center embedded recursion are cited from https://www.quora.com/In-what-sense-is-the-term-recursion-used-in-linguistics

be merged and regulated to a new language which becomes a new communication tool of studying and developing a novel disciplinary. The creativity of interaction between different domain languages may depend on the similarity of these languages, their complementarity, the subjects being discussed, and the participants' tolerances, curiosity and the breath of knowledge they mastered.

7.5 Evaluation of the Results

The results of this research support the creativity of evolving multiple associations between words and meanings in linguistic conceptualisation. Language as a social process could be more powerful than that as a medium for creative designing; social creativity can be more useful than individual creativity for design emergence on a large scale; and computational simulation could provide more freedom of exploring design creativity with comparable small cost compared with traditional investigation. This section describes these key results.

7.5.1 Linguistic Conceptualisation

The results of linguistic conceptualisation support the creative potential of multiple mappings between utterances and meanings, and the capability of emergence of complex concepts using language rules such as recursion and composition. They have been explored in the experiments of evolving compositional languages for creative design (see Chapter 4) and the exploration of creative features of language, in particular ambiguity with many-to-many mapping (see Section 5.1) and elaboration using context free grammar (see Section 5.4).

The cooperation of linguistic representation and visual conceptualisation such as sketch may be more productive by compensating for the limitations of each other to enlighten ideas and guide the flow of thoughts through both symbolic and visual responses. As Jackendoff (1987) claimed, linguistic representation is good at describing algebraic characteristics such as the type-token distinction while visual representation is expert in clarifying geometric characteristics such as the distinction between jogging and loping. Creative concepts could be better expressed by the coordination between linguistic representation and visual illustration.

7.5.2 Language as Social Process

The results of generating abundant interesting products and utterances through language games suggest that language as a social process may be more powerful for computational creativity than language as a medium, which is a form of creative output compared with

the former being an "engine" of creativity. As Lebuda et al. (2016) stated, linguistic social interaction is important for creativity. Even deadlocks in communication can become opportunities for creative problem solving. Language games have been utilised in most of the experiments to simulate the communication of initiators and receivers, and that of clients and designers, which lead to the evolution of artificial languages and a number of interesting conceptual designs (see Section 4.3, 5.1, 5.3, 5.4, and 6.3).

However, the utility of languages as both processes and mediums could be more dynamic and productive in terms of the grounded functions of languages as communication tools and idea catalysts. "Design briefs" may be strong catalysts for creative search and they may also be creative in themselves. This would suggest that design briefs may be both creative mediums for developing interesting creative briefs and creative tools for communication and stimulating creative search.

7.5.3 Social Creativity

Different strategies of social communications have been implemented and tested in the experiments, Growing Population (see Section 6.1), Education in Guessing Game (see Section 6.2) and Clique Formation (see Section 6.3), for exploring design creativity and diversity. Results show that participants with different roles, the inheritance of generations, and the conversation between educators and learners in double directions as well as the formation of cliques based on multi-linguistic evolution may generate designs more efficient and diverse compared with fixed population, singular direction of education and single group. Therefore, social creativity may be more effective for design communication on a large scale with dynamic change leading to emergence by the cooperation of multi-agents than individual creativity. In addition, social creativity is not only related with collaborative process but also affected by particular spatial and temporal contexts as well as cultural norms (Lebuda et al., 2016).

Individuals as the units of social communities, their personal creativity and cognitive creativity result into aggregation effect which could become either positive via cooperation or negative due to conflict to some extent. So it is important to adopt appropriate mechanisms of communication between individuals to engage individual creativity, even though the intelligence of specific individuals may not be important for social creativity. As Kratzer et al. (2004) suggested, too much communication may hinder creative behaviours due to the emergence of "group-think". And the centralization of communication results into information imbalance which could also hamper social creativity. Hence, sufficient information may need to be distributed to all entities instead of being overloaded to dominators (Kratzer et al., 2004).

7.5.4 Computational Simulation

Compared with traditional linguistic investigation, the data including process-data and result-data collected from the computational simulation of language evolution are more transparent and abundant. The initial settings provide more adjustable options. And it is considerably easier to implement the experiments repeatedly with small cost compared with traditional interview and real language games in which the collected data is normally on a small scale and incomplete like a "black box", which can only be analysed through the output rather than the inner information.

Although the computational simulation of the evolution of language provides more freedom and efficiency of exploring design creativity, its results still need to be tested in real world, which is filled with possibilities and uncertain influencing factors compared with a simplified computational environment. Though a simplified simulation can dramatically reduce the simulation time, it might also decrease the accuracy of the results (Craig, 1996). Even if the simulation results matched the results of the real environment, the mechanisms of the simulation may not be the same as that in the real world. In addition, it is still difficult to grasp the logical route connecting initial settings and final results in a simulation compared with mathematical reasoning. Thus computational simulation might lack a deep level of understanding the real world and computation itself although the "inner" process of evolving artificial languages could be investigated.

To enhance the practical value of simulation, it needs to be validated and adapted to the real environment. For example, Zagal and Ruiz-Del-Solar (2007) tried to develop the structural coupling between simulation and the real environment to increase the speed of evolving robots' adaptive behaviours. And results showed that the average difference of individual's fitness obtained in simulation versus their corresponding fitness resulting from the evaluation in the real environment decreased during the adaptation process. This provided a feasible method to integrate simulation into the real environment.

7.6 Conclusion

A computational model of evolving language for creative design at the sociocultural level has been developed and tested in the experiments of evolving compositional languages, exploring the creative features of language, e.g. ambiguity and incongruity, in conceptual design, and exploring sociocultural creativity. Results show that the impact of artificial languages on both the process and outcome of creative conceptualisation at the sociocultural level through multi-agent simulation is probably competitive with visual conceptualisation, language only as a medium, individual creativity and traditional linguistic investigation of creativity. However, the combination of linguistic conceptualisation and visual illustration, the usage of language as both process and medium, the

consideration of individual creativity when developing social creativity, and the cooperation of computational simulation and investigation may be more effective for design creativity than a single mode.

Chapter 8

Conclusions and Future Work

An artificial language system based on the Domain-Individual-Field-Interaction (DIFI) framework for creative designing at the sociocultural level has been developed in this research. And the language features including compositionality, ambiguity, incongruity, exaggeration and elaboration have been explored to generate creative conceptual designs in this system. Based on the results of this research related with its potentials and limitations, several topics including meta-creativity, positive learning, complex simulations and autonomous creative system could be explored in future study.

8.1 Conclusions

The goal of this research is to develop an artificial language system based on the Domain-Individual-Field-Interaction (DIFI) framework for creative designing that addresses some of the issues in traditional conceptual design and the currently limited application of language in computational creativity. This has been achieved by exploring grounded creative features of language with multi-agent simulation, and applying them in conceptual designing at the sociocultural level. The grounded features of language such as ambiguity, incongruity, exaggeration and elaboration at the micro-level, diversity and dynamics at the macro-level, and composition and transformation as mechanisms have been explored in this system. Ambiguity, incongruity and scalability have been evaluated and applied in conceptual design while dynamics and transformation have been tested and discussed.

In addition, the methods of representation, communication and evaluation in the evolution of artificial languages have been developed and applied to produce creative and successful results in the system. In particular, weighted context free grammar has been efficiently utilised to extract design features and expand them to interesting compositional concepts and surprising design results with the aid of inverse weighted random

choice. Both short term memory for topic selection and communication preference and long term memory for dynamic categorisation have been embedded in this system. In particular, Growing Neural Gas, combining the advantages of both SOM and ART, has been utilised to categorise exaggerational topics with the aid of probability density. The weights of associations between meanings and utterances can change dynamically by applying effective learning algorithms in both guessing games and generation games. The hedonic functions based on the Wundt curve and a newly developed variation of the Wundt curve, sine curve, have been utilised to evaluate the creativity of both products and utterances, aided by the measurement of cosine similarity and the Euclidian distance to address different situations.

The whole system can be abstracted to a hierarchical partial graph network (see Section 3.5.4) including the nodes of environment and agents, and the edges connecting features, such as the rules of generating compositional utterances and designs using context free grammar (see Section 3.2.3) particularly in the experiment, Incongruity (see Section 5.3) and the experiment, Extensibility and Other Features (see Section 5.4). The data structure of the system using only hash-table has not only enhanced the efficiency of exchanging data through each "corner" of the system but also made the results available to map to various design forms such as simplified buildings and vases sharing the same data structure. In addition, this structure also provides the potential of storing multiple languages and mapping them with each other (see Section 7.3.2). Further, it is possible to update the system to become a complete graph network including both structures and functions. The former have been realised in this thesis while the latter are tested (see Algorithm 2) and is capable of being implemented.

8.2 Future Research

This system has potential of being extended to the transformation of design space related to meta-creativity. In addition, positive rather than passive learning would be embedded into the system to develop designer-agents' capability of predicting clients' future demands. More complicated multi-agent simulations of evolving artificial languages for creative designing could be studied by developing new game types, individual roles and machine learning methods as well as increasing the data samples and population to a larger scale. Consequently, it might be possible to develop deeper and broader applications of the creative features of language in CAD and HCI based on this system to enhance human creativity.

Future Research may include investigations into meta-creativity for finding new design spaces, positive learning for predicting clients' future demands, complex simulations for exploring design emergence and diverse creativity, and the development of autonomous creative system for exploring strong artificial social intelligence.

8.2.1 Meta-Creativity

An important topic for future research is the development of computational models of meta-creativity relating to transformational creativity (Boden, 1996) and the evolution of language grammars and rules to expand design dimensions, and find new design spaces. Meta-creativity may be used to address the issue illustrated in Fig. 8.1. It shows that individuals "feel bored" after becoming familiar with all designs in a domain. This could easily occur in a small domain with simple artefacts which may be explored completely in a short time without exhausting the collective memory of agents.

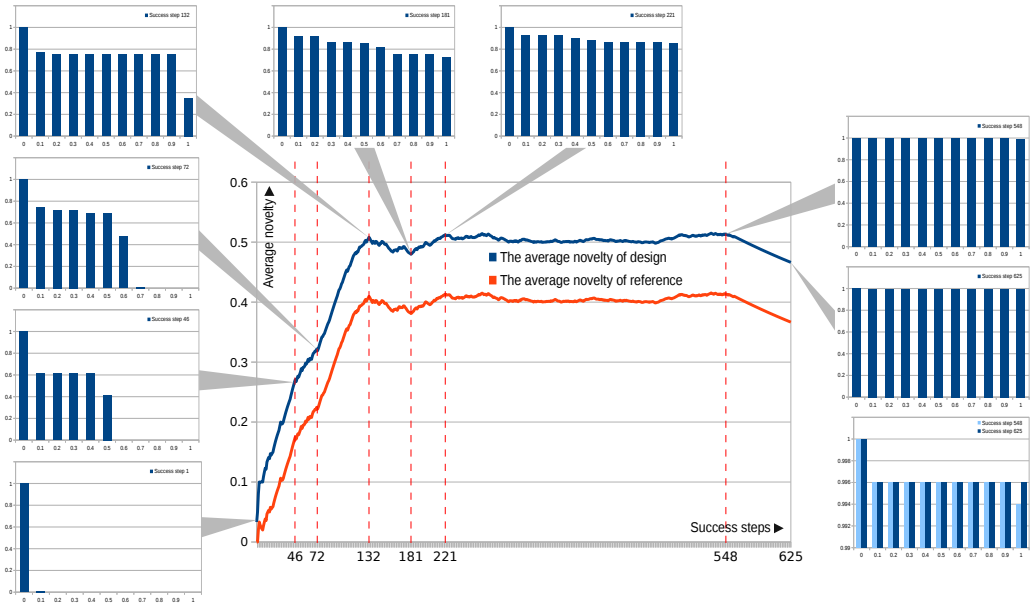


FIGURE 8.1: The trend of interestingness change

Fig. 8.1 illustrates the trend in the assessment of novelty by individual agents. The assessment relates to interestingness, via the application of a hedonic function, and creativity when combined with value. The graph illustrates three stages: the growing stage, the mature period and the decline stage. Most of explorations, i.e., generation and evaluation of novelty, occur in the first stage, at the end of which many of the designs in particular domain have been generated and evaluated. This leads to continuous increase in the criteria for evaluating interestingness although the relative level of interestingness might remain stable. For example, a client-agent's interested design works may be at a similar-yet-different range such as [0.3, 0.4] given the whole range as [0,1]. The individual's experience changes when some "new" designs are generated. These "new" designs become "old" after more similar "new" works are experienced. Compared with the initial situation, the individual's absolute interesting degree may appear to have risen, e.g., [0.35, 0.45], although the relative interesting range remains at the same level e.g. [0.3, 0.4].

When the experiences of all designs become mature, e.g. the degree of familiarity of all designs surpasses 0.6, both the success-data-averages and active-data-averages stop

increasing. This means that the client is "bored" by the "new" generated designs, which are almost all familiar. At this point, it is worthless to continue to play language games in the same way (see from step 132 to 625 in Fig. 8.1). Ideally the agents should adapt to the new environment by adapting their strategies for the generation of novelty. The designer-agents should explore new features and combinations while the client-agents should try to find new requirements related to new conceptual spaces. The exploration of novelty is related with Martindale (1990) 's "clockwork muse". The ability to successfully communicate might support the sawtooth pattern of innovation and revolution (Martindale, 1990).

To explore new design spaces, a meta-language composed of rules is proposed to evolve using extensible neural networks with the support of interactions among multi-cultural groups. New concepts would be generated and classified based on relevant new detected features which could be found using existing techniques that have been shown to be able to produce feature detectors, e.g., genetic programming (Belpaeme, 1999). These new detected features would be used to classify and categorise existing designs in novel ways to stimulate new ideas.

New features could be extracted from raw materials to obtain abstract concepts. Then they would be transformed to another domain using analogy (Grace et al., 2011). For example, when people find that the function of a computer is like a human brain, some compositional utterances such as "electrical brain", "bio-inspired brain" and "watery brain" can be generated to represent similar-yet-different meanings to clarify their relationships (Goertzel et al., 2010), thus leading to new innovations. The transfer of knowledge between domains could also be completed based on the same structure of a graph network and via the interaction between multi-cultures (i.e., multi-languages) which has been explored in the experiment, Clique Formation (see Section 6.3).

Both language rules and design rules may be detected and evolved from simple random elements (Wiggins, 2006a). For operative design, each operational concept can be represented with compositional languages using HRRs (see Section 3.2.2) aided by graph networks (see Section 3.5.4). To evolve feature detectors and decision trees via multi-agent cooperation, the agents would generate sentences representing programs or meta-programs. For example, "$(>(-\ x_{11}\ x_{01})\ (+\ x_{12}\ x_{02})\ )$" could be a relation between two edges with terminals x_{01}, x_{02}, x_{11}, x_{12}. More complicated expressions may contain both functions $+, -, *, /, >, <, =$, and conditions such as if-else and case. Building on the work presented in experiments Incongruity (see Section 5.3) and Extensibility and Other Features (see Section 5.4) agents using language games may be able to generate representations of such structures using compositional languages.

The direction of language evolution could be from embodiment to abstract meanings, and from detailed graphic symbols to abstract signals (Sternberg, 1985). For example,

schematic pictures include signs while geometric forms include shapes. Lakoff and Johnson (1999) argue that many features of natural language arise from analogies with our embodied selves and our physical understanding of our relationship to the world. Hence, grounding in the specific may be an excellent basis for the development of more abstract concepts. More abstract signals, grounded in specifics, provides a basis for knowledge transfer between domains. Language evolution can support knowledge transfer between creative domains by supporting the development of languages that capture abstract signals in such a way that it supports high level mechanisms such as analogical mapping (e.g. {1,2} → {3,?}), completions (e.g. {1,2,3,?}) and classifications (e.g. {1,2 - 3,4 or 1,3 - 2,4 or 1,4 - 2,3 or ?}) that can be translated between domains.

The direction of evolution might also be from statistics to reasoning. creative activity is often more about the process of creation than achieving a desired result that is related with the work of Amabile (1988) on the importance of intrinsic motivation in creative work. Younger children tend to rely more heavily on word association, i.e., statistical learning rather than inherent structure learning, and less heavily on logical inference (Sternberg, 1985). Sternberg (1985) suggests that statistical association might be more creative than logical reasoning in terms of most new utterances created by young generations. One source of the observed decline in error rates with age is the increased use of more nearly complete information processing and a more reflective and less impulsive cognitive style. Fewer mistakes are likely to be related to less progress (Sternberg, 1985). Is this the case, or do we need to consider different types of mistakes?

8.2.2 Positive Learning

A client's brief may be parsed to similar meanings mapping to design works. It is necessary, however, to address the question of how to predict the client's taste in novelty (i.e., the ratio of similar to difference) to generate satisfied works. The learning strategy in this thesis might be described as passive learning because change in the weights of relevant meanings is followed by strengthening or weakening the successful topics or the failure topics as decided by the client. Future research would investigate how designer-agents can analyse the history of interactions with the client to predict the client's future requirements. For example, an agent who is capable of anticipating the aesthetic tastes of other agents would be more successful than other agents. This seems very reasonable. But there is some nuance to this, for example, should an agent learn the aesthetic tastes of individual agents and optimise for them, or should they attempt to learn general aesthetic preferences to appeal to as many potential consumers as possible in a system that support designer-consumer relations besides client-designer ones?

The recent development of approaches to using neural networks to build "world models" can inform the predictive abilities of a system within a smaller domain. This is a potentially interesting approach to the development of curious agents. For example, the

development of grounded descriptions of colours based on 20 million images is important because agents exposed to such databases can take advantage of these types of grounded representations to play more sophisticated language games; it also potentially opens up the ability for artificial agents to communicate in grounded ways with humans exploring a domain, e.g., that of colours or images. Google and other companies used similar technology to revolutionise the way people search for images. This would be an approach to "positive learning" because it is an approach to grounding language in a common set of experiences that are available to both artificial agents and humans. This type of "positive learning" is about learning general biases as a foundation for exploring specific features of a creative domain. Hadjeres and Pachet (2016) used deep learning and style transfer in music. It shows that "world models" in neural networks can significantly benefit the exploration of creative domains.

The availability of large data sets to learn from is driving deep learning but work still needs to be done to realise the potential of these systems as the foundation for creative computational systems.

8.2.3 Complex Simulations

Future research could explore more complicated interactions between individuals in artificial creative systems. First, different types of language games can be implemented in parallel rather than series. For instance, if the success rate of communication between two agents is over 70%, and they have already successfully communicated many times, an appropriate strategy may be for these agents to play generation games while other agents continue to play guessing games.

New language games could also be developed, e.g., such a language game might involve three seemingly unrelated words being presented to see if an agent can derive a fourth word connecting them (Kaufman, 2009).

Each agent's categorisations could be generated by selecting different machine learning methods depending on the change in the success rate. And the parameters of each method could be adjusted, e.g. ART tolerance depends on adjustable vigilance. Hybrid machine learning systems might provide a promising model. In terms of the composition of a field, it would allow for learners to specialise in different techniques and for fields to change strategies over time. At the same time, the novelty of categorisation, i.e., the results of machine learning, may be evaluated by comparing new categorisations with previous categorisations according to the measurement of their similarity. The adaptation of learning parameters and novelty categorisation as a type of machine learning, i.e., Hyperparameter Optimisation (Bergstra et al., 2011), has been sufficiently well-established. This concept would need to be adapted to take into account the intrinsically motivated nature of curious agents and the continuous learning required, which

hasn't been the focus of this type of optimisation to date—presenting an opportunity for future research.

More complicated individual roles (e.g. customers, consultants, sales people and engineers as well as clients and designers) and types (e.g. naive and mature, nature and nurture) could be simulated to explore the impact of role structure on social creativity. For example, some individual divisions may be senior designers and junior designers (mature and naive, competitive and cooperative) as well as naive clients and experienced clients, etc.

Another simulation involves mixing nature with nurture, i.e., next generation generated by genetic algorithm or by educating naive agents. By utilising genetic algorithm, more descendants will be generated via the crossover of two agents due to their high communication success rate ($> 70\%$), high communication frequency and high success rates of playing language games with other agents, as well as the mutation of creative agents. Alternatively, one could try to keep evolving parts of failure agents and adjust the ratio of loser-descendants and winner-descendants to determine the best ratio under different conditions. To explore the potential for exploring the effects of combining inherited and learned traits, a possible simulation might combine the generation of individuals using evolutionary algorithms combined with varying implementations of education (see Section 6.2). This might permit the exploration of the evolution of different traits that allow an agent to take advantage of the education system used by an artificial creative system, and explore the different strategies for selecting parents of a new individual.

More large scale populations (e.g. with more than 1000 individuals) can be simulated dynamically as in the Experiment, Growing Population (see Section 6.1). This could provide a basis for the study of dynamic populations with extensible agent-roles to address the following questions: Does social creativity or a high level of social creativity depend on the size of a population? What population size is required for certain kinds of creativity? The emergence of creative behaviours may occur relatively easily in large populations for three reasons. Firstly, given a diversity of agent experiences there are greater opportunities for agents to specialise to serve a sub-population, potentially leading to different roles emerging for agents across the population. Within the context of communication networks global communication has increasingly allowed creative individuals to reach a sufficiently large (but highly selective) sub-population of consumers that they can support themselves without the need to cater to mass tastes (Kelly, 2008). Secondly, increasing demands are placed on technology to exploit the environment more fully. Thirdly, with large-scale and long-running simulations there is the potential to explore features of H-creativity (Boden, 2004), such as the re-discovery of previously unheralded creative ideas, or ideas that only become creative when considered within a later social context. In addition, the sudden illumination, in which grand ideas appear, occurs only after a long incubation period of subconscious exploration. It could be regarded as historical emergence.

8.2.4 Autonomous Creative System

An autonomous creative system (Smithers, 1997) is self-governing and open to communication. Free from interference, an autonomous creative system can develop its own values, meanings, languages and practices. An autonomous creative system could be developed by starting with a few of simple rules and iteratively evolving rules that govern the processes of generation, evaluation and communication (Saunders, 2012). Oudeyer and Kaplan (2006) conducted an experiment in which an autonomous system was developed from a few fixed rules into a complicated structure.

Interacting with an autonomous creative system requires a period of mutual learning and negotiation through repeated communications (Smithers, 1997). Any autonomous creative system may have the features of one or more coupled autopoietic systems; specifically, the creative process could be confined to its boundary. This raises significant challenges for the evaluation of the creativity of an autonomous system. Clearly, ad hoc evaluations based on versions of the Turing Test could be of little or no use in determining the creativity of such systems. Even if, as Pease et al. (2001) suggest, these tests include an interactive component, the products of the system and the responses of its members cannot be assessed by humans uncoupled from the system. Thus, the evaluation of the creativity of autopoietic systems would require us to first achieve a coupling between the system and artificial agents or humans, as a means of establishing the necessary shared frame of reference (Saunders, 2012).

In an autonomous system, the role of social communication in designing, could be performed via large scale simulation of curious design agents who form the system that is large enough to allow individuals to join and leave while the system remains intact (Saunders, 2002). At the same time, the system could evolve and renew based on both its own recycling structure and its response to other autonomous systems and the environment. The simulations may suggest the importance of communications between different domains for design creativity. The simulation of a simple autonomous system is proposed to test the relationship between autopoiesis and design creativity while the simulation of dual autonomous systems interacting with each other would be used to study the creativity of interacting multiple design systems via merger, interchange, cooperation and heterogeneity.

Multiple autonomous systems might be connected and integrated by breaking their borders. Each autonomous system is in harmony and has its own identity. However, a mature system could collapse and be replaced by a new system or it could evolve into a new system by changing the borders and recombining different sub-systems. Thus, new meanings are beyond different self-identity. Sub-systems could be presented via new compositions of topics and contexts in different domains. This approach can raise the study of creativity from the interaction of individuals to the interaction of fields. The

studies of super-organisms in a-life (Ruiz-Mirazo et al., 2000) would ground this idea of simulation at both the level of the individual and the field at the same time.

8.3 Summary

A computational model of creative design based on the Domain-Individual-Field-Interaction (DIFI) framework as a sociocultural process involving the evolution of language has been developed to investigate the role of language as a process rather than a medium in conceptual designing. The language games including guessing games and generation games are integrated in the process of representation, communication and evaluation of designs.

Based on the computational model, three aspects of experiments have been completed to support the role of the evolution of language in social creativity for designing. First, the experiments of evolving compositional languages show that composition is more efficient than holistic for evolving associations between utterances and designs with the similar-yet-different selection criteria. Then, the experiments of exploring creative features of language have applied ambiguity, exaggeration, incongruence and elaboration in creative designing. Last but not least, the experiments of evolving language at the sociocultural level have developed some strategies for improving the efficiency and diversity of creative design.

Bibliography

Achermann, F., Lumpe, M., Schneider, J.-G., and Nierstrasz, O. (2001). Piccola-a small composition language. *Formal Methods for Distributed Processing—A Survey of Object-Oriented Approaches*, pages 403–426.

Allaire, Y. and Firsirotu, M. E. (1984). Theories of organizational culture. *Organization studies*, 5(3):193–226.

Amabile, T. (1988). The intrinsic motivation principle of creativity. *Research in organizational behavior*, 10.

Bagha, K. N. (2011). A short introduction to semantics. *Journal of Language Teaching and Research*, 2(6):1411–1419.

Barsalou, L. W. (1999). Language comprehension: Archival memory or preparation for situated action?

Barthes, R. (1977). *Elements of semiology*. Macmillan.

Belpaeme, T. (1999). Evolution of visual feature detectors. In *University of Birmingham School of Computer Science technical*. Citeseer.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Berlyne, D. E. (1971). *Aesthetics and psychobiology*, volume 336. JSTOR.

Bird, S., Klein, E., and Loper, E. (2009). Natural language processing with python—analyzing text with the natural language toolkit o'reilly media.

Blayse, A. M. and Manley, K. (2004). Key influences on construction innovation. *Construction innovation*, 4(3):143–154.

Boden, M. A. (1996). *Dimensions of creativity*. MIT Press.

Boden, M. A. (2004). *The creative mind: Myths and mechanisms*. Routledge.

Borensztajn, G. (2006). Luc steels, the talking heads experiment and cognitive philosophy. *Philosophy*, pages 1–13.

Bradski, G. et al. (2000). The opencv library. *Doctor Dobbs Journal*, 25(11):120–126.

Brill, E. and Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Burton, R. M. and Obel, B. (1995). The validity of computational models in organization science: From model realism to purpose of the model. *Computational & Mathematical Organization Theory*, 1(1):57–71.

Byrne, D. and Callaghan, G. (2013). *Complexity theory and the social sciences: The state of the art*. Routledge.

Candy, L. and Bilda, Z. (2009). Understanding and evaluating creativity. In *Proceedings of the seventh ACM conference on Creativity and cognition*, pages 497–498. ACM.

Cangelosi, A. (2001). Evolution of communication and language using signals, symbols, and words. *Evolutionary Computation, IEEE Transactions on*, 5(2):93–101.

Cangelosi, A. and Parisi, D. (2012). *Simulating the evolution of language*. Springer Science & Business Media.

Carpenter, G. A. and Grossberg, S. (2016). Adaptive resonance theory. In *Encyclopedia of Machine Learning and Data Mining*, pages 1–17. Springer.

Carter, R. and McCarthy, M. (2004). Talking, creating: interactional language, creativity, and context. *Applied Linguistics*, 25(1):62–88.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005(598-603):18.

Christophe, F., Bernard, A., and Coatanéa, É. (2010). Rfbs: A model for knowledge representation of conceptual design. *CIRP Annals-Manufacturing Technology*, 59(1):155–158.

Clark, A. (1997). *Being there: Putting brain, body, and world together again*. MIT press.

Cohen, W., Ravikumar, P., and Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78.

Colton, S. (2008). Creativity versus the perception of creativity in computational systems. In *AAAI spring symposium: creative intelligent systems*, pages 14–20.

Colton, S. (2012). The painting fool: Stories from building an automated painter. In *Computers and creativity*, pages 3–38. Springer.

Colton, S., de Mántaras, R. L., Stock, O., et al. (2009). Computational creativity: Coming of age. *AI Magazine*, 30(3):11–14.

Colton, S., Pease, A., and Charnley, J. (2011). Computational creativity theory: The face and idea descriptive models. In *Proceedings of the Second International Conference on Computational Creativity*, pages 90–95.

Colton, S., Wiggins, G. A., et al. (2012). Computational creativity: The final frontier? In *ECAI*, volume 12, pages 21–26.

Conte, R., Gilbert, N., Bonelli, G., Cioffi-Revilla, C., Deffuant, G., Kertesz, J., Loreto, V., Moat, S., Nadal, J.-P., Sanchez, A., et al. (2012). Manifesto of computational social science. *The European Physical Journal Special Topics*, 214(1):325–346.

Corneli, J. (2016). An institutional approach to computational social creativity. *arXiv preprint arXiv:1605.02309*.

Craig, D. C. (1996). *Extensible hierarchical object-oriented logic simulation with an adaptable graphical user interface*. PhD thesis, Memorial University of Newfoundland.

Csikszentmihalyi, M. (1996). *Creativity: The work and lives of 91 eminent people*. HarperCollins Publishers.

Csikszentmihalyi, M. (1999). *A Systems Perspective on Creativity*, pages 313–335. Cambridge University Press, Cambridge.

Cushing, J. T. (1998). Philosophical concepts in physics. *Cambridge: CUP*.

Davidsson, P. (2002). Agent based social simulation: A computer science view. *Journal of artificial societies and social simulation*, 5(1).

de Boer, B. (2000). Emergence of vowel systems through self-organisation. *AI Communications*, 13(1):27–39.

De Boer, B. (2001). *The origins of vowel systems*. Number 1. Oxford University Press.

DeGraff, J. and Lawrence, K. A. (2002). *Creativity at work: Developing the right practices to make innovation happen*, volume 28. John Wiley & Sons.

Den Ouden, B. (1975). *Language and creativity*. De Ridder.

Dewey, J. (2004). *Democracy and education*. Courier Corporation.

Dewey, J. and Small, A. W. (1897). *My pedagogic creed*. Number 25. EL Kellogg & Company.

Di Mari, A. (2013). *Operative design: a catalogue of spatial verbs*. BIS Publishers.

Dorst, K. and Cross, N. (2001). Creativity in the design process: co-evolution of problem–solution. *Design studies*, 22(5):425–437.

Duarte, J. M., Santos, J. B. d., and Melo, L. C. (1999). Comparison of similarity coefficients based on rapd markers in the common bean. *Genetics and Molecular Biology*, 22(3):427–432.

Einstein, A. and Infeld, L. (1971). *The evolution of physics: The growth of ideas from early concepts to relativity and quanta*. CUP Archive.

Escher, M. C., Bool, F., and Locher, J. (1982). *MC Escher: His Life and Complete Graphic Work: with a Fully Illustrated Catalogue*. HN Abrams New York.

Feldman, D. H., Csikszentmihalyi, M., and Gardner, H. (1994). *Changing the world: A framework for the study of creativity*. Praeger Publishers/Greenwood Publishing Group.

Finke, R. A., Ward, T. B., and Smith, S. M. (1992). Creative cognition: Theory, research, and applications.

Fischer, G. (2005a). Distances and diversity: sources for social creativity. In *Proceedings of the 5th conference on Creativity & cognition*, pages 128–136. ACM.

Fischer, G. (2005b). *Social creativity: Making all voices heard*. Citeseer.

Fischer, G., Giaccardi, E., Eden, H., Sugimoto, M., and Ye, Y. (2005). Beyond binary choices: Integrating individual and social creativity. *International Journal of Human-Computer Studies*, 63(4):482–512.

Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT Press.

Forney, G. (1966). Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12(2):125–131.

Fortuny, J. (2010). On the duality of patterning. *Structure Preserved: Studies in syntax for Jan Koster*, 164:131–140.

Fritzke, B. et al. (1995). A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7:625–632.

Gardner, H. (1993). *Creating minds: an anatomy as seenthrough the lives of Freud, Einstein, Picasso, Stravinsky, Eliot, Graham and Gandhi*. HarperCollinsPublishers.

Gardner, H. (2011). *Creating minds: An anatomy of creativity seen through the lives of Freud, Einstein, Picasso, Stravinsky, Eliot, Graham, and Gandhi*. Basic Books.

Gassmann, O. (2001). Multicultural teams: Increasing creativity and innovation by diversity. *Creativity and Innovation Management*, 10(2):88–95.

Gero, J. (1994). Computational models of creative design processes. *AI in Creativity. Kluwer, Dordrecht*, pages 269–281.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI magazine*, 11(4):26.

Gero, J. S. and Fujii, H. (2000). A computational framework for concept formation for a situated design agent. *Knowledge-Based Systems*, 13(6):361–368.

Gilbert, N. (1999). Simulation: A new way of doing social science. *American Behavioral Scientist*, 42(10):1485–1487.

Gilbert, N. and Troitzsch, K. (2005). *Simulation for the social scientist*. McGraw-Hill Education (UK).

Glenberg, A. M. and Kaschak, M. P. (2002). Grounding language in action. *Psychonomic bulletin & review*, 9(3):558–565.

Goertzel, B., Lian, R., Arel, I., De Garis, H., and Chen, S. (2010). A world survey of artificial brain projects, part ii: Biologically inspired cognitive architectures. *Neurocomputing*, 74(1):30–49.

Gotts, N. M., Polhill, J. G., and Law, A. N. R. (2003). Agent-based simulation in the study of social dilemmas. *Artificial Intelligence Review*, 19(1):3–92.

Grace, K., Saunders, R., and Gero, J. (2011). Interpretation-driven visual association. In *Proceedings of the Second International Conference on Computational Creativity*, pages 132–134. Citeseer.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological cybernetics*, 23(3):121–134.

Guilford, J. P. (1967). The nature of human intelligence. *McGraw-Hill*.

Hadjeres, G. and Pachet, F. (2016). Deepbach: a steerable model for bach chorales generation. *arXiv preprint arXiv:1612.01010*.

Hannah, G. G. (2002). *Elements of design: Rowena Reed Kostellow and the structure of visual relationships*. Princeton Architectural Pr.

Hayes, J. R. (1989). *Cognitive processes in creativity*. Springer.

Helmore, G. A. (1969). *Piaget-a practical consideration*. Pergamon Press.

Herrmann, T. (1997). Communicable models for cooperative processes. In *HCI (1)*, pages 285–288.

Hockett, C. F. and Altmann, S. A. (1968). A note on design features. *Animal communication: Techniques of study and results of research*, pages 61–72.

Hori, K. (1994). A system for aiding creative concept formation. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(6):882–894.

Iba, T. (2010). An autopoietic systems theory for creativity. *Procedia-Social and Behavioral Sciences*, 2(4):6610–6625.

Irwing, B. . (2010). http://www.e-metrixx.com/creativity-profit/me2-spec/.

Jackendoff, R. (1987). On beyond zebra: The relation of linguistic and visual information. *Cognition*, 26(2):89–114.

Jordanous, A. and Keller, B. (2012). What makes musical improvisation creative. *Journal of Interdisciplinary Music Studies*, 6(2):151–175.

Jordanous, A. K. (2013). *Evaluating computational creativity: a standardised procedure for evaluating creative systems and its application*. PhD thesis, University of Sussex.

Kahl, C. H. and Hansen, H. (2015). Simulating creativity from a systems perspective: Cresy. *Journal of Artificial Societies and Social Simulation*, 18(1):4.

Kan, J. W. and Gero, J. S. (2009). Using entropy to measure design creativity.

Kaufman, J. C. (2009). *Creativity 101*. Springer Publishing Company.

Kaufman, J. C. and Sternberg, R. J. (2010). *The Cambridge handbook of creativity*. Cambridge University Press.

Kelly, K. (2008). One thousand true fans. *KK. org. March*, 4.

Kim, K. H. (2006). Can we trust creativity tests? a review of the torrance tests of creative thinking (ttct). *Creativity research journal*, 18(1):3–14.

Kindratenko, V. (1997). *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerp, Belgium.

Kirby, S. (1998). Learning , bottlenecks and the evolution of recursive syntax. *Evolution*.

Kirby, S. (2001). Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110.

Kirby, S., Cornish, H., and Smith, K. (2008). Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686.

Kirby, S., Griffiths, T., and Smith, K. (2014). Iterated learning and the evolution of language. *Current opinion in neurobiology*, 28:108–114.

Kliemt, H. (1996). Simulation and rational practice. In *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, pages 13–27. Springer.

Kononen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9).

Kratzer, J., Leenders, O. T. A., and Van Engelen, J. M. (2004). Stimulating the potential: Creative performance and communication in innovation teams. *Creativity and Innovation Management*, 13(1):63–71.

Lakoff, G. and Johnson, M. (1999). *Philosophy in the flesh: The embodied mind and its challenge to western thought*. Basic books.

Langton, C. G. et al. (1989). *Artificial life*. Addison-Wesley Publishing Company Redwood City, CA.

Lebuda, I., Galewska-Kustra, M., and Glăveanu, V. P. (2016). Creativity and social interactions. *Creativity. Theories–Research-Applications*, 3(2):187–193.

Lesgold, A. (1988). 7 problem solving. *The psychology of human thought*, page 188.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Levy, S. D. and Kirby, S. (2006). Evolving distributed representations for language with self-organizing maps. In *Symbol Grounding and Beyond*, pages 57–71. Springer.

Lidov, D. (1999). *Elements of semiotics*. Palgrave Macmillan.

Lindqvist, G. (2003). Vygotsky's theory of creativity. *Creativity Research Journal*, 15(2-3):245–251.

Liu, L., Zhu, F., Chen, C., Yan, X., Han, J., Philip, S. Y., and Yang, S. (2010). Mining diversity on networks. In *Database Systems for Advanced Applications*, pages 384–398. Springer.

Liu, Y.-T. (2000). Creativity or novelty?: Cognitive-computational versus social-cultural. *Design Studies*, 21(3):261–276.

Luff, P., Frohlich, D., and Gilbert, N. G. (2014). *Computers and conversation*. Elsevier.

Luhmann, N. (1986). The autopoiesis of social systems. *Sociocybernetic paradoxes*, pages 172–192.

Luhmann, N. (1995). *Social systems*. Stanford University Press.

MacDorman, K. F. (2007). Life after the symbol system metaphor. *Interaction Studies 8.1*.

Machado, P., Nunes, H., and Romero, J. (2010). Graph-based evolution of visual languages. In *European Conference on the Applications of Evolutionary Computation*, pages 271–280. Springer.

Maher, M. L. (2012). Computational and collective creativity: Who's being creative. In *International Conference on Computational Creativity*, pages 67–71. Citeseer.

Mamykina, L., Candy, L., and Edmonds, E. (2002). Collaborative creativity. *Communications of the ACM*, 45(10):96–99.

Marsland, S., Nehmzow, U., and Shapiro, J. (1999). A model of habituation applied to mobile robots. *Proceedings of Towards Intelligent Mobile Robots*.

Marsland, S., Nehmzow, U., and Shapiro, J. (2000). Detecting novel features of an environment using habituation. In *Proc. Simulation of Adaptive Behavior*.

Martindale, C. (1990). *The clockwork muse: The predictability of artistic change*. Basic Books.

Maturana, H. R. and Varela, F. J. (1991). *Autopoiesis and cognition: The realization of the living*, volume 42. Springer Science & Business Media.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miranda, E. R., Kirby, S., and Todd, P. (2003). On computational models of the evolution of music: From the origins of musical taste to the emergence of grammars. *Contemporary Music Review*, 22(3):91–111.

Mitchell, W. T. (1995). *Picture theory: Essays on verbal and visual representation*. University of Chicago Press.

Mott, R. (2005). Smith–waterman algorithm. *eLS*.

Mubin, O., Bartneck, C., Feijs, L., Hooft van Huysduynen, H., Hu, J., and Muelver, J. (2012). Improving speech recognition with the robot interaction language. *Disruptive Science and Technology*, 1(2):79–88.

Mulder, M. P. and Nijholt, A. (2002). Humour research: State of art. Technical report, University of Twente, Centre for Telematics and Information Technology.

Murali, P. (2014). Cognitive cooking: Applying computational creativity to recipe generation. *IBM Research Blog*.

Narzt, W., Wilflingseder, U., Pomberger, G., Kolb, D., and Hörtner, H. (2010). Self-organising congestion evasion strategies using ant-based pheromones. *IET Intelligent Transport Systems*, 4(1):93–102.

Neath, I. (1998). *Human memory: An introduction to research, data, and theory.* Thomson Brooks/Cole Publishing Co.

Noble, W. and Davidson, I. (1996). *Human evolution, language and mind: a psychological and archaeological inquiry.* CUP Archive.

Oudeyer, P.-Y. and Kaplan, F. (2006). Discovering communication. *Connection Science*, 18(2):189–206.

Pang-Ning, T., Steinbach, M., Kumar, V., et al. (2006). Introduction to data mining. In *Library of Congress*, page 74.

Patel, A. X., Gibson, E., Ratner, J., Besson, M., and Holcomb, P. J. (1998). Processing syntactic relations in language and music: An event-related potential study. *Cognitive Neuroscience, Journal of*, 10(6):717–733.

Pearce, M. T. and Wiggins, G. A. (2007). Evaluating cognitive models of musical composition. In *Proceedings of the 4th international joint workshop on computational creativity*, pages 73–80. Goldsmiths, University of London.

Pease, A. and Colton, S. (2011). On impact and evaluation in computational creativity: A discussion of the turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and Philosophy.*

Pease, A., Colton, S., Ramezani, R., Charnley, J., and Reed, K. (2013). A discussion on serendipity in creative systems. In *Proceedings of the fourth international conference on computational creativity*, pages 64–71.

Pease, A., Winterstein, D., and Colton, S. (2001). Evaluating machine creativity. In *Workshop on Creative Systems, 4th International Conference on Case Based Reasoning*, pages 129–137.

Piaget, J. (1977). *The development of thought: Equilibration of cognitive structures.(Trans A. Rosin).* Viking.

Plate, T. A. (1995). Holographic reduced representations. *Neural Networks, IEEE Transactions on*, 6(3):623–641.

Plate, T. A. (2003). *Holographic Reduced Representation: Distributed representation for cognitive structures.* CSLI Publications Stanford, CA, USA.

Plattner, H., Meinel, C., and Leifer, L. (2010). *Design thinking: understand–improve–apply.* Springer Science & Business Media.

Poincaré, H. (1913). The foundations of science: Science and hypothesis. *English Trans.*

Putman, R. (2006). The emergence of language. *Physics 569: Emergent States of Matter.*

Ramachandran, V. S. and Hubbard, E. M. (2001). Synaesthesia–a window into perception, thought and language. *Journal of Consciousness Studies*, 8(12):3–34.

Reynolds, C. (1986). Boids: Flocks. *Herds and Schools–a Distributed Behavorial Model.*

Rhodes, M. (1961). An analysis of creativity. *The Phi Delta Kappan*, 42(7):305–310.

Ritchie, G. (2001). Assessing creativity. In *Proc. of AISB'01 Symposium*. Citeseer.

Ritchie, G. (2006). The transformational creativity hypothesis. *New Generation Computing*, 24(3):241–266.

Ritchie, G. (2007). Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99.

Roser, F. and Hebela, N. M. (2015). The whole is other than the sum of the parts. *World neurosurgery.*

Ruiz-Mirazo, K., Etxeberria, A., Moreno, A., and Ibáñez, J. (2000). Organisms and their place in biology. *Theory in biosciences*, 119(3-4):209–233.

Runco, M. A. (2003a). Creativity, cognition, and their educational implications. *The educational psychology of creativity*, pages 25–56.

Runco, M. A. (2003b). Education for creative potential. *Scandinavian Journal of Educational Research*, 47(3):317–324.

Saunders, R. (2002). *Curious Design Agents and Artificial Creativity: A Synthetic Approach to the Study of Creative Behaviour.* PhD thesis, The University of Sydney.

Saunders, R. (2011). Artificial creative systems and the evolution of language. In *Proceedings of the second international conference on computational creativity. México City*, pages 36–41.

Saunders, R. (2012). Towards autonomous creative systems: A computational approach. *Cognitive Computation*, 4(3):216–225.

Saunders, R. and Bown, O. (2015). Computational social creativity. *Artificial life*, 21(3):366–378.

Saunders, R. and Gero, J. S. (2001a). Artificial creativity: A synthetic approach to the study of creative behaviour. *Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney*, pages 113–139.

Saunders, R. and Gero, J. S. (2001b). A curious design agent. In *CAADRIA*, volume 1, pages 345–350.

Saunders, R. and Grace, K. (2008). Towards a computational model of creative cultures. In *AAAI Spring Symposium: Creative Intelligent Systems*, pages 67–74.

Sawyer, R. K. (2004). Social explanation and computational simulation. *Philosophical Explorations*, 7(3):219–231.

Scott-Phillips, T. C. and Blythe, R. A. (2013). Why is combinatorial communication rare in the natural world, and why is language an exception to this trend? *Journal of The Royal Society Interface*, 10(88):20130520.

Seker, S. E., Altun, O., Ayan, U., and Mert, C. (2014). A novel string distance function based on most frequent k characters. *arXiv preprint arXiv:1401.6596*.

Sherzer, L. (2009). *The Key to Language: An Essay on the Meaning That Exists Prior to and Independent of Language*. Laurence Sherzer.

Simonton, D. K. (1990). *Psychology, science, and history: An introduction to historiometry.* Yale University Press.

Simonton, D. K. (1999). *Origins of genius: Darwinian perspectives on creativity*. Oxford University Press.

Simpson, E. H. (1949). Measurement of diversity. *Nature*.

Sipser, M. (1997). Introduction to the theory of computation. *Boston, MA: PWS Pub.*, 49.

Smagorinsky, P. (2011). Vygotsky's stage theory: The psychology of art and the actor under the direction of perezhivanie. *Mind, Culture, and Activity*, 18(4):319–341.

Smith, K., Kirby, S., and Brighton, H. (2003). Iterated learning: A framework for the emergence of language. *Artificial life*, 9(4):371–386.

Smithers, T. (1997). Autonomy in robots and other agents. *Brain and Cognition*, 34(1):88–106.

Sosa, R. and Gero, J. S. (2005). A computational study of creativity in design: The role of society. *AIE EDAM*, 19(04):229–244.

Stacey, M., Eckert, C., and McFadzean, J. (1999). Sketch interpretation in design communication. In *Proceedings of the 12th International Conference on Engineering Design*, volume 2, pages 923–928.

Steels, L. (1995). A self-organizing spatial vocabulary. *Artificial life*, 2(3):319–332.

Steels, L. (1998). The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 1(2):169–194.

Steels, L. (2000). Language as a complex adaptive system. In *International Conference on Parallel Problem Solving from Nature*, pages 17–26. Springer.

Steels, L. (2001a). Language games for autonomous robots. *IEEE Intelligent systems*, 16(5):16–22.

Steels, L. (2001b). Social learning and verbal communication with humanoid robots. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 335–342.

Steels, L. (2006a). How to do experiments in artificial language evolution and why. In *Proceedings of the 6th International Conference on the Evolution of Language*, pages 323–332.

Steels, L. (2006b). Semiotic dynamics for embodied agents. *IEEE Intelligent Systems*, 21(3):32–38.

Steels, L. (2012). *Experiments in cultural language evolution*, volume 3. John Benjamins Publishing.

Steels, L. (2016). Human language is a culturally evolving system. *Psychonomic Bulletin & Review*, pages 1–4.

Steffensen, S. V. and Fill, A. (2014). Ecolinguistics: the state of the art and future horizons. *Language sciences*, 41:6–25.

Sternberg, R. J. (1985). *Beyond IQ: A triarchic theory of human intelligence*. CUP Archive.

Stiny, G. et al. (1980). Introduction to shape and shape grammars. *Environment and planning B*, 7(3):343–351.

Sun, R. and Naveh, I. (2007). Social institution, cognition, and survival: a cognitive–social simulation. *Mind & Society*, 6(2):115–142.

Swenson, R. (1992). Autocatakinetics, yes—autopoiesis, no: Steps toward a unified theory of evolutionary ordering. *International Journal Of General System*, 21(2):207–228.

Techtarget (2003). Converting business requirements into functional specs. *http://searchsap.techtarget.com/answer/Converting-business-requirements-into-functional-specs*.

Temperley, N. and Temperley, D. (2011). Music-language correlations and the "scotch snap". *Music Perception*, 29(1):51–63.

Torrance, E. P. (1968). *Torrance tests of creative thinking.* Personnel Press, Incorporated.

Turkle, S., Clancey, W. J., Helmreich, S., Loukissas, Y. A., and Myers, N. (2009). *Simulation and its Discontents.* mit Press Cambridge, MA.

Varshney, L. R., Pinel, F., Varshney, K. R., Bhattacharjya, D., Schoergendorfer, A., and Chee, Y.-M. (2013). A big data approach to computational creativity. *arXiv preprint arXiv:1311.1213.*

Ventura, D. A. (2008). A reductio ad absurdum experiment in sufficiency for evaluating (computational) creative systems. *Computational Creativity.*

Vogt, P. (2005). The emergence of compositional structures in perceptually grounded language games. *Artificial intelligence*, 167(1):206–242.

Vyas, D., van der Veer, G., and Nijholt, A. (2013). Creative practices in the design studio culture: collaboration and communication. *Cognition, Technology & Work*, 15(4):415–443.

Vygotsky, L. (1971). The psychology of art (scripta technica, inc., trans.). *Cambridge, MA 8c London: MIT press.(Original work published 1925).*

Vygotsky, L. S. (1986). *Thought and language (rev. ed.).* Cambridge, MA: MIT Press.

Wallas, G. (1926). *The art of thought.* J. Cape.

Watts, C. and Gilbert, N. (2014). *Simulating innovation: Computer-based tools for rethinking innovation.* Edward Elgar Publishing.

West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

Wheaton, K. (2014). Three convergent thinking techniques every analyst should master. *http://sourcesandmethods.blogspot.com.au/2014/09/three-convergent-thinking-techniques.html.*

Wiggins, G. A. (2006a). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458.

Wiggins, G. A. (2006b). Searching for computational creativity. *New Generation Computing*, 24(3):209–222.

Wijngaards, N. J., Overeinder, B. J., van Steen, M., and Brazier, F. M. (2002). Supporting internet-scale multi-agent systems. *Data & Knowledge Engineering*, 41(2):229–245.

Wittgenstein, L. (1958). *Philosophical investigations.* Blackwell Oxford.

Wooldridge, M. (2009). *An introduction to multiagent systems.* John Wiley & Sons.

Wooldridge, M. J. (2002). *An introduction to multiagent systems*. Wiley.

Zadeh, L. A. (1996). *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, volume 6. World Scientific.

Zagal, J. C. and Ruiz-Del-Solar, J. (2007). Combining simulation and reality in evolutionary robotics. *Journal of Intelligent & Robotic Systems*, 50(1):19–39.

Zhang, A. and Saunders, R. (2012). Towards the evolution of a language for creative design. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–6. IEEE.

Zhang, A. and Saunders, R. (2014). Exploring conceptual space in language games using hedonic functions. *Fifth International Conference on Computational Creativity*.

Zwaan, R. A., Stanfield, R. A., and Yaxley, R. H. (2002). Language comprehenders mentally represent the shapes of objects. *Psychological science*, 13(2):168–171.

Appendix A

Algorithms of Agents' Functions

A.1 Primary Functions

A.1.1 Categorisation

In Algorithm 17, Method ANN, i.e., Artificial Neural Network can be Self-Organising Map (SOM), the network based on Adaptive Resonance Theory (ART), or Growing Neural Gas (GNG). BMU is the Abbreviation of Best Matching Unit, which is a neuron having the shortest distance with an input sample compared with other neurons. This neuron can be found and adjusted in SOM, ART and GNG or created in ART and GNG. Besides using ANN, general combination of prototypes and tolerances/variances can be used to make simple categorisations although they lack dynamic change and update compared with ANN.

This function can be used to categorise a whole topic, or categorise each dimension of the topic for the convenience of evolving compositional languages. For example, A colour with four dimensions, red, green, blue and alpha, can be categorised using four separated ANN. These categorisations of the four features can be used to generate new compositional colours represented by compositional utterances.

A.1.2 Parsing and Production

Parsing (see Algorithm 18) is the process of mapping utterances to relevant meanings/-categories, whereas production (see Algorithm 19) is the procedure of transforming meanings/categories to relevant utterances.

Algorithm 17 Categorising Topic

```
1: function CATEGORISE(topic, method, utterance)
2:     if method is ANN then                    ▷ ANN includes SOM, ART and GNG
3:         BMU ← make/findBMU(topic, ANN)
4:         ANN ← update(ANN, BMU)
5:         category ← updatedBMU
6:     else
7:         differences ← compare(topic, prototypes)
8:         candidates ← filter(prototypes, differences, tolerance)
9:         if candidates then
10:            category ← mostSimilarCandidate(candidates, topic)
11:        else
12:            prototypes ← append(prototypes, topic)
13:            category ← topic
14:        end if
15:    end if
16:    if utterance then
17:        add/update(relation(category, utterance))
18:    end if
19:    return category
20: end function
```

Algorithm 18 Parsing

```
1: function PARSING(utterance, type)
2:     meaning ← emptyList
3:     for utter in utterance do
4:         candidates ← emptyList
5:         for unitMeaning in storedMeanings do
6:             if utter ∈ storedUtterances_unitMeaning then
7:                 candidates ← append(candidates, unitMeaning)
8:             end if
9:         end for
10:        if candidates then
11:            if type =′ best then
12:                unitM ← weightedChoice(candidates, utter)
13:            else if type =′ roulette then
14:                unitM ← weightedRandomChoice(candidates, utter)
15:            else
16:                unitM ← randomChoice(candidates, utter)
17:            end if
18:        else
19:            unitM ← randomChoice(storedMeanings)
20:        end if
21:        meaning ← append(meaning, unitM)
22:    end for
23:    return meaning
24: end function
```

Algorithm 19 Production

```
1: function PRODUCTION(meaning, type)
2:     for unitMeaning in storedMeanings do
3:         if member(unitMeaning, meaning) then
4:             U ← storedUtterances_unitMeaning
5:             if type =′ best then
6:                 unitUtterance ← weightedChoice(U)
7:             else if type =′ roulette then
8:                 unitUtterance ← weightedRandomChoice(U)
9:             else
10:                unitUtterance ← randomChoice(U)
11:            end if
12:            meaning ← replace(unitMeaning, unitUtterance, meaning)
13:        end if
14:    end for
15:    utterance ← meaning
16:    return utterance
17: end function
```

A.1.3 Tracing Forward/Backward

The function, Tracing Forward (see Algorithm 20, 22, 23 and 24), is used to trace a rule sequence, which includes ≥ 1 rule(s), to a sequence of terminal rules or sub-rules using Context Free Grammar (CFG). For example, given $ruleChain = [compoSize]$, $rules = \{0 : \{rule : [compoSize],\ expan : [(0.21, very, size), (0.01, size, very), (0.19, size, size)]\},\ 1 : \{rule : [size],\ expan : [(0.21, small), (0.11, medium), (0.01,\ large)]\}\}$, $[compoSize]$ could be traced forward to $[very\ small]$. Sub-rules rather than terminal rules may be generated when certain *step* is given to limit the number of recursions.

On the contrary, Tracing Backward (see Algorithm 21) is utilised to trace a sequence of expansions, which could be terminal rules or sub-rules to a sequence of parent rules or a root rule affected by the argument *step*.

A.1.4 Learning/Unlearning Association Rules

Association rules include the weighted associations between meanings/categories and utterances, and the weighted relations between rules and their expansions. These rules can be learned/strengthened by increasing their weights using Function 25, 27 or unlearned/weakened by decreasing their weights using Function 26, 28.

A.1.5 Others

Some other primary algorithms include the function of weighted random choice (see Algorithm 29), e.g. weightedRandomChoice([(0.2, 2, b), (0.3, 3, c), (0.1, 1, a)]) → (0.3,

Algorithm 20 Tracing Forward

```
1: function traceForward(ruleChain, rules, activeRules, familiar?, step, type)
2:     ruleNodes ← emptyList
3:     for k in hashKeys(rules) do
4:         ruleNodes ← append(ruleNodes, [k, rules[k]['rule]])        ▷ e.g. [0,'size]
5:     end for
6:     ruleChain0 ← copy(ruleChain)
7:     ruleNodes ← sort(ruleNodes, 'listLength >)
8:     rrra ← replaceRuleChain(ruleChain, ruleNodes, rules, activeRules, familiar?, type)
9:     ruleChain ← rrra[0]
10:    ruleNodes ← rrra[1]
11:    rules ← rrra[2]
12:    activeRules ← rrra[3]
13:    subRule? ← False
14:    for rule in ruleNodes do
15:        if isSubList(rule[1], ruleChain) then        ▷ ('size, ['very, 'size])
16:            subRule? ← True
17:            break
18:        end if
19:    end for
20:    if ((ruleChain ≠ ruleChain0) ∨ subRule?) ∧ ((step = False) ∨ (step > 0)) then
21:        if step then
22:            step ← step − 1
23:        end if
24:        ruleChain ← traceForward(ruleChain, rules, activeRules, familiar?, step, type)
25:    end if
26:    return ruleChain
27: end function
```

3, c), and that of simplifying a list (see Algorithm 30), e.g. simplifyList([((('down)), ('right))]) → ['down, 'right], .

A.2 Functions for Playing Guessing Games

A.2.1 Selecting Topic

The function of selecting topic (see Algorithm 31, 32, 33) is mainly used by a speaker-agent to select a topic from context in different ways such as randomly selecting a topic, selecting the most common or different topic, or selecting the most confident or unconfident topic based on the stored meanings or instances collected from previous guessing games.

Algorithm 21 Tracing Backward

```
1: function traceBackward(expanChain, rules, activeRules, step)
2:     expanNodes ← emptyList
3:     expanChain0 ← copy(expanChain)                          ▷ for if-recursion
4:     for k in hashKeys(rules) do
5:         for e in rules[k]['expan] do
6:             expanNodes ← append(expanNodes, [k, e])          ▷ e.g. [0,[0.01,'small]]
7:         end for
8:     end for
9:     expanNodes ← sort(expanNodes, 'listLen >)  ▷ to replace longer expan firstly
10:    for expan in expanNodes do
11:        if isSubList(expan[1][1 :], expanChain) then
12:            k ← expan[0]
13:            expanChain ← replace(expan[1][1 :], rules[k]['rule], expanChain)   ▷ Key Line
14:            activeRules ← append(activeRules, [k, expan[1][1 :]])
15:        end if
16:    end for
17:    subExpan? ← False
18:    for expan in expanNodes do
19:        if isSubList(expan[1][1 :], expanChain) then
20:            subExpan? ← True
21:            break
22:        end if
23:    end for
24:    if ((expanChain ≠ expanChain0) ∨ subExpan?) ∧ ((step = False) ∨ (step > 0)) then
25:        if step then
26:            step ← step − 1
27:        end if
28:        expanChain ← traceBackward(expanChain, rules, activeRules, step)
29:    end if
30:    return expanChain
31: end function
```

A.2.2 Mapping Utterance to Topic

The procedure of mapping utterance to topic (see Algorithm 34) is implemented by a speaker-agent from categorising topic (see Algorithm 17) to making utterance via Function, *production*(*category*) (see Algorithm 19). Categories are related with meanings. If topic has only one dimension, only one unit-meaning needs to be used. Some experiments may use topics directly without categorising them, e.g., the experiment of exploring incongruity (see Sec. 5.3).

Algorithm 22 Replacing Rule Chain

```
1: function replaceRuleChain(ruleChain, ruleNodes, rules, activeRules, familiar?, type)
2:     for rule in ruleNodes do
3:         while isSubList(rule[1], ruleChain) do
4:             k ← rule[0]
5:             expans ← copy(rules[k]['expansion])
6:             expan ← getExpan(expans, familiar?, type)          ▷ expan=[weight,
   sequence, optional:maxDepth]
7:             maxDepth ← expan[−1]      ▷ maxDepth=[depth, threshold] or the last
   item of sequence
8:             switch ← True
9:             while switch do
10:                if isList?(maxDepth) then
11:                    if maxDepth[0] < maxDepth[1] then              ▷ update depth
12:                        iExpan ← position(expan, rules[k]['expansion])
13:                        expans ← copy(rules[k]['expansion])
14:                        maxDepth[0] ← maxDepth[0] + 1
15:                        expan[−1] ← maxDepth
16:                        expans[iExpan] ← expan
17:                        rules[k]['expansion] ← expans
18:                        expan ← expan[1 : −1]
19:                        switch ← False
20:                    else                                          ▷ get new expansion
21:                        expan ← getExpan(expans, familiar?, type)
22:                        maxDepth ← expan[−1]
23:                    end if
24:                else
25:                    expan ← expan[1 :]
26:                    switch ← False
27:                end if
28:            end while
29:            if member({, expan) then
30:                expan ← evalExpan(expan)
31:            end if
32:            if member({, rule) then
33:                evalExpan(rule)
34:            end if
35:            ruleChain ← replace(rule[1], expan, ruleChain)  ▷ rule[1] is rule name
36:            activeRules ← append(activeRules, [rule[0], expan])      ▷ rule[0] is rule
   key
37:        end while
38:    end for
39:    return [ruleChain, ruleNodes, rules, activeRules]
40: end function
```

Algorithm 23 Getting Expansion

```
1: function getExpan(expans, familiar?, type)
2:     if member(nil, map(x → x[0], expans)) then        ▷ nil means no competition
3:         type ← random
4:     end if
5:     if type = best then
6:         if familiar? then
7:             expan ← first(sort(expans, 'list >))
8:         else
9:             expan ← first(sort(expans, 'list <))
10:        end if
11:    else if type = roulette then
12:        if familiar? then
13:            expan ← weightedRandomChoice(expans, reverse? = False)
14:        else
15:            expan ← weightedRandomChoice(expans, reverse? = True)
16:        end if
17:    else
18:        expan ← randomChoice(expans)
19:    end if
20:    return expan
21: end function
```

Algorithm 24 Evaluating Expansion

```
1: function evalExpan(expan)                                          ▷ e.g.
   expan ←' (direction? { − 2 1 } obj0 { first [ 1 2 3 ] } obj1)
2:     funStarts ← emptyList
3:     funEnds ← emptyList
4:     for i = 1 → length(expan) do
5:         if expan[i] = { then
6:             funStarts ← append(funStarts, i)
7:         else if expan[i] =} then
8:             funEnds ← append(funEnds, i)
9:         end if
10:    end for
11:    for j = 1 → length(funStarts) do
12:        expression ← expan[(funStarts[j] + 1) : funEnds[j]]
13:        expan ← replace(expression, eval(expression), expan)
14:    end for
15:    expan ← removeIf(x → (x = {) ∨ (x =}), expan)
16:    return expan                          ▷ e.g. expan ← (direction? 1 obj0 1 obj1)
17: end function
```

Algorithm 25 Learning

```
1: function LEARN(A_mu, A_re, rate)  ▷ A_mu: associations of meaning and utterance, A_re: associations of rule and expansion
2:     weights_mu ← emptyList
3:     weights_re ← emptyList
4:     if A_mu then
5:         for a in A_mu do
6:             w ← strengthen(weight_a, rate)
7:             weights_mu ← append(weights_mu, w)
8:         end for
9:     end if
10:    if A_re then
11:        for a in A_re do
12:            w ← strengthen(weight_a, rate)
13:            weights_re ← append(weights_re, w)
14:        end for
15:    end if
16:    return [weights_mu, weights_re]
17: end function
```

Algorithm 26 Unlearning

```
1: function UNLEARN(A_mu, A_re, rate)  ▷ A_mu: associations of meaning and utterance, A_re: associations of rule and expansion
2:     weights_mu ← emptyList
3:     weights_re ← emptyList
4:     if A_mu then
5:         for a in A_mu do
6:             w ← weaken(weight_a, rate)
7:             weights_mu ← append(weights_mu, w)
8:         end for
9:     end if
10:    if A_re then
11:        for a in A_re do
12:            w ← weaken(weight_a, rate)
13:            weights_re ← append(weights_re, w)
14:        end for
15:    end if
16:    return [weights_mu, weights_re]
17: end function
```

Algorithm 27 Strengthen

```
1: function STRENGTHEN(w, r)  ▷ w: weight, r: learning rate
2:     return r + (1 − r) × w
3: end function
```

Algorithm 28 Weaken

```
1: function WEAKEN(w, r)  ▷ w: weight, r: learning rate
2:     return (1 − r) × w
3: end function
```

Algorithm 29 Weighted Random Choice

```
1: function WEIGHTEDRANDOMCHOICE(weightedItems, reverse?)
2:     if reverse? then
3:         weightedItems ← map(x → [1/x[0], x[1 :]], weightedItems)
4:     end if
5:     max ← sum(map(x → x[0], weightedItems))    ▷ x[0] is weight, e.g. x = [0.21, 'very, 'small]
6:     pick ← random(max)
7:     current ← 0
8:     for x in weightedItems do
9:         current ← current + x[0]
10:        if current > pick then
11:            if reverse? then
12:                result ← [1/x[0], x[1 :]]
13:            else
14:                result ← x
15:            end if
16:            Break
17:        end if
18:    end for
19:    return result
20: end function
```

Algorithm 30 simplifying A List

```
1: function SIMPLIFYLIST(lst)
2:     if length(lst) = 1 then
3:         lst ← lst[0]
4:         if isList?(lst) then
5:             lst ← simplifyList(lst)
6:         end if
7:     else
8:         lst1 ← emptyList
9:         for item in lst do
10:            if isList?(item) then
11:                lst1 ← append(lst1, simplifyList(item))
12:            else
13:                lst1 ← append(lst1, item)
14:            end if
15:        end for
16:        lst ← lst1
17:    end if
18:    return lst
19: end function
```

Algorithm 31 Selecting Topic

```
1: function selectTopic(context, type)
2:     if type =′ random then
3:         topic ← randomChoice(context)
4:     else if type =′ common then
5:         differences ← differEachOther(context)
6:         iCommon ← position(min(differences), differences)
7:         topic ← context_iCommon
8:     else if type =′ different then
9:         differences ← differEachOther(context)
10:        iDifferent ← position(max(differences), differences)
11:        topic ← context_iDifferent
12:    else if type =′ confident then
13:        confidences ← confidenceContext(context)
14:        iConfident ← position(max(confidences), confidences)
15:        topic ← context_iConfident
16:    else if type =′ unconfident then
17:        confidences ← confidenceContext(context)
18:        iUnconfident ← position(min(confidences), confidences)
19:        topic ← context_iUnconfident
20:    else
21:        topic ← randomChoice(context)
22:    end if
23:    return topic
24: end function
```

Algorithm 32 Differences Between Each Other

```
1: function differEachOther(context)
2:     differences ← emptyList
3:     for i = 1 → length(context) do
4:         d ← 0
5:         for j = 1 → length(context) do
6:             d ← d + difference(context_i, context_j)
7:         end for
8:         differences ← append(differences, d)
9:     end for
10:    return differences
11: end function
```

Algorithm 33 Measuring Confidence of Context

```
1: function CONFIDENCECONTEXT(context)
2:     confidences ← emptyList
3:     for topic in context do
4:         c ← 0
5:         for unitMeaning in topic do
6:             if member(unitMeaning, storedMeanings) then
7:                 c ← c + weight(unitMeaning, storedMeanings)
8:             end if
9:         end for
10:        confidences ← append(confidences, c)
11:    end for
12:    return confidences
13: end function
```

Algorithm 34 Getting Utterance

```
1: function GETUTTER(topic)
2:     category ← categorise(topic)
3:     utterance ← production(category)
4:     return utterance
5: end function
```

A.2.3 Guessing Topic

The algorithm of guessing topic according to the provided utterance and context (see Algorithm 35) is processed by a listener-agent with two methods. The first method is parsing the utterance to relevant category, and finding samples matching the category from context. If ≥ 1 sample(s) match the category, one of them is selected as the guessed topic via a choice technique such as random-choice, interesting-choice (see Algorithm 6), weighted-choice or weighted-random-choice (see Algorithm 29). Otherwise, another method is adopted. It is getting the utterances of all samples in context via Function *getUtter*(*topic*) (see Algorithm 34), comparing the generated utterances with the provided utterance to find the most similar one, then selecting the related sample as the guessed topic.

A.3 Functions for Playing Generation Games

A.3.1 Generating Utterance

The function of generating a compositional utterance (see Algorithm 36) is realised with two steps. First, a root rule or a sequence of parent rules are traced forward (see Algorithm 20) to a sequence of terminal rules, i.e., a compositional prototype. Then the prototype is mapped to a compositional utterance via Function *production*(*compoPrototype*)

Algorithm 35 Getting Topic

```
1: function getTopic(utterance, context, type)
2:     category ← parsing(utterance)
3:     candidates ← emptyList
4:     for sample in context do
5:         if sample ∈ category then
6:             candidates ← append(candidates, sample)
7:         end if
8:     end for
9:     if candidates then
10:        if type =′ interesting then
11:            design ← interestingChoice(candidates, category)
12:        else if type =′ best then
13:            topic ← weightedChoice(candidates, category)
14:        else if type =′ roulette then
15:            topic ← weightedRandomChoice(candidates, category)
16:        else
17:            topic ← randomChoice(candidates)
18:        end if
19:    else
20:        distances ← emptyList
21:        for sample in context do
22:            utter ← getUtterance(sample)
23:            d ← LevenshteinDistance(utter, utterance)
24:            distances ← append(distances, d)
25:        end for
26:        iTopic ← position(min(distances), distances)
27:        topic ← context_iTopic
28:    end if
29:    return topic
30: end function
```

(see Algorithm 19). The function, Generating Utterance, is used by client-agents to generate design briefs, i.e., requirements.

Algorithm 36 Generating Utterance

```
1: function generateUtterance(rule)
2:     compoPrototype ← traceForward(rule)
3:     utterance ← production(compoPrototype)
4:     return utterance
5: end function
```

A.3.2 Generating Design

The procedure of generating design (see Algorithm 37) is implemented by a designer-agent in three steps. At first, an utterance, i.e., the client-agent's requirement, is parsed to a relevant meaning. Secondly, the meaning is categorised to a category. Then a

design is generated to match the category with a generation type such as interesting design (see Algorithm 6) or best matched design. If the argument, CFG?, is *True*, the meaning parsed in the first step is traced to a new meaning before running the second step. New compositional meanings could be generated by processing association rules via the cooperation of tracing forward (see Algorithm 20) and tracing backward (see Algorithm 21). The rules are generated using Context Free Grammar (CFG).

Algorithm 37 Generating Design

```
1: function GENERATEDESIGN(utterance, type, CFG?)
2:     meaning ← parsing(utterance)
3:     if CFG? then
4:         rule ← traceBackward(meaning)
5:         meaning ← traceForward(rule)                    ▷ new meaning
6:     end if          ▷ weighted-CFG can generate interesting meaning
7:     category ← categorise(meaning)
8:     if type =′ interesting then
9:         design ← interestingChoice(range_category)
10:    else if type =′ best then
11:        design ← weightedChoice(range_category)
12:    else if type =′ roulette then
13:        design ← weightedRandomChoice(range_category)
14:    else
15:        design ← randomChoice(range_category)
16:    end if
17:    return design
18: end function
```

A.3.3 Selecting Design

In the algorithm of selecting design (see Algorithm 38), two methods are used to select winning design(s). The first method is *interestingChoice*(*designs*) (see Algorithm 6) for the selection type, "interesting". The second method is removing the designs which are not in the acceptable range, then selecting a winning-design from the remaining designs using a selection method such as weighted-choice, weighted-random-choice or random-choice. Besides the two methods, some other methods may be used to satisfy client-agents' specific requirements such as incongruity (see Sec. 5.3) by using inverse-weighted-random-choice and elaboration (see Sec. 5.4) by matching certain criteria settings.

A.4 Conclusion

The algorithms of agents' functions consist of basic functions, the functions for playing guessing games and the functions for playing generation games. They can be used to

Algorithm 38 Selecting Design

```
1: function SELECTDESIGN(designs, prototype, type)
2:     if type =′ interesting then
3:         winningDesign ← interestingChoice(designs, prototype)
4:     else
5:         candidates ← emptyList
6:         for design in designs do
7:             distan ← distance(design, prototype)
8:             if distan < tolerance then
9:                 if distan = 0 then
10:                    distan ← 0.000001
11:                end if
12:                candidates ← append(candidates, [1/distan, design])
13:            end if
14:        end for
15:        if candidates then
16:            if type =′ best then
17:                winningDesign ← weightedChoice(candidates)
18:            else if type =′ roulette then
19:                winningDesign ← weightedRandomChoice(candidates)
20:            else
21:                winningDesign ← randomChoice(candidates)
22:            end if
23:        else
24:            winningDesign ← False
25:        end if
26:    end if
27:    return winningDesign
28: end function
```

establish a computational model of design agents based on curious agents (Saunders, 2002) to evolve artificial languages for creative design at the sociocultural level. These algorithms may need to be adjusted or modified to match the requirements of different experiments. And some other specific functions may need to be added to address certain conditions of the experiments such as developing graph networks (see Appendix B) by adding edges (see Algorithm 41) connecting compositional objects in Experiment 5.4.

Appendix B

Algorithms of Graph Networks

Algorithm 39 Making Graph

```
1: function makeGraph
2:     nodes ← makeHashTable()
3:     edges ← makeHashTable()
4:     el ← emptyList
5:     return {'nodes : nodes, 'edges : edges, 'activeNodes : el, 'traceStart : el, 'traceEnd : el, 'path : el}
6: end function
```

Algorithm 40 Adding A Node

```
1: function addNode(graph, node, props, utter?)
2:     if ¬node then
3:         node ← count(graph['nodes])
4:     end if
5:     if utter? then
6:         if isList?(utter?) ∨ isString?(utter?) then
7:             props['utter] ← [[0.01, utter?]]
8:         else
9:             props['utter] ← [[0.01, randomUtterance()]]
10:        end if
11:    end if
12:    graph['nodes][node] ← props
13:    return graph
14: end function
```

Algorithm 41 Adding An Edge

```
1: function addEdge(graph, edge, nodes, props, utter?, weight)
2:     if nodes then
3:         props['nodes] ← nodes
4:         if isList?(nodes[0]) then
5:             for node in flatten(nodes) do
6:                 if node ∉ hashKeys(graph['nodes]) then
7:                     graph ← addNode(graph, node, {}, utter?)
8:                 end if
9:             end for
10:        else if isList?(nodes[1]) then
11:            for node in flatten(nodes[1 :]) do
12:                if node ∉ hashKeys(graph['nodes]) then
13:                    graph ← addNode(graph, node, {}, utter?)
14:                end if
15:            end for
16:            if nodes[0] ∉ hashKeys(graph['nodes]) then
17:                graph ← addNode(graph, nodes[0], {}, utter?)
18:            end if
19:        else
20:            if nodes[0] ∉ hashKeys(graph['nodes]) then
21:                graph ← addNode(graph, nodes[0], {}, utter?)
22:            end if
23:            if nodes[1] ∉ hashKeys(graph['nodes]) then
24:                graph ← addNode(graph, nodes[1], {}, utter?)
25:            end if
26:        end if
27:    end if
28:    if weight then
29:        props['weight] ← weight
30:    end if
31:    switch ← True
32:    for e in hashValues(graph['edges]) do
33:        if e = props then
34:            switch ← False
35:            Break
36:        end if
37:    end for
38:    if switch then
39:        if ¬edge then
40:            edge ← count(graph['edges])
41:        end if
42:        graph['edges][edge] ← props
43:    end if
44:    return graph
45: end function
```

Algorithm 42 Tracing Forward Nodes

```
1: function traceForwardNodes(graph, nodes, choice, start, nonterminal?, level)
2:     start2 ← start                                  ▷ default setting: start = True
3:     if start then
4:         graph['traceEnd] ← emptyList
5:         cleanDepth(graph)
6:         start ← False
7:     end if
8:     for node in nodes do
9:         runNode(graph, node)
10:    end for
11:    trace ← emptyList
12:    for node in nodes do
13:        newNodes ← getOutNodes(graph, node, choice)
14:        if newNodes ∧ ((¬level) ∨ (level > 0)) then
15:            if nonterminal? then
16:                trace ← append(trace, runNode(graph, node))
17:            end if
18:            if level then
19:                level ← level − 1
20:            end if
21:            trace2 ← traceForwardNodes(graph, newNodes, choice, start, nonterminal?, level)
22:            if trace2 then
23:                trace ← append(trace, trace2)
24:            end if
25:        else
26:            trace ← append(trace, runNode(graph, node))
27:        end if
28:    end for
29:    trace ← simplifyList(trace)
30:    if start2 ∧ trace then
31:        graph['traceEnd] ← trace
32:    end if
33:    return trace
34: end function
```

Algorithm 43 Running A Node

```
function runNode(graph, node)
    if 'function ∈ hashKeys(graph['nodes][node]) then
        if 'input ∈ hashKeys(graph['nodes][node]) then
            newInput ← getInput(graph, node)
            if newInput ∧ ('noData ∉ newInput) then
                graph['nodes][node]['input] ← newInput
            end if
        end if
        graph['nodes][node]['output] ← eval(graph['nodes][node]['function])
        result ← graph['nodes][node]['output]
    else
        result ← False
    end if
    return result
end function
```

Algorithm 44 Getting Input

```
function getInput(graph, node)
    input ← emptyList
    nodesIn ← getInNodes(graph, node)
    if nodesIn then
        for n in nodesIn do
            if 'output ∈ hashKeys(graph['nodes][n]) then
                input ← append(input, graph['nodes][n]['output])
            else
                input ← append(input,' noData)
            end if
        end for
    end if
    return input
end function
```

Algorithm 45 Getting In-Nodes

```
function getInNodes(graph, node, choice)
    nodesIn ← emptyList
    edgesIn ← getInEdges(graph, node, choice)
    if edgesIn then
        for edge in edgesIn do
            nodesIn ← append(nodesIn, graph['edges][edge]['nodes][0])
        end for
    end if
    return flatten(nodesIn)
end function
```

Algorithm 46 Getting Out-Nodes

```
1: function getOutNodes(graph, node, choice)
2:     nodesOut ← emptyList
3:     edgesOut ← getOutEdges(graph, node, choice)
4:     if edgesOut then
5:         for edge in edgesOut do
6:             nodesOut ← append(nodesOut, graph['edges][edge]['nodes][1])
7:         end for
8:     end if
9:     return flatten(nodesOut)
10: end function
```

Algorithm 47 Getting In-Edges

```
1: function getInEdges(graph, node, choice)
2:     edgesIn ← emptyList
3:     for k in hashKeys(graph['edges]) do
4:         if node ∈ graph['edges][k]['nodes][1] then
5:             edgesIn ← append(edgesIn, [graph['edges][k]['weight], k])
6:         end if
7:     end for
8:     if edgesIn ∧ (choice =' weightedRandomChoice) then
9:         edgeIn ← second(weightedRandomChoice(edgesIn))
10:        if ('maxDepth ∈ hashKeys(graph['edges][edgeIn])) ∧ graph['edges][edgeIn]['maxDepth] then
11:            if graph['edges][edgeIn]['maxDepth] > graph['edges][edgeIn]['depth] then
12:                graph['edges][edgeIn]['depth] ← graph['edges][edgeIn]['depth] + 1
13:            else
14:                go to 9
15:            end if
16:        end if
17:        edgesIn ← [edgeIn]
18:    else if edgesIn then
19:        edgesIn ← map(x → x[1], edgesIn)
20:    end if
21:    return edgesIn
22: end function
```

Algorithm 48 Getting Out-Edges

```
1: function getOutEdges(graph, node, choice)
2:     edgesOut ← emptyList
3:     for k in hashKeys(graph['edges']) do
4:         if node ∈ graph['edges'][k]['nodes'][0] then
5:             edgesOut ← append(edgesOut, [graph['edges'][k]['weight], k])
6:         end if
7:     end for
8:     if edgesOut ∧ (choice =' weightedRandomChoice) then
9:         edgeOut ← second(weightedRandomChoice(edgesOut))
10:        if ('maxDepth ∈ hashKeys(graph['edges'][edgeOut])) ∧ graph['edges'][edgeOut]['maxDepth] then
11:            if graph['edges][edgeOut]['maxDepth] > graph['edges][edgeOut]['depth] then
12:                graph['edges][edgeOut]['depth] ← graph['edges][edgeOut]['depth] + 1
13:            else
14:                go to 9
15:            end if
16:        end if
17:        edgesOut ← [edgeOut]
18:    else if edgesOut then
19:        edgesOut ← map(x → x[1], edgesOut)
20:    end if
21:    return edgesOut
22: end function
```

Algorithm 49 Cleaning Depth

```
1: function cleanDepth(graph)
2:     for edge in hashKeys(graph['edges]) do
3:         if 'depth ∈ hashKeys(graph['edges][edge]) then
4:             graph['edges][edge]['depth] ← 0
5:         end if
6:     end for
7:     return graph
8: end function
```