Project Number: CS DCB-1002

A Computational Model of Surprise

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCERSTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Daniel L. Spitz

Date:

4/28/11

Approved:

Professor David C. Brown, Advisor

Abstract

The computation of surprise is one factor necessary for the computational detection of creativity in product design. An original computational model of surprise is proposed. A demonstration program was written to demonstrate the model's validity. An experiment on human subjects was performed to test the program's accuracy. The results of the experiment are analyzed, the model is assessed, and conclusions are made. This work is a promising start toward the goal of a practical computational surprise detector.

Acknowledgements

Thank you to my project advisor, Professor Brown, for all of your time, your feedback, your guidance on project direction, and the thoroughly enjoyable discussions we had on a weekly basis. Thanks to Professor Janice Gobert for her help with experiment design and for getting me access to some experimental subjects. Thanks to Dr. Alex Smith for the use of your lab to conduct the experiments. Thanks to my friends and family for all your support.

Table of Contents

Table of Figures

1 Introduction

1.1 Background

Computational creativity is a broad concept. It is potentially applicable in product evaluation and in the creation of a computational design aid. The task of computationally rating a product's creativity involves the computation of a number of more concrete facets of the product. One of these facets is surprise. Knowing a product's surprisingness is potentially useful for understanding how best to market the product, or how best to alter its design to be more successful.

This project focused on the computation of surprise. Prior work on the computation of surprise has been done. There has been discourse on surprise's role in creativity and customer product assessment. There has also been separate work done on a cognitive-psychological model of surprise, and the simulation of surprise in artificial intelligence systems.

1.2 Motivation

None of the computational models of surprise from the literature were aimed at making concrete, testable predictions about the surprisingness of real products. Likewise, none of the existing work on surprise as it pertains to creativity or product design is concrete enough to build a testable computational model. This project aimed to bridge the gap between surprise in the real-world area of creative product assessment, and the psychological and AI models of surprise.

1.3 General Project Description

In this project an original computational model of surprise was developed. The model was developed with the goal of leading to a concrete implementation of a surprise detector. A demonstration program was developed based on the model, with the goal of generating testable predictions of the surprisingness of products. An experiment was conducted on human subjects to assess the accuracy of the demonstration program's predictions, and the validity of the proposed model.

2 Literature Review

This section will review the work serving as the basis for the research described by this report. The body of work discussed here comes from a diverse set of research fields. The following topics will be discussed:

- two models for assessing creativity in product design, with a discussion about their conceptual similarities and differences;
- the phenomenon of surprise in humans, as studied from a psychological perspective;
- the modeling of surprise for AI applications, and two distinct lines of research concerning the implementation and study of surprise in an AI system;
- the role of the surprise reaction in user product assessment, and a general examination of test methodologies used in experiments that gauge user surprise reactions to products .

Discussion for each of these topics will include a brief summary of the work performed, and an explanation of the work's contribution to this report.

2.1 Assessing Design Creativity

Two models for assessing the creativity of a product are described. Besemer and Treffinger (1981) developed the Creative Product Assessment Model (CPAM), which has been shown to produce accurate, practically useful assessments of product creativity. Maher provides another model, aimed in particular at the assessment of creativity by computational means (Maher, 2010).

2.1.1 Besemer's Creative Product Analysis Model

The CPAM (Besemer & Treffinger, 1981) is a model for assessing a product's creativity. It aims to integrate the many disparate criteria proposed for creative analysis into one consistent model. The CPAM is intended to allow us to "discuss, analyze and improve products" (Besemer, 2011). The concepts laid out in the CPAM provide the basis of the Creative Product Semantic Scale, a tool with a proven track record as an effective product assessment aid. The CPSS consists of a questionnaire about a product that users or designers fill out. Next a report is generated based on responses rating the product in the numerous facets of creativity put forth in the CPAM. The results have led to useful insight into improvement of a product's design.

The CPAM uses the three dimensions Novelty, Resolution and Style as its basis. Each of these dimensions is further broken down into multiple facets. Each dimension identifies an independent aspect of creativity.

Novelty is the dimension concerned with the newness of a product. Novelty may appear in a product in multiple ways, including a new feature or new use of an existing product. It is composed by the two facets Original and Surprising. Originality is the quality of being unusual or infrequently seen. Surprise is the quality of producing a reaction of shock or grabbing attention.

Resolution is the dimension concerned with a product's functionality and utility. The first its four facets is Logical. Here, Logical refers to how intuitive or straightforward the functioning of the product is. The facet Useful is a measure of how well a product does the job it is intended to do, and is especially applicable for home appliances or medical technology, in which proper function is highly important. Valuable is the third facet. Products may be valuable in the straightforward sense of monetary worth or by saving time. Value is also present in products addressing psychological needs, such as a feeling of safety or security. Finally, there is the facet Understandable, the degree to which a product is easy to learn to use.

The Style dimension is the way that a product is presented to the user, influencing the user's opinions and perceptions about it. Well-Crafted is the first of three Style facets. This facet measures the extent to which a product conveys that time and effort were invested in its making. The facet Organic is the quality of appearing natural, in which "the parts of the object fit together in a logical, economic, and flowing way". Finally, the facet Elegant refers to the product's being refined and simplified, perhaps in its design or in the way it solves a problem.

2.1.2 Maher's Common Evaluation Metric

Maher presents "a common metric for evaluating creativity that is based on three essential criteria for creativity" (Maher, 2010). The criteria used are novelty, unexpectedness, and value, and are defined in order that they may apply commonly to all domains of creative products. Maher argues that the combination of these three measures is "necessary and sufficient criteria for creativity." Each measure is defined in terms of how it might be formally computed, and concrete strategies for its computation are discussed. Thus, the metric is a step toward computational evaluation of creativity. Since the metric is intended to apply to any domain of products, it is too general to be implemented as-is. Instead, the definition of each measure must be interpreted to produce a computational metric that may apply to a particular domain. The metric is "a formalism that can be adapted and applied to generate different algorithms or to guide human judgment of creativity."

Maher describes novelty as “a measure of how different the artifact is from known artifacts in its class.” The computation of novelty is a measurement of one product’s distance from other products in the space. Methods for measuring distance in the multi-dimensional space of products are discussed, in particular drawing on clustering and self-organizing maps.

Unexpectedness, also referred to as surprise, is present “when we recognize a pattern in recent artifacts, and the potentially creative artifact does not follow the expected next artifact in the pattern.” Maher distinguishes surprise from novelty, highlighting the computation of surprise as being “achieved by setting up expectations over a period of time or over a sequence of designs.” Essential to the computation of surprise is the facility to identify patterns occurring over the sequence of events. Products of a certain type will tend to have many identifiable similarities. Certain structural or functional aspects of the products may be identical or have only slight variations. A pattern is recognized when such similarities are identified across products. Surprise occurs when a product does not follow a pattern that has been identified.

Value is computed as the “weighted sum of the performance variables of the artifact.” The computation of each performance variable is highly dependent on the domain of the product. A performance variable may encapsulate a product’s level of social acceptance, or its compliance with design requirements inherent in the product space. Additionally, the computation of performance variables must be subject to change as creative products are encountered. As new products are encountered, they may introduce new ways of solving a problem, or recast the problem in a different light. Our valuation of other products may then change in response to such a novel product. One proposed method for computing value is with “a coevolutionary algorithm in which the fitness function changes in response to the current population of potential.”

2.1.3 Comparison of Creativity Assessment Models

The two models presented are targeted at solving similar but distinct problems in the area of creativity. While they do have a good bit of overlap, the differences in overall goal lead to discrepancies in how the different aspects of creativity are categorized. The CPAM is a model trying to help people understand and talk about the creative aspects of products. On the other hand, Maher’s model aims to highlight how these aspects may be understood and rated computationally. Besemer’s model is more likely to give people an intuitional understanding of creativity that may help them think and communicate about it. Maher’s model makes progress in the direction of formalizing such concepts to the point that they might be computationally measured.

Both models recognize the newness of a product as important and aspects of creativity. The CPAM has the dimension Novelty, containing the Originality and Surprising facets. Maher's model uses the terms Novelty and Surprise to refer to two different aspects of creativity, distinguished by different formal methods for their computation. Maher captures much of the rest of the facets present in the CPAM in one Value category. While the CPAM distinguishes Resolution from Style, and highlights numerous facets therein, Maher's model labels them as being computed by the same single formal method.

The models differ in the manner by which they were developed. The CPAM was developed by analyzing the ways people tended to judge products when prompted. The CPAM has its dimensions separated as it does to best capture the way people tended to categorize products. Maher's model, with its goal of guiding progress in computational creativity, separates creativity by the type of algorithm necessary to compute a particular facet.

2.2 Psychological Model of Surprise

A psychological model of surprise as a process occurring in humans has been developed and verified (Meyer, Reisenzein, & Schützwohl, 1997). The model describes a series of four cognitive activities that occur during surprise. It uses the concept of the schema to represent a person's current set of expectations about the situation they are in.

The series of cognitive activities is as follows:

1. An event is appraised as "exceeding some threshold value of schema-discrepancy (or unexpectedness)." That is, upon witnessing some event, one or more of the person's current expectations is violated to a significant degree.
2. As a consequence, the person experiences a feeling of surprise. At the same time, the person will refocus her attention on the surprising event, interrupting her previous task or thought.
3. Motivated by the feeling of surprise, the person analyzes the surprising event, consciously verifying its unexpectedness, assessing the cause of the event, and deciding whether some response or change in plans is necessary.
4. If a response is necessary, the person begins responding.

2.2.1 Discussion

This work is relevant to anyone aiming to model surprise computationally. Any complete model of surprise will implement each of the four parts of the surprise process as detailed in the work above. The work is not specified in a computationally rigorous way; it is not clear what exactly a schema may represent, nor how an event is assessed as unexpected. It therefore serves as a credible and useful basis for a computational model of surprise.

2.3 AI Models of Surprise

Work is presented that makes progress toward a computational definition of surprise. The implementation for an Artificially Intelligent system supporting a form of surprise is discussed.

2.3.1 Three Causes of Surprise

Ortony & Partridge (1987) detail a model outlining three ways in which expectations may fail, and thus activate surprise. In terms of the psychological model discussed above, this model elucidates the first part of the surprise process: namely, the assessment of an event as unexpected.

The model is presented in terms of a knowledge-base whose knowledge is both episodic and semantic. The knowledge-base has a current set of explicitly-represented propositions. There is a mechanism by which new propositions may be deduced via reasoning from existing propositions. A set of deducible propositions may be reached by this reasoning mechanism from the current propositions. There also exists a subset of these deducible propositions, referred to as the practically deducible propositions. A proposition is said to be practically deducible "to the extent that it does not require many and complex inferences." That is, the proposition might be quickly and easily deduced from the current propositions.

Using these definitions, three types of expectation failure or cause for surprise are defined:

- **Active expectation failure, or prediction failure.** This refers to situations in which an input proposition to the knowledge-base conflicts directly with one or more of the current, explicit propositions. It is likened to situations in which a person has made a conscious prediction about some outcome, and then discovers the prediction was wrong. Active expectation failure may be detected by checking the input proposition against each of the current propositions.

- **Passive expectation failure, or assumption failure.** This refers to the violation of one or more practically deducible propositions upon receiving the next input proposition. For a person, this might be likened to suddenly realizing that a person's assumptions have led her astray, and that she would have expected that which has just happened, not to have. In order to detect passive expectation failure there must be a search mechanism which attempts to construct a practically deducible proposition contradicting the current input proposition. If such a thing is found, passive expectation failure is occurring.
- **Unanticipated incongruity**. This refers to those (quite numerous) situations in which the input proposition expresses information unrelated to any active expectations, and further having no practically deducible propositions relating to it. A person would never have considered the likelihood of this event, because it is so unrelated to her current expectations. In these cases, the likelihood of the input proposition must be computed by impractically deducible propositions, or ones requiring more attention and resources than were available at the time.

2.3.2 Surprise-based Agent Architecture

Macedo and Cardoso have taken the theoretical work of Meyer, Ortony & Partridge, and others, several steps further in the direction of a working computational model of surprise. They describe an implementation of an AI system in which agents experience and react to surprise (Macedo, Cardoso & Reisenzein, 2006).

Macedo and Cardoso's system consists of a number of utility-based agents within a simple virtual environment. Virtual buildings are located throughout the environment, and the agents move freely to examine buildings up close. Buildings have properties relating to their structure, such as roof shape and number of windows, and functions, such as post office and church. Agents feel surprise when they encounter properties in a building that were unexpected.

Agents may be surprised in three ways, drawing on the three types of surprise put forth by Ortony and Partridge (described in section 3.2).

Active Expectation Failure

Agents form active expectations when they see a building's structure in the distance but are not yet close enough to know its function. When this happens, the agent will generate a prediction about the building's function based on other buildings in its memory. If this prediction is violated when the agent reaches the building, it experiences surprise via active expectation failure.

Passive Expectation Failure

Because agents perceive all of a building's structural properties simultaneously, it gets no opportunity to form active expectations about them. However, the agent may still experience surprise upon encountering unexpected structure, by way of passive expectation failure. This is done by computing the conditional probability of a given property given the other properties.

Unanticipated Incongruity

When an agent encounters a property it has never seen before, it has no way of calculating its probability. In these cases, surprise is felt because it did not (and could not) expect that it would encounter the property. This is known as unanticipated incongruity, and the probability is calculated after the fact.

Because the buildings in this environment have multiple structural and functional components, a method was devised for calculating the total surprise intensity of a single object. Each component of an object each has its own degree of unexpectedness. It is computed by taking the conditional probability of the component, given the rest of the components of the object. In the case of the virtual buildings, the unexpectedness of a building's function as post-office would be determined by the probability of post-offices having this building's structural properties. The total surprise of an object is the mean unexpectedness of each of its components. According to a study by the authors, it has been demonstrated that this method is a close approximation of the way humans react to surprising objects.

2.3.3 Knowledge Representation

The topic of Knowledge Representation is relevant to this project for its application in representing physical structures.

2.4 Human Surprise Reactions to Products

Ludden has done work on the occurrence of surprise in reactions to products (Ludden, 2008). Her work focuses on the case of surprise that is elicited by visual-tactile incongruities in products. Such surprise is produced when an observer forms expectations about a product based on its appearance, and then one or more expectations are violated when the observer touches and handles the product. Ludden presents a model for the types of surprise that may occur in these situations, and demonstrates its validity using human experiments.

2.4.1 Visible Novelty and Hidden Novelty

Ludden divides visual-tactile surprisingness into two types: Visible Novelty and Hidden Novelty. Visible Novelty is present in products that are visibly distinct from known products. An observer will be uncertain about what to expect of the product's texture or weight because it does not look familiar. As a result, proposes Ludden, the observer forms tactile expectations with a low degree of certainty. When they are violated, surprise due to Visible Novelty is experienced. For instance, consider a vase with fabric covering. Because you do not know what material lies underneath the fabric, you have little confidence in your expectations about the vase's weight. You have seen similar fabric covering on couch cushions, and might therefore expect the vase to share other properties with them. But you are unconvinced, because, unlike a cushion, the vase is rigid, smooth and perfectly shaped. So you form only a tentative expectation that the vase will be light and plush like a cushion. When you pick up the vase and find that the fabric is actually covering a heavy, carved wooden frame, your "light and plush" expectation is violated and you feel surprise. Since the vase had such an unfamiliar texture, you formed your expectation without much confidence. The vase is thus said to have Visible Novelty, because the surprise comes from a feature that is known beforehand to be potentially surprising. Visible Novelty is indicated as producing normal-to-large surprise.

Hidden Novelty, on the other hand, appears in products with visual features that are strongly recognizable, but whose tactile properties are different from those of what is resembled. An observer is "tricked" into expecting one weight, texture or stiffness, and is surprised when the expectation is violated. For example, consider a vase covered with what appears to be porcelain finish. You have seen other vases with porcelain finish before, all of which were ceramic, heavy and rigid. You have also seen other types of objects with porcelain finish, such as plates and sculptures, with the same tactile properties. So you strongly expect that this vase will share those properties too. Upon lifting it up, however, you are surprised to find it far lighter weight than you had expected. The vase is not ceramic at all—it is actually made of a shiny, lightweight plastic material, dyed and molded to resemble porcelain. The features that proved surprising—weight and material—were not in question until the moment of surprise. Therefore, this vase is said to have Hidden Novelty. Unlike with Visible Novelty, the observer has a high degree of confidence in his expectation before it is violated. For this reason, Hidden Novelty always produces large surprise.

2.4.2 Experiment

An experiment was performed to test the validity of Ludden's model with humans. First, eighteen products were chosen for the experiment, consisting of three products in each of the following categories: vase, lamp, tablecloth, bench, cup and tile. In each category, one product had Visible Novelty, one had Hidden Novelty, and the last has no or few visual-tactile incongruities, a control. Test subjects were exposed to products under two conditions: first the "see" condition, and then the "see and feel" condition. The procedure for the experiment was as follows:

1. First, in the "see" condition, the subject is led into a setting in which the product would typically be found. The product is initially covered so that it is completely obscured from view. The subject is instructed to sit in a chair two meters away from the covered product.
2. The experimenter lifts the cover, revealing the product to the subject for five seconds before covering it again.
3. The subject takes a questionnaire concerning the product's familiarity, expectations about how the product will feel, and the certainty of these expectations.
4. After evaluating six products in the "see" condition, the "see and feel" condition begins for the other six products. The subject is again led into a typical setting for the product, with the product covered, and the subject is seated two meters from the covered the product.
5. The experimenter lifts the cover and allows two seconds to pass.
6. The subject is instructed to stand up and approach the product, touch it and lift it. The subject explores the product at her own pace.
7. When the subject puts down the product, the experimenter covers it again.
8. The subject takes a questionnaire concerning how surprised she was, how much confidence she had in her initial expectations, and the degree of difference between what she felt and what she had expected to feel.

2.4.3 Results

The results of the experiment suggest that the two hypothesized types of surprise do indeed exist. A correlation was observed between product familiarity and degree of confidence of expectations. Visible Novelty products had less familiarity and led to less confident expectations. Hidden Novelty products, together with control (non-novel) products, scored higher on familiarity and confidence. On surprisingness, Hidden Novelty scored highest, followed closely by Visible Novelty, with control products scoring low on surprise. The clear divisions in per-category results indicate that the hypothesized types of surprise are likely valid. However, there were some issues with the experiment. Three of the six products initially labeled Hidden Novelty produced responses indicating Visual Novelty (two products) or control (one product). The author suggests this is because it is difficult to create a situation in which the "trick" necessary for Hidden Novelty is properly delivered; the observer may not notice the misleading feature, or may not realize when her erroneous expectation has been disconfirmed.

2.4.4 Interpretation of Ludden's Types of Surprise

The experiment provides insufficient information for distinguishing active from passive expectation failure (Section 3.3.1). This is because the questionnaires administered primed test subjects with leading questions that encouraged them to think of each possible tactile-related expectation allowed by the experiment. This caused all observable expectations to become active before being measured. Thus, any potential correlation between Ludden's types of surprise and whether the expectation is active or passive may not be measured. Such a problem is difficult to avoid, and was not the focus of the experiment.

It seems likely that surprise by way of Visible Novelty differs from surprise by Hidden Novelty, with respect to the cause of surprise. For Hidden Novelty, the surprise may be seen as straightforward expectation failure, though it is unclear whether passive or active. An expectation is violated by new tactile information, and surprise is felt in rough proportion to the expectation's confidence. The same model does not appear to hold for Visible Novelty products. These products have high surprise intensity, but low expectation confidence. The intensity of surprise is not in proportion to the confidence of the expectation. It appears, therefore, that there is another variable affecting surprise intensity.

My hypothesis is that Visible Novelty products produce surprise by unanticipated incongruity rather than active or passive expectation failure. This would explain why Visible Novelty products have such high surprise intensity despite their low expectation confidence. In cases of unanticipated incongruity, an observer first realizes that an input event was not expected by active or passive expectation. Once this realization is made, the intensity of surprise is determined by considering the input event's degree of deviation from norms. This is an altogether different method than finding intensity in proportion to the violated expectation’s confidence. In fact, it appears to be a measure of the input event’s novelty.

3 Problem Statement

The aim of this project is to produce a computational model of surprise, and to demonstrate its accuracy with a demonstration program. The surprise model will be applied to the area of product evaluation, so that the demonstration program will evaluate the surprisingness of a product.

Current computational models of surprise are either too simple or too general to be readily applied to the accurate simulation of surprise as it occurs in humans. This report attempts to refine existing surprise models with the goal of producing results that are directly testable against human surprise reactions.

The model will be evaluated by way of an experiment with human subjects. The experiment will measure human surprise reactions in situations that can be predicted by the demonstration program. The accuracy of the demonstration program, and by implication, the surprise model, will evaluated by comparing the demonstration program's predictions to human reactions.

4 A Hypothesized Computational Model of Surprise

A hypothesized computational model of surprise is proposed. Following the high-level description of the model is a set of design requirements that further specify the model.

4.1 Model

It is hypothesized that surprise is caused by either active or passive expectation failure in response to some input event. Active expectations are conscious expectations formed just before the surprising event, while passive expectations are formed after the surprising event has occurred. This is based on Ortony and Partridge's model of the causes of surprise (Section 2.3.1).

Active expectations are formed based on knowledge about the current situation. Active expectations are always about information which is currently unknown, but expected to become known soon. Passive expectations are always about information which has just become known.

Knowledge is in the form of facts, and each fact has a particular confidence value. Expectations are facts which have been selected. Violated expectations are surprising in proportion to their confidence values.

Active expectation failure leads to proportionally more surprise than passive expectation failure, for the same underlying violated fact.

4.2 Design Requirements

In this section a program demonstrating the proposed computational model of surprise is specified.

4.2.1 General Requirements

1. The program should accurately determine the level of surprisingness and causes of surprise of an input product that belongs to a particular product class.
2. The program should describe the cause or causes of the surprise, including:
    a. the elements of the product involved in producing surprise
    b. the expectations that were violated
    c. the reasons they were violated

4.2.2 High-level Program Requirements

1. The program should receive as input a valid, structured data-representation of a product. It receives both a partial description and full description of the product. The partial description should be a subset of the components making up the full description.

2. The program should output a textual report on the input product's surprisingness, including:
    a. The overall level of surprisingness of the product.
    b. Descriptions of the causes of surprise.
    c. The intensity of surprise for each cause.

4.2.3 Product Representation

1. The program should have a robust internal representation for products.
2. The representation will specify product structure.
    a. Certain aspects of behavior and function *may* be assumed or inferred.
3. The representation will be a composition of parts.
    a. Parts may have attributes.
    b. Parts may have certain relationships to other parts.
4. The representation must be **valid**. A valid product must:
    a. Be a member of the product class.
    b. Be reasonably recognizable by a person as belonging to the product class.

4.2.4 Input Data Format

1. The input data should be human-readable and writable as text.
2. It should be flexible to the extent that it can clearly represent a wide range of valid products from the product class, and nothing else.
3. The structure of the input data should reflect the structure of the product as represented and scrutinized internally by the program.

4.2.5 Facts

1. The program should have a representation for facts. (This representation will cover related concepts, such as expectations, too.)
2. The representation should cover many levels of concreteness or abstractness, so that it can represent:
    a. Predictions about the value of an attribute.
    b. Predictions about a range of possible values for an attribute.
    c. Relationships between attributes.
    d. Conditional facts, in which a condition property must hold true for a consequent property to be expected.

   e. The program should be able to determine a fact's **topic**, derived from the content of the fact. (Knowledge of expectation topics will facilitate deducing new expectations that are potentially discrepant with the input - See 5.1.4)
      i. If the fact is a prediction about an attribute's value or range of possible values, its topic is that attribute.
      ii. If the fact is a relationship between attributes, its topic is the set of attributes present in the relationship.
      iii. If the fact is a conditional, its topic is a mapping from the set of attributes in the conditional property to the set of attributes in the consequent property.
3. The program should represent a fact's **confidence**. Confidence is the expected likelihood that a fact will be correct.

4.2.6 Knowledge Representation

1. The program should use a knowledge base containing facts relevant to judging products in the product class.
2. The program should have a mechanism for detecting discrepancies between a concrete expectation and the input.
   a. We say that an expectation whose content is concrete, and whose topic is the same as a known portion of the input, is **discrepancy-compatible** with the input.

4.2.7 Expectations

1. Expectations should be represented by facts that have been selected from the knowledge base.
2. Expectations should be either active or passive.
   a. Active expectations should be selected from facts with topics dealing with information not available in the partial input description.
   b. Passive expectations should be selected from facts with any topics.
3. The program should be able to determine the intensity of surprise felt in response to the violation of an expectation by an input.
   a. A violated expectation with high confidence will cause surprise of greater intensity than one with low confidence.
4. The program should be able to detect multiple expectation violations for the same input product.

5 The Surputation System

In this section we present the design of the demonstration program, called Surputation.

The program produces predictions about how a person will react to seeing a pair of sunglasses. In this scenario, the person first sees a partially obscured view of the sunglasses, and then sees the whole thing. To recreate this scenario, the program receives as inputs a partial description of a pair of sunglasses, and a full description of the same sunglasses. The output of the program is a textual report on the surprisingness of the sunglasses (see Figure 5-1).

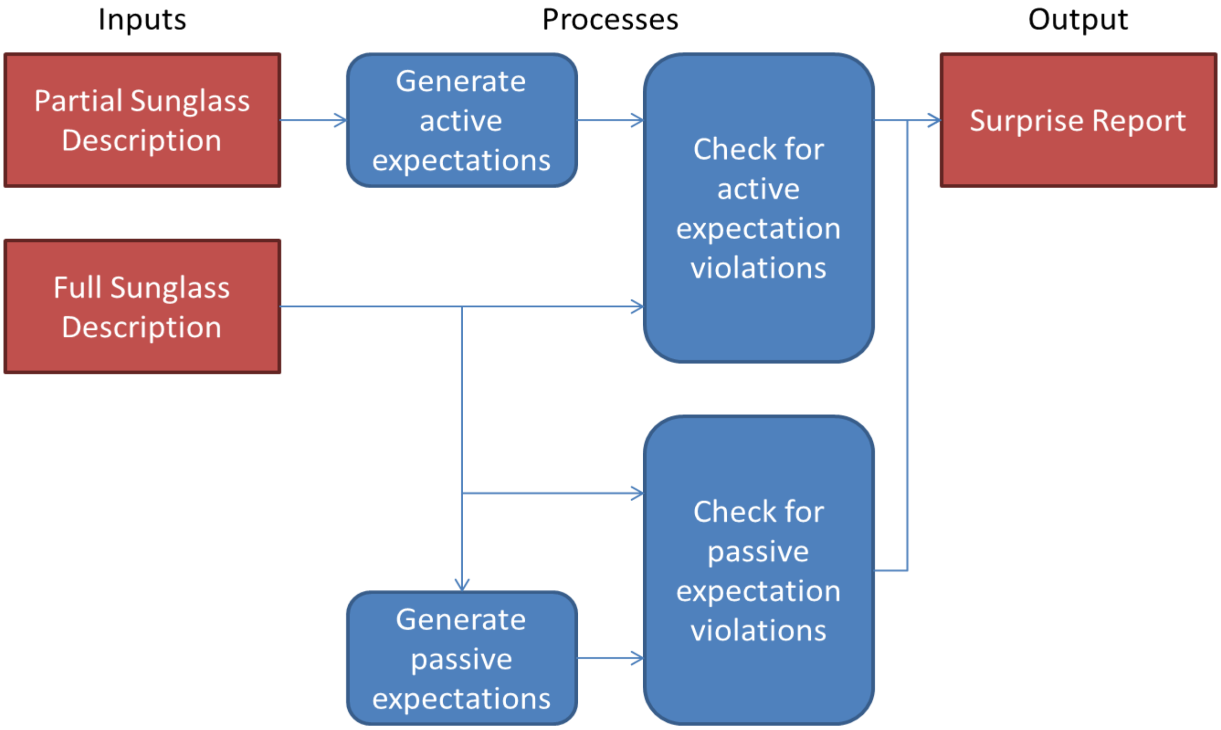
Inputs
Processes
Output
Partial Sunglass Description
Full Sunglass Description
Generate active expectations
Generate passive expectations
Check for active expectation violations
Check for passive expectation violations
Surprise Report

Figure 5-1: Program Flow

The program uses a knowledge representation of sunglass structure to create an internal representation of the sunglasses from each of the input descriptions. It uses a knowledge representation of facts about sunglasses to generate expectations about the input sunglasses.

The program first uses the partial description of the sunglasses to generate active expectations about the rest of the sunglasses. It then uses the full description of the sunglasses to check if any of the active expectations were violated. Next it uses the full description of the sunglasses to generate passive expectations, and checks these for violation as well. Finally, it produces a report of all the violated expectations, and outputs it.

First, the computational representations of sunglasses are discussed. Next the representation of knowledge is discussed. Next the mechanism of expectation formation and overall flow of the program is discussed.

5.1 Knowledge Representation

The program uses two kinds of knowledge representation, each playing a unique role in generating expectations and checking them for violation.

First, there is the knowledge used to represent general facts about sunglasses. This is referred to as generic knowledge. This is the knowledge from which expectations are generated.

Second, there is the knowledge that governs how a given pair of sunglasses is structurally represented by the program. It is referred to as structural knowledge. This knowledge is used to relate the generic knowledge to the particular pair of sunglasses input to the program.

A single type system is involved in both generic and structural knowledge. In order to discuss either kind of knowledge in further depth, the type system must be described first.

5.1.1 Component Types

The program uses a system of types describing the different components making up a pair of sunglasses. Sunglasses are conceptually broken down into the following set of component types:

- Left and right arm
- Left and right frame
- Left and right lens
- Bridge

The system of types is used in both generic knowledge and structural knowledge. Generic knowledge is made up of a collection of facts about the (types of) components above. Structural knowledge is a set of rules dictating how particular components, called instances of the above types, may be specified and combined together to represent a particular pair of sunglasses.

The system of types acts as a conceptual glue between the general facts of generic knowledge and the particular instances described using structural knowledge.

Note that for a product class other than sunglasses, it is likely necessary to use a different system of types.

5.1.2 Generic Knowledge

Generic knowledge is a knowledge representation of general facts about sunglasses. Some examples of facts found in the program's generic knowledge base are:

- *Arms are usually metal or plastic.*
- *There are always two lenses.*
- *When the frame is metal, the sunglasses are often heavy.*

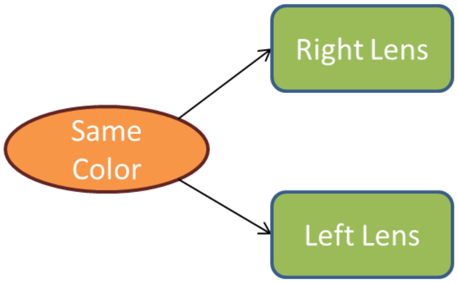

Figure 5-2: The *Same Color* fact applies to the component types *Left Lens* and *Right Lens*

Each fact is associated with one or more component types to which it is applicable. These are called the fact's *topics*. In the diagram above, the topics are *Left Lens* and *Right Lens.* Additionally, facts have a confidence value, represented as a number between 0 and 1. The higher this value is, the more likely it is that the fact is true of any particular pair of sunglasses. In the example sentences above, the words "usually", "always" and "often" are used to roughly indicate the fact's confidence value.

5.1.3 Structural Knowledge

Structural knowledge dictates how specific pairs of sunglasses are represented by the program. A pair of sunglasses is defined by a collection of component instances.

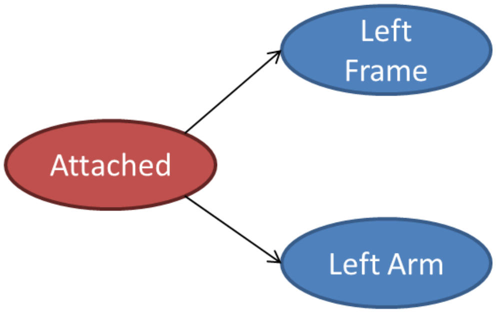

Figure 5-3: The *Attached* relation relates the component instances *Left Arm* and *Left Frame*

Each component instance has a type, such as *Left Arm* or *Left Frame*. Component instances may be conceptually related to one another via *relations*. An example of a relation is *Attached*, which is used to indicate that two component instances are physically attached to one another. Component instances may also have a set of attribute-value pairs. These are used to specify details of a component instance. For example, if a pair of sunglasses has an orange-tinted left lens, then the corresponding *Left Lens* instance will have the attribute *color = orange*.

5.2 Expectations

The program generates expectations, and then checks these expectations for violation. This is the basic mechanism by which surprise is detected; a violated expectation indicates surprise. To the program, an expectation is just a fact from the generic knowledge base that has been selected through the process of expectation generation. The program generates expectations in two ways – actively, and passively. In this section, the two methods that generate expectations are described. Then, the detection of expectation violations is explained.

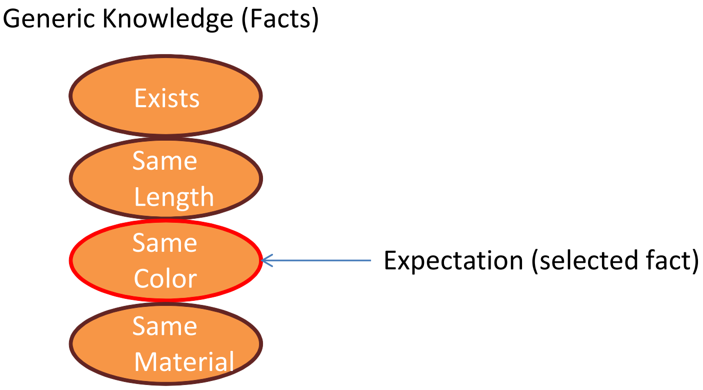

Figure 5-4: A fact in the generic knowledge base is selected to be an Active Expectation

5.2.1 Active Expectation Generation

Active expectations in the program correspond to expectations that a person is explicitly thinking of. They have the following straightforward “lifespan”, from formation to violation:

1. Process partial sunglasses description.
2. Select facts that cannot be verified with just the partial description. These become active expectations.
3. Process full sunglasses description.
4. Identify the active expectations that have been violated by the new information.

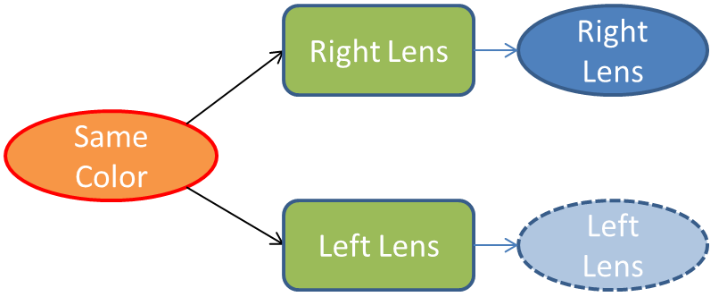

Figure 5-5: The *Same Color* fact is selected because one of its topics, *Left Lens*, is not in the partial input description. It is thus made into an *Active Expectation*.

Active expectations are generated before the information that violates them is available. This differentiates them from passive expectations, which are generated after the offending information is processed. After processing the partial sunglasses description, the program selects facts with topics pertaining to as-yet unknown information. These become the set of active expectations.

5.2.2 Passive Expectation Generation

Passive expectations in the program correspond to a person's after-the-fact realizations of surprise. The sequence of events for passive expectations is shorter than that of active expectations:

1. Process full sunglasses description.
2. Select facts that are known to be violated, and which were not selected as Active Expectations. These become passive expectations.

Passive expectations are generated only after the full description of the sunglasses has been processed. Unlike with active expectations, there is no particular criteria by which facts are selected to be passive expectations. The facts selected will be those that have been violated by the full description. Surprises caused by passive expectation failure will also have a lower intensity than active expectation failures.

5.2.3 Checking For Expectation Violations

Once the active expectations have been generated, they are checked against the full sunglasses description for violation. For each of the fact's topics, the program identifies the input sunglasses' corresponding component instance. The program tests the fact against these component instances. If the fact holds true over the instance, then the expectation is not violated. If it is false for this instance, then the expectation is violated.

5.3 Calculation Of Surprise

The total surprise of the input sunglasses is calculated using all of the violated expectations. The surprise of a particular failed expectation *e*, or *S(e)* is calculated as the expectation's confidence value multiplied by a coefficient *t,* or *S(e) = confidence(e) * t*. If it is an active expectation, then *t=1.*If it is a passive expectation, then *t=.7*. This way, surprises by way of active expectation failure have slightly higher intensity than ones caused by passive expectation failure. This is done in order to support the model's hypothesis that active expectation failure causes greater surprise than passive expectation failure. The total surprise is calculated as the sum of the surprisingness of all violated expectations, divided by the number of potentially violated expectations. This division is used to normalize the result within a range. For this program, the number of potentially violated expectations is the number of facts in the knowledge base. A more sophisticated approach might put an upper limit on the number of active and passive expectations that can be formed, based on principles of attention.

5.4 Report Generation

Once the program has computed the total surprise for the input product, it is ready to generate and output the report. The report is a textual description of the total surprisingness of the product, and a listing in plain English of the causes of surprise. Each cause corresponds with one violated expectation. See the Appendix for an example of program output.

6 Program Implementation

Here we discuss the program implementation. We also discuss the programming language used to write the demonstration program. We then describe the input method used to read descriptions of sunglasses into the program. Finally, the data structures used to represent the design concepts laid out in chapter 5 are discussed

6.1 Programming Language

The programming language used for the demonstration program was Python version 3.2. Python was chosen for a number of reasons. For one thing, this author is very familiar and comfortable with programming in Python, and so using it gained a lot in terms of programming speed. Python supports object-oriented programming, which was used to create abstractions for many of the concepts discussed in chapter 6.

6.2 Input Method

The program receives as input a partial and a full description of a pair of sunglasses. A pair of sunglasses is described as a set of component instances, with a partial set of these instances being the partial description. A mini-language was used to define input for the program. Descriptions of sunglasses are specified in this language, and then interpreted by the program to produce its internal representation.

The language supports the concepts laid out in section 6.1.3. Namely, it supports the specification of a set of component instances. Each instance is given a name and a type. The name is used to refer to the component instance later, while the type dictates which part of the sunglasses the component instance is. The language requires that each of the following names must be specified:

- *left_lens*
- *right_lens*
- *left_frame*
- *right_frame*
- *left_arm*
- *right_arm*
- *bridge*

In addition, the language allows for component instances to have a set of attribute-value pairs specified when they are created. An example of a component instance declaration is as follows:

left_lens = Lens(color = "black")

In the above example, a *Lens* instance is created with the name *left_lens*. It is given the attribute *color = black*.

In certain cases it is necessary to specify an absent component instance. To do this, the instance in question is assigned to *Empty()*, as in the following example:

right_arm = Empty()

This will indicate that the sunglasses described are missing the right arm, in contrast to the description writer simply forgetting to specify them. If the description writer does forget the specify them, an error will be signaled.

The language also supports the specification of relations between component instances. Relations are specified declaratively, as follows:

Attached(left_lens, left_frame)

Once all component instances and relations have been declared, all that remains to do is to specify which components belong to the partial description. This is done with the following notation:

partial = {left_arm, right_arm}

As shown above, the members of the partial description are grouped together in comma-separated list enclosed in braces. This set is assigned to the name *partial*.

Input descriptions are saved in text files ending with ".sgl". The demonstration program is given the file name as input when it is run, and it reads from and interprets the file. Because the specification of this mini-language has much in common with standard Python syntax, the Python interpreter is used to interpret the description files, with no need for a hand-written parser.

6.3 Data Structures

A class hierarchy built into the program was used to represent the different component types. The type *Component* was the root class in the hierarchy, with each concrete component type, such as *Lens* or *Bridge*, having its own *Component* subclass. Instances of any of these classes correspond exactly to particular component instances.

A class was used to represent Facts. Instances of the *Fact* class had a confidence value between 0 and 1, and a set of references to their topic component types. Each particular kind of fact had its own *Fact* subclass. The subclass was responsible for implementing a method *activate().* The *activate()* method returns either True or False to indicate whether, given the partial set of component instances from input, the fact would become selected to be an active expectation. Additionally, *Fact* subclasses implemented a method *check()*, which would be passed the full set of component instances, and returned True or False to indicate whether the fact held true for the sunglasses or not.

The generic knowledge base was represented by a collection of *Fact* instances. The selecting of facts to become active expectations involves iterating over all the facts in the collection and calling their *activate()* methods to see if they should be selected. Passive expectation selection worked similarly; the set of facts that have not been selected as active expectations are iterated over and their *check()* methods are called. Those facts whose call returns false are made passive expectations. Once all active and passive expectations have been selected, the set of all expectations is iterated over, checked for violation (via the *check()* method), and used to update the count of total surprise.

7 Evaluation

An experiment was conducted to assess the accuracy of our computational model of surprise. The experiment measured subjects' reactions to surprising images of sunglasses. We would compare these results to predictions made by the Surputation system. The following is a description of our experimental design. Next we discuss the conditions used to run the demonstration program to match the conditions from the experiment.

7.1 Experimental Design

In the experiment, images were presented in a way that was intended to cause either active or passive expectation failure. We hypothesized that there would be a difference in the intensity of experienced surprise depending on whether the surprise was active or passive. Active expectation failure was hypothesized to cause a more intense reaction than passive. We discuss the experimental procedure. Next we discuss the experimental conditions.

7.1.1 Procedure

1. Subjects were seated in front of computer screens in a WPI Social Science Lab.
2. Subjects were given Informed Consent Forms to read and sign.
3. They used web browsers to interact with a web application for the experiment.
4. They began by answering the following background questions before being allowed to proceed:
    - What is your age?
    - What is your gender?
        - Male
        - Female
        - I decline to answer.
    - How frequently do you use computers?
        - 30 minutes a day or less.
        - 30 minutes to 2 hours a day.
        - 2 to 5 hours a day.
        - Over 5 hours a day.
    - How frequently do you wear glasses or sunglasses?
        - Never.
        - 0 to 3 hours a week.
        - 3 to 7 hours a week.
        - Most of the time.

5. Subjects then participated in four trials without interruption. In each trial, the following happened:
    - First an image was displayed on the screen. The image was a partially obscured photograph of a pair of sunglasses. The obstruction was a black rectangle covering part of the image. After viewing it, the subject clicked a button to advance.
    - Next a second image was displayed. It was the same photograph from the first image, but fully visible with no obstruction. After viewing it, the subject clicked a button to advance.
    - The subject then answered the following questions:
        - Please indicate how strongly you agree or disagree with the following statements:
            - The contents of the second image were expected.
                - 5-point scale
            - The sunglasses shown in this trial were surprising.
                - 5-point scale
    - Once the questions were answered, the subject proceeded to the next task by selecting a button.
6. Once all four trials were complete, the following final questions were asked:
    - Please indicate how strongly you agree or disagree with the following statements:
        - The first (obscured) image affected my expectations about the second (fully revealed) image.
            - 5-point scale
        - The first (obscured) image affected how surprising I found the second (fully revealed) image.
            - 5-point scale

7.1.2 Trial conditions and randomization

Of the four images used in the trials, three had surprising features, and one was a control. The four surprise conditions are thus:

- Missing right arm (Figure 7-1)
- Short left arm (Figure 7-2)
- Different colored lenses (Figure 7-3)
- Normal (control condition) (Figure 7-4)

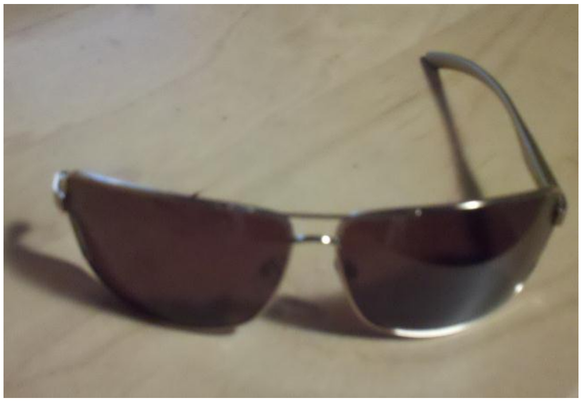

Figure 7-1: Missing Right Arm condition

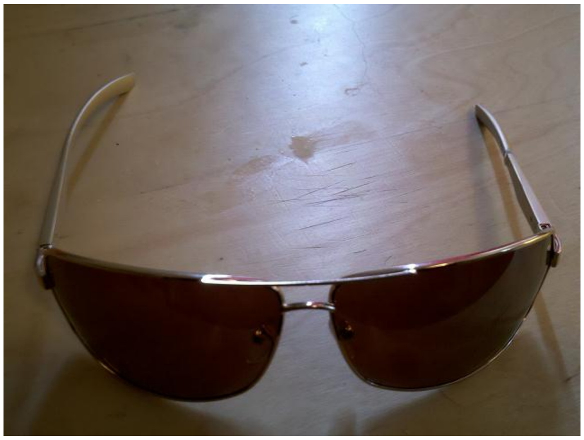

Figure 7-2: Short Left Arm condition

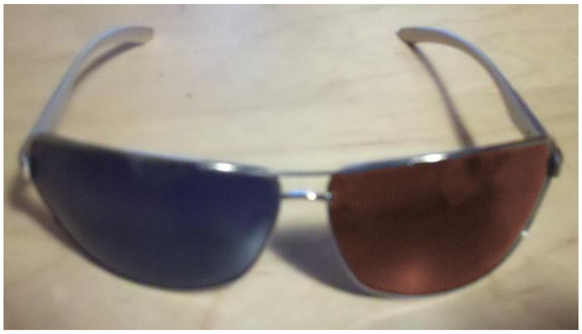

Figure 7-3: Different Colored Lenses condition

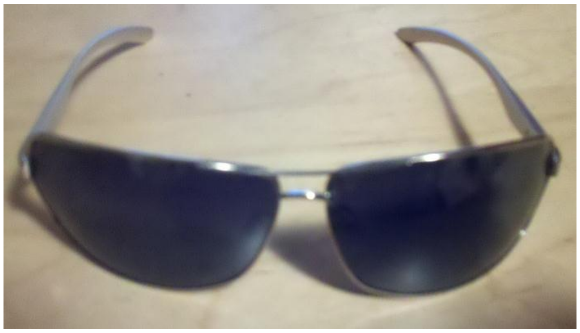

Figure 7-4: Control condition

Another condition determined two ways in which the surprise was presented to the subject:

- Active Expectation Failure. In this condition, the surprising feature is obscured by the black box in the first image. (Figure 7-5)
- Passive Expectation Failure. In this condition, the surprising feature is apparent in the first image, with the black box obscuring an unrelated, and unsurprising, portion of the sunglasses. (Figure 7-6)

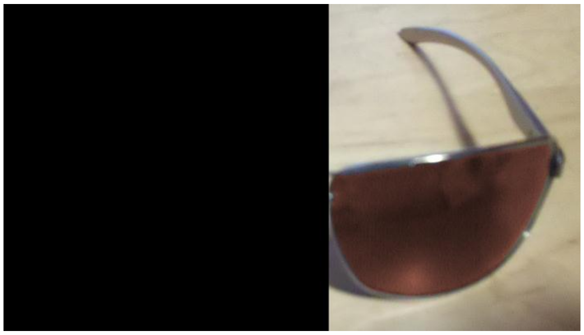

Figure 7-5: "Active" condition for Different Colored Lenses partial image

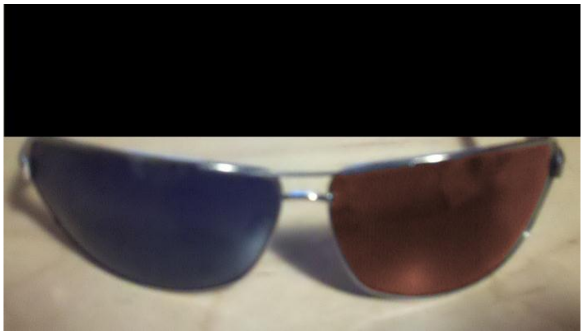

Figure 7-6: "Passive" condition for Different Colored Lenses partial image

Note that the control condition also has two possible presentations, with two locations chosen for the black box to appear in the first image of the control trial. Neither condition obscures or reveals any surprising features. (Figures 7-7 and 7-8)

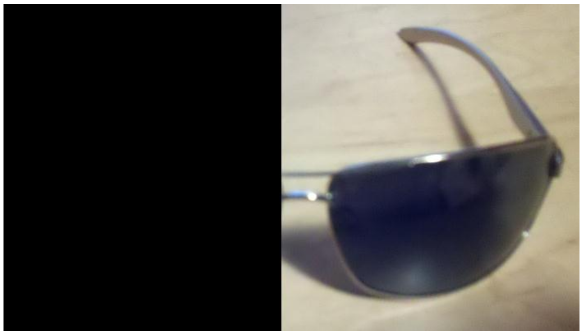

Figure 7-7: First partial image for Control

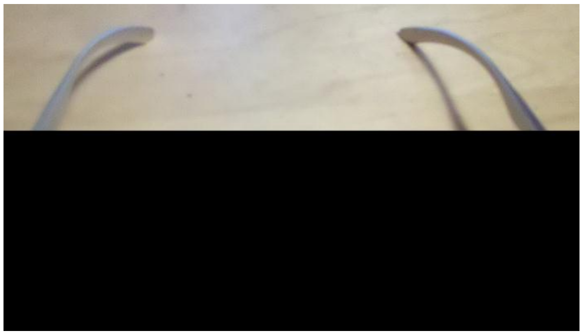

Figure 7-8: Second partial image for Control

Each subject was provided with all four surprise conditions in their trials. The order of the trials was random. Of the three non-control (surprising) trials, two used Active Expectation Failure and one used Passive Expectation Failure, distributed randomly. The presentation of the control trial was also distributed randomly.

7.2 Program Testing

We created a set of inputs for the demonstration program that would closely mimic the conditions of the experiment. The program takes as input both a partial and full description of a pair of sunglasses. In the experiment, each trial consists of the same setup. First the subject is shown a partial image of some sunglasses, and then the full image. The subject then answers questions about the extent to which they expected and were surprised by the images. The program gives its responses to the input by outputting a report indicating the surprisingness of the input. The second of the two questions answered for each trial was directly measuring subjects' assessment of the sunglasses' surprisingness. Thus, by giving the program input describing sunglasses similar to those appearing in the images in the experiment, we could measure the program's accuracy in predicting subject reactions.

Properly assessing the program's accuracy is dependent on accurately describing the various images used in experiment trials. To do this, eight sets of input were created for the program. Each input corresponds to the active or passive condition of each of the four surprise conditions in the experiment.

For each input, the description specified the pair of sunglasses appearing in the corresponding surprise condition. For instance, the description for the “Different Colored Lenses” condition had a right lens with attribute *color = black* and left lens with attribute *color = orange*.

Furthermore, the set of components present in the input’s partial description was dependent on what was visible in the first image of the corresponding experiment trial. For instance, for the “Active” trial of the “Different Colored Lenses” condition, the set of components in the partial description was *Right Arm, Right Frame, Right Lens*. In the “Passive” trial, it was *Left Frame, Left Lens, Bridge, Right Frame, Right Lens*. Note that these sets both correspond to the set of components visible in the actual images used.

By carefully matching the input descriptions to their corresponding trial conditions, we aimed to get the most accurate predictions possible from the demonstration program.

8 Results And Evaluation

In this chapter we discuss the results of the experiment and demonstration program evaluation. We examine the implications of the results for the accuracy of the demonstration program.

8.1 Program Evaluation Results

As discussed in section 7.2, the program was run with 8 inputs, one for each “Active” and “Passive” condition in the four surprise trials. For each run, a number was returned by the program indicating the total surprisingness calculated for the run. The following are the results:

Table 8-1 – Total surprise intensities given by the demonstration program

Surprise Condition	**Active Condition**	**Passive Condition**
Missing Right Arm	3.6	2.52
Different Colored Lenses	2.8	1.96
Short Left Arm	2.8	1.96
Control	0	0

The results rate “Missing Right Arm” as the most surprising condition, with “Different Colored Lenses” and “Short Left Arm” tied with the next most surprise, and the “Control” condition having the least surprisingness. The results show a trend of rating inputs from the “Active” condition more surprising than their “Passive” counterparts in the same surprise condition.

In addition to the total surprise intensities in Table 8-1, the program also output textual descriptions of the causes of surprise for each trial (see Appendix). The output properly labeled trials in the “Active” condition as having Active Expectation failure, and trials in the “Passive” condition as having Passive Expectation failure. Furthermore, it gave distinct causes for surprise in all four of the different surprise conditions, which all correctly reflected the intended surprising feature for the trial.

8.2 Experiment Results

There were 12 experimental subjects. Five of these subjects completed the experiment in a lab. The lab had a number of computers, with dividers between each workspace. The other seven subjects completed the experiment from personal computers, and accessed the experiment over the internet. The variation in location was unlikely to affect subjects’ abilities to interact with a computer, and so it is unlikely that this affected performance. All subjects completed all four trials, in randomized order. The following shows the mean questionnaire results for question two for each of the 8 trial conditions:

Table 8-2 – Mean total surprise intensities given by experimental subjects

Surprise Condition	Active Condition	Passive Condition
Missing Right Arm	3.2	2.4
Different Colored Lenses	3.375	2.5
Short Left Arm	2	3.333
Control	0.7	0.7

It must be noted that a sample size of only 12 would not be enough to produce any statistically significant results. We will instead examine these data and give an interpretation for the sake of insight.

These results show that "Different Colored Lenses" and "Missing Right Arm" yielded the highest surprise ratings in the "Active" condition. Following these were "Short Left Arm" and then "Control". For the "Passive" condition, "Short Left Arm" had highest surprise, followed by "Different Colored Lenses" and "Missing Right Arm", followed by "Control".

It is noted that the mean total surprise for the "Control" condition was 0.7, a result somewhat higher than zero. This is likely due to the nature of the questionnaires used in the experiment; they gave the option to either "Disagree" or "Strongly Disagree" with the statement that the sunglasses are surprising. Some subjects chose not to use the stronger option, and the result is a mean value between one and zero, rather than exactly zero.

For the surprise conditions "Missing Right Arm" and "Different Colored Lenses" there is a higher relative surprise rating in "Active" than in "Passive". These results were expected, given our hypothesis that active expectation failure leads to a greater surprise than passive. The opposite was true for the "Short Left Arm" condition. While there are many possible explanations for this phenomenon, we believe that the images used in the experiment for this condition were unintentionally reducing some active expectations.

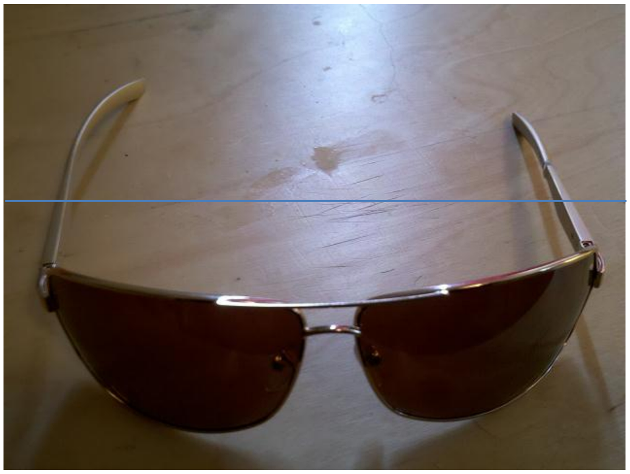

Figure 8-1 - Full image from "Short Left Arm" condition. Note that the added dividing line minimizes the appearance of a difference in arm length.

Figure 8-1 is the full image used for the “Short Left Arm” condition. Notice that aside from the shortened left arm on the sunglasses, the angle of the sunglasses in the image is slightly slanted. This may have served to make the left arm appear to be closer to the length of the right arm than it actually was.

8.3 Analysis of Results

The results of the demonstration program evaluation and experiment were compared. Because of the unintended visual biases in the “Short Left Arm” condition, this condition was left out of the comparison. The following is a chart that shows the results for the remaining conditions together.

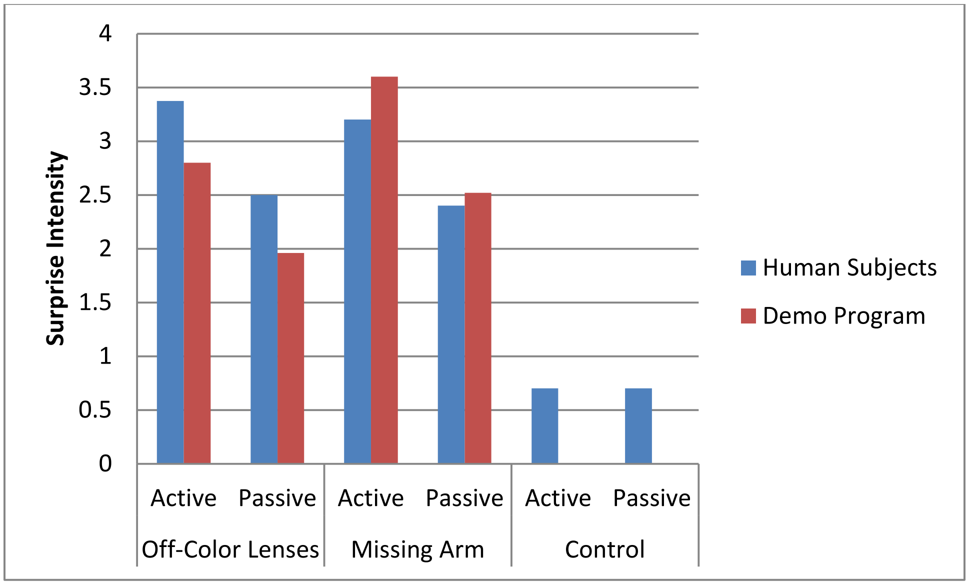

Figure 8-2 – Chart depicting total surprise ratings given by human subjects (blue) and the demonstration program (red)

One property of the experimental results that is clearly preserved by the demonstration program is the higher surprise rating for the “Active” condition than the “Passive” condition within each surprise condition. The program implemented this via the “t” value discussed in section 5.3. The program also successfully distinguished between the causes of surprise for the different surprise conditions, as evidenced by the distinct textual outputs for each of the different surprise conditions used for input. The program does not appear to capture accurately the relative intensities measured between different surprise conditions. However, this outcome was predictable, given the ad-hoc nature of the demonstration program’s knowledge base; facts in the knowledge base were given confidence values according to the author’s intuitions alone. It is therefore expected that, given the chance to adjust these confidence values, the demonstration program would behave more accurately.

9 Conclusion

In this section we give our conclusions on the success of the project. We then detail potential future work. Finally we discuss the author's project experience.

9.1 Project Assessment

This project set out to develop a computational model of surprise. Such a model was developed and implemented in a demonstration program. The model aimed specifically to be applicable to real-world, testable scenarios concerning surprise reactions to products. This too was achieved, as demonstrated by the evaluation of the demonstration program using an experiment with human subjects.

It is necessary to discuss the particulars of the surprise model, and their accuracy given the results of the evaluation. First, we discuss the model's take on expectations as they pertain to surprise. Active expectations are those that have been formed explicitly in a person's mind. They are formed by a combination of the person's general knowledge and their understanding of the current context. Namely, active expectations will reflect what a person believes is about to happen. When an active expectation is violated it leads to a large amount of surprise. Passive expectations are formed in the mind only after a surprising event has occurred. In our model, they act as a "catch-all" for surprises not caused by active expectations. They also cause surprise when it is realized that they have been violated, but not as much as active expectations.

The model's conception of active and passive expectations appears to have some merit given the experimental results. The experiment showed that for conditions intended to cause the hypothesized active expectation failure, people did seem to feel greater surprise than when the same surprise was presented in a passive way. However, there were not enough participants to achieve statistically significant results. Additionally, confusion involving the "Short Left Arm" trial muddies the waters for this topic, making it clear that additional data must be collected, with a slightly revised experiment.

Most of the requirements laid out in section 4 were met. Of particular exception is the set of requirements about facts (Section 4.2.5.2). While all of the types of facts, of varying levels of concreteness, were possible to represent, only a limited set of facts were ever implemented. There was no mechanism for extending the knowledge base beyond the set of facts built into the program.

Overall, the project was a success. Although the demonstration program did produce some inaccuracies, the intent to produce a working, testable instantiation of the hypothesized model of surprise was successful. What's more, it made a case for the most concrete prediction of the model, namely that active expectation failure is more surprising than passive expectation failure, given the same cause of surprise.

9.2 Sources of Error and Future Work

Next we touch upon some of the inaccurate predictions produced by the demonstration program. The demonstration program was not able to accurately predict the relative intensities of surprise between different surprise conditions. The immediate reason for this is that the facts from the program's knowledge base had incorrect confidence values. It would be possible to alleviate the inaccuracy by adjusting the confidence values. However, this is not a satisfactory solution to the problem of inaccurate predictions of surprise intensity. For this solution is mere trial and error; some arbitrary values are chosen by guesswork, and then adjusted as necessary based on performance.

A good solution to this problem would propose a way of computing the right confidence values before they are tested against real humans. Note that the surprise model from this report takes no stance on this issue; it does not concern itself directly with the construction of an accurate knowledge base. Instead, a knowledge base was constructed by intuition in order to produce a working, testable demonstration program.

Thus, one area of future work would be the construction of knowledge bases that produce accurate predictions of surprise. This includes the accurate use of confidence values, as well as answers to some more fundamental questions. Of particular concern is how to structure knowledge about multiple categories of products. The problem of categorical "corner cases", in which products exist that have no accurate instantiation in a particular knowledge representation, or whose categorization is ambiguous, remains unsolved. How are different types of product represented as input? How does the program distinguish which product category a given input belongs to? The accurate prediction of surprise would serve as a concrete metric for testing different solutions to these difficult problems.

Another area of future work is the role of attention in expectation generation. A shortcoming of the demonstration program is that it considers every possible fact in its knowledge base as a potential active expectation. It is possible for the demonstration program to have every single fact in its knowledge base selected as an active expectation at once. This author believes that attention imposes limits on how much a person can expect in a given situation. Without attention to limit and guide the generation of expectations, a person could at any time be expecting a very large number of possible things to happen. It is therefore important to examine the ways that attention might decide which few expectations do, in practice, become active.

Another issue is this model's overly simple conception of Passive Expectation Failure. Effectively, this model uses Passive Expectation as a catch-all for any source of surprise that is not Active Expectation Failure. Ortony and Partridge (see section 2.3.1) describe Passive Expectation Failure as one of three ways for surprise to occur. Their conception of Passive Expectation Failure is limited to those expectations which are "practically deducible" given the current set of Active Expectations. This project ended up making no attempt to limit its passive expectations at all; all facts in the knowledge base are fair game. There is also no deduction that occurs; the knowledge base is just searched for facts that are violated. What's more, this model has no mention of Ortony and Partridge's Unanticipated Incongruity. It would have involved using a mechanism by which the knowledge base is automatically updated when new input is received. This is another area of future work involving the design of the knowledge base.

If any of these features are to be implemented in future work, a good way of testing them must be designed. Many of these concepts were initially incorporated in this project's design requirements. But all attempts at designing them failed because we were not able to propose a simple way of testing whether they were worthwhile or not. So, an evaluation method of, for example, confirming the existence of and measuring Unanticipated Incongruity, is a highly important area of future work.

9.3 Project Experience

This project was a very educational experience. I initially took it on because I found the area of Computational Creativity very interesting. After learning a lot about it, I can say that this is still true. Aside from what I gained academically, I learned a lot about what it takes to complete a long, difficult, individual project.

One of the most difficult aspects of the project was the large amount of writing involved. I have never had to write so much before, and I had a lot of trouble organizing my thoughts into something coherent. I also had difficulty with breaking down the writing conceptually, so that I knew what to write about for any given section. I have certainly improved a lot at writing reports.

The second difficulty I faced was the open-ended nature of the work. I have never worked on a project as academic as this one. I find that I am most productive when I have a concrete goal to work toward, some set of requirements that must be met. For this project, it often felt like the requirements were changing too quickly to be concrete. There was often no good metric for measuring my progress. My goals for the project, for the demonstration program, and for the evaluation, were not quite clear to me until the very end. This was frustrating, and made it very hard for me to motivate myself to get things done. Now, having completed the project, I understand that to an extent, this is the nature of research. I also have a better concept of the sorts of long-term goals that are present throughout a big project, but which I just wasn't familiar with.

There is a lot that I would have done differently, in retrospect. I would have made more concrete goals for myself where possible. I would have developed more metrics for measuring my current level of success. For instance, the knowledge representation used in the demonstration program ended up simple and minimal. This was because in the end, I did not have any way to measure benefits beyond those of such a simple representation. There was much discussion of knowledge representation for designed products that I read about, and discussed with my advisor, but I had no reason to add complexity given what I found myself doing. The result was, I wasted a lot of time going back and forth about what knowledge representation to use, and ended up using an extremely simple one with little academic grounding. Nothing more was required for the surprise conditions I planned to test. If I had come up with a good metric for what my knowledge representation was required to do, I would have been able to spend much less time spinning my wheels, and made more progress.

This project required me to improve my time management skills. I had to schedule my time to work on all the different aspects of the project—programming, designing, reading or researching, and of course, writing. I had to adapt my goals to what seemed to be working as I progressed. It was necessary to develop my critical thinking skills, as I had to decide which parts of the large body of work that I read about were actually relevant to my project. By the end of the project, I had acquired a better ability to finesse the abstract into something concrete. Turning something abstract, such as a theoretical model, into something concrete, requires going beyond the model's concepts, and coming up with ways it can be manifest in reality. That is, it is necessary to devise and run experiments or tests to know what to design and how it should behave.

10 References

Besemer, S. P. (2011). *The CPAM*. Retrieved from http://ideafusion.biz/home/creative-product-analysis-model

Besemer, S. P., & Treffinger, D. J. (1981). Analysis of Creative Products: Review and Synthesis. *Journal of Creative Behavior* (15), 158-178.

Ludden, G. (2008). *Sensory Incongruity and Surprise in Product Design.*

Macedo, L., Cardoso, A., & Reisenzein, R. (2006). A Surprise-based Agent Architecture. *Cybernetics and Systems Conference.* Vienna.

Maher, M. L. (2010). Evaluating Creativity in Humans, Computers, and Collectively Intelligent Systems. *Creativity and Innovation in Design Conference.* Arrhus.

Meyer, W., Reisenzein, R., & Schützwohl, A. (1997). Toward a Process Analysis of Emotions: The Case of Surprise. *Motivation and Emotion, 21*(3).

Ortony, A., & Partridge, D. (1987). Surprisingness and Expectation Failure: What's the Difference? *Proceedings of the 10th International Joint Conference on Artificial intelligence* (pp. 106-108). San Francisco: Morgan Kaufmann Publishers Inc.

11 Appendix: Program Output

```
surprise report for missing_right_arm_active.sgl
active expectation failures:
   [Right Arm exists]
passive expectation failures:
total surprise: 0.9
normalized total surprise: 0.060000000000000005

surprise report for missing_right_arm_passive.sgl
active expectation failures:
passive expectation failures:
   [Right Arm exists]
total surprise: 0.63
normalized total surprise: 0.042

surprise report for different_colored_lenses_active.sgl
active expectation failures:
   [Right Lens and Left Lens have the same color]
passive expectation failures:
total surprise: 0.7
normalized total surprise: 0.04666666666666666

surprise report for different_colored_lenses_passive.sgl
active expectation failures:
passive expectation failures:
   [Right Lens and Left Lens have the same color]
total surprise: 0.48999999999999994
normalized total surprise: 0.03266666666666666

surprise report for short_left_arm_active.sgl
active expectation failures:
   [Left Arm and Right Arm have equal length]
passive expectation failures:
total surprise: 0.7
```

```
normalized total surprise: 0.04666666666666666

surprise report for short_left_arm_passive.sgl
active expectation failures:
passive expectation failures:
   [Left Arm and Right Arm have equal length]
total surprise: 0.48999999999999994
normalized total surprise: 0.03266666666666666

surprise report for control_active.sgl
active expectation failures:
passive expectation failures:
total surprise: 0
normalized total surprise: 0.0

surprise report for control_passive.sgl
active expectation failures:
passive expectation failures:
total surprise: 0
normalized total surprise: 0.0
```