FABIANO DA SILVEIRA PINTO

GRAMÁTICAS DE FORMAS COMO MODELO COMPUTACIONAL TEÓRICO

O poder computacional de gramáticas de formas comparado a outros modelos computacionais teóricos, como máquina de Turing e gramática generativa de Chomsky

PORTO ALEGRE – RS

2010

FABIANO DA SILVEIRA PINTO

GRAMÁTICAS DE FORMAS COMO MODELO COMPUTACIONAL TEÓRICO

O poder computacional de gramáticas de formas comparado a outros modelos computacionais teóricos, como máquina de Turing e gramática generativa de Chomsky

Trabalho de conclusão de curso apresentado ao Instituto de Informática da Universidade Federal do Rio Grande do Sul, requisito para a obtenção do grau em Ciência da Computação.

Orientador: Prof. Dr. Luis C. Lamb.

PORTO ALEGRE – RS

2010

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Graduação: Profª. Valquiria Linck Bassani
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador da CIC: Prof. João César Netto
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

FABIANO DA SILVEIRA PINTO

GRAMÁTICAS DE FORMAS COMO MODELO COMPUTACIONAL TEÓRICO

O poder computacional de gramáticas de formas comparado a outros modelos computacionais teóricos, como máquina de Turing e gramática generativa de Chomsky

Trabalho de conclusão de curso apresentado ao Instituto de Informática da Universidade Federal do Rio Grande do Sul, requisito para a obtenção do grau em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Luis C. Lamb
Orientador – Universidade Federal do Rio Grande do Sul

Prof. Dr. Álvaro Freiras Moreira
Universidade Federal do Rio Grande do Sul

Prof. Dr. Marcelo Soares Pimenta
Universidade Federal do Rio Grande do Sul

Prof. Dr. Paulo Fernando Blauth Menezes
Universidade Federal do Rio Grande do Sul

PORTO ALEGRE – RS, 6 DE JULHO DE 2010

AGRADECIMENTOS

Agradeço inicialmente à minha família por ter sempre acreditado em minha capacidade e também à minha namorada Fernanda por toda sua a paciência e carinho.

Também sou grato pelo apoio e orientação do prof. Dr. Benamy Turkienicz e da profª. Dra. Rosirene Mayer, durante meu período como bolsista de Ciência da Computação no Laboratório para Simulação e Modelagem em Arquitetura e Urbanismo (SIMMLAB) da Faculdade de Arquitetura da Universidade Federal do Rio Grande do Sul.

Finalmente agradeço a todos os professores e servidores do Instituto de Informática, responsáveis pela excelência máxima do curso de Ciência da Computação desta universidade.

Ao meu pai, Pedro Paulo da Costa Pinto (*in memoriam*), com quem trocava mensagens em antigos cartões perfurados e que me ensinou os princípios da recursividade brincando com uma Torre de Hanoi de plástico.

RESUMO

Gramáticas de formas têm sido as principais ferramentas para o estudo envolvendo linguagens de design, permitindo os processos de análise e de síntese sobre elementos de desenho, servindo em última instância para a concepção e manipulação de estilos estéticos específicos.

Comparado a outros modelos generativos, especialmente os simbólicos como a gramática generativa de Chomsky, e considerando as equivalências com outros modelos computacionais teóricos como a máquina de Turing, surge um significante diferencial, a emergência de formas, não havendo similar imediato entre sistemas de derivação sobre símbolos.

As características únicas das gramáticas de formas apresentam qualidades especiais para a manipulação de elementos estéticos, especialmente para o desenvolvimento e a exploração de linguagens de design. Este estudo apresenta tais características, buscando desenvolver processos de máxima eficiência para o desenvolvimento e exploração de processos generativos sobre elementos estéticos.

Palavras-chave: gramática de formas, linguagem de design, modelo generativo, gramática generativa, Chomsky, modelo computacional, máquina de Turing, emergência de formas, computabilidade, teoria das categorias.

ABSTRACT

Shape grammars have been the main tools for the study of design languages, allowing the processes of analysis and synthesis of design elements, serving ultimately to the development and handling of specific aesthetic styles.

Compared to other generative models, especially the symbolic ones as Chomsky's generative grammar, and considering the equivalence with other theoretical computational models such as the Turing machine, there is a significant difference, the shape emergence, which has no direct correspondence on symbolic derivational systems.

The unique aspects of shape grammars show special qualities for aesthetic elements handling, especially for the development and exploitation of design languages. This study presents such characteristics, aiming to develop processes for efficient exploration of aesthetic elements and languages.

Keywords: shape grammar, design language, generative model, generative grammar, Chomsky, computational model, Turing machine, shape emergence, computability, category theory.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

O conceito de emergência é crucial para muitos sistemas computacionais. A emergência de formas tem importante papel no processo criativo e é definida por diversos autores como o surgimento de formas não definidas explicitamente em uma gramática de formas (Gips, 1975) (Stiny, 1975). Formas emergentes não são apenas os resultados de computações sobre uma gramática de formas, mas sim a entrada para novas computações. A emergência de formas, caracterizada através de gramáticas de formas, é apresentada aqui e a seguir comparada a outros modelos computacionais teóricos, especialmente a máquina de Turing e a gramática generativa de Chomsky, consideradas equivalentes em seu poder computacional. Diferentes tipos de emergência de formas são discutidos aqui, especialmente a emergência antecipada, a emergência possível e a emergência não antecipada. A emergência não antecipada, de maior interesse como modelo computacional comparativo, não é premeditada pelo autor ou utilizador da gramática, geralmente necessitando consecutivas derivações e o correto reconhecimento e parametrização de padrões para a sua identificação. Todavia, para alguns problemas interessantes de design é possível antecipar os resultados obtidos através de formas não antecipadas, definindo regras de acordo. Nesta abordagem, a teoria das categorias serve como ferramenta para a representação e estudo da linguagem resultante de gramáticas específicas, delimitando o universo de estudo e possibilitando o melhor entendimento das conseqüências derivacionais para uma determinada gramática.

Este trabalho está organizado em capítulos e no capítulo 2 são apresentados diferentes modelos computacionais, com destaque para modelos clássicos como a máquina de Turing e a gramática generativa de Chomsky. No capítulo 3 é apresentada a gramática de formas, um modelo computacional não

simbólico, com destaque para ambiente, estrutura e processos envolvidos, culminando com a aplicação da teoria das categorias como ferramenta para o estudo eficiente de gramáticas e linguagens. O capítulo 4 descreve os sistemas de *software* desenvolvidos durante este estudo, onde implementação computacional refere-se especificamente aquelas desenvolvidas como programas de computador, pois implementações de outras naturezas são possíveis, seja através de processos manuais de desenho ou até com a manipulação de formas utilizando dispositivos mecânicos. Finalmente, no capítulo 5 são apresentadas algumas aplicações práticas reais bem como algumas aplicações visando apenas a comparação entre diferentes modelos, por fim são discutidas algumas propriedades de especial interesse observadas em gramáticas de formas.

No tratamento de elementos visuais e espaciais, a emergência de formas a partir de uma gramática definida, permite a decomposição e recombinação de acordo com requisitos específicos. Esta pesquisa procurou desenvolver métodos para desenho assistido utilizando estratégias generativas, permitindo a construção de gramáticas de formas satisfatórias e a conseqüente produção de resultados para uma gramática definida, inclusive com a avaliação de consistência de possibilidades no espaço solução. O sistema deverá assistir o usuário na criação de gramáticas de formas originais em um processo muito similar ao processo tradicional de design definido por Knight (1998). O objetivo desta pesquisa é ultrapassar as limitações através da criação de gramáticas de formas de modo formal, constituindo uma ferramenta efetiva para o suporte de atividades de design para projetistas experientes ou não, assim como ferramenta pedagógica no ensino de design. Como objetivo secundário deste trabalho está o desenvolvimento de implementações computacionais que permitam a construção de gramáticas de formas, de modo formal e preciso.

Grande parte do conteúdo apresentado neste estudo foi desenvolvida no período entre março de 2007 e setembro de 2009, durante trabalho como bolsista de Ciência da Computação no Laboratório para Simulação e Modelagem em Arquitetura e Urbanismo (SIMMLAB) da Faculdade de Arquitetura da Universidade Federal do Rio Grande do Sul, contando com as supervisões do prof. Dr. Benamy Turkienicz e profª. Dra. Rosirene Mayer. Os

focos principais para o desenvolvimento do estudo foi a produção de estratégias generativas para arquitetura, urbanismo e design, surgindo então as implicações para o conceito de computabilidade.

2. MODELOS COMPUTACIONAIS

2.1. Modelos computacionais tradicionais

2.1.1. Máquina de Turing

A máquina de Turing é considerada o mais simples computador digital, onde a palavra digital indica computação sobre números definidos, sem qualquer sobreposição de estados da mecânica quântica (Berman, Doolen, Mainieri, & Tsifrinovich, 1998). Sugerida pelo matemático britânico Alan Turing, esta máquina é composta por uma fita dividida em quadros, um leitor de símbolos, um dispositivo para marcar ou apagar símbolos na fita e um mostrador para apresentar resultados. Durante a operação da máquina é possível marcar símbolos “X” ou “1” em quadros livres da fita, assim como apagar tais símbolos. Qualquer número positivo inteiro é escrito como uma seqüência de símbolos “1”, por exemplo, o número 5 corresponde a seqüência “11111”. O símbolo “X” indica onde inicia e termina um número, por exemplo, a seqüência “X1X X1X” representa duas instâncias do número “1”, em seqüência e disponíveis para operações como soma ou subtração. O programa específico para adição na máquina de Turing é representado pela tabela de transição a seguir:

		“X”	“1”
1	D6	E2	R1
2	R2	E3	?
3	R3	E4	E5
4	L4	?	R6
5	L5	?	R1
6	X6	!	R3

Esta tabela relaciona posições na fita e símbolos lidos para obter operações a efetuar. O símbolo “D” comanda a máquina para “escreva o dígito 1 na posição atual da fita”; o símbolo “X” significa “escreva X”; “E” significa “apague”; “R” significa “mova para a direita”; e finalmente “L” significa “mova para a esquerda”. Os números entre 1 e 6 após cada símbolo indicam qual o resultado deve ser apresentado no mostrador. O símbolo “?” representa um erro, enquanto o símbolo “!” representa o término da computação, comandando a máquina a parar.

A máquina de Turing tem os mesmos componentes que qualquer computador tem. O dispositivo para escrever e apagar símbolos corresponde à unidade aritmética, onde os cálculos são operados. A tabela de instruções corresponde à unidade de controle. Finalmente, a fita e o mostrador correspondem à unidade de memória.

Inúmeras máquinas podem inicialmente ser consideradas como tendo maior poder computacional que a máquina de Turing, porém seguindo um estudo mais aprofundado e formal a máquina de Turing mostra ter o mesmo poder computacional, não mais (Turing machine equivalents, 2009). Outras modelos de máquinas poderão computar mais rapidamente, com melhor uso de memória, ou com um conjunto de instruções menor, porém não com maior poder computacional.

Na segunda metade do Século XX foram feitas diversas tentativas para formalizar o conceito de computabilidade (Church–Turing thesis, 2008):

- O matemático americano Alonzo Church criou o cálculo-λ, um método para definir funções;
- O matemático britânico Alan Turing criou um modelo teórico de uma máquina capaz de operar cálculos sobre números;
- Church juntamente com o matemático Stephen Kleene e o estudioso de lógica J. B. Rosser definiram formalmente uma classe de funções cujos valores podem ser calculados por recursão.

Todos os três processos computacionais (recursão, o cálculo-λ e a máquina de Turing) são vistos como equivalentes – todas as três abordagens definem a mesma classe de funções. Isto levou matemáticos e cientistas da

computação a acreditar que o conceito de computabilidade é precisamente caracterizado por estes três processos equivalentes. Informalmente, a Hipótese de Church-Turing afirma que se existir algum método algorítmico para efetuar um cálculo, então o mesmo cálculo poderá ser obtido com uma máquina de Turing, assim como por funções recursivas ou pelo cálculo-λ. Resumidamente, “qualquer coisa computável é computável por uma máquina de Turing”.

A tese de Church e Turing é uma afirmação que caracteriza a natureza da computabilidade e não é possível ser formalmente provada. Mesmo com a prova de equivalência dos três processos descritos, a premissa fundamental para a tese – a noção de o que é uma função “efetivamente calculável” (computável) – é de certo modo vaga e intuitiva. Portanto a “tese” continua como uma “hipótese”.

Apesar do fato de não ter sido provada de modo formal, a tese de Church e Turing é universalmente aceita.

2.1.2. Gramática generativa de Chomsky

A gramática para uma linguagem pode ser vista como a teoria estrutural desta linguagem. Em lingüística teórica, uma gramática generativa refere-se à abordagem particular para o estudo da sintaxe. Uma gramática generativa para uma determinada linguagem especifica um conjunto de regras que irá produzir a correta combinação de palavras, formando sentenças gramaticais válidas naquela linguagem. Na maioria das abordagens para a definição de uma gramática generativa as regras irão também prever a morfologia de uma sentença, ou seja, a estrutura, a formação, a flexão e a classificação das palavras.

Gramáticas generativas têm origem no trabalho de Noam Chomsky (Chomsky, 1956), iniciado no final da década de 50. Versões preliminares da teoria de Chomsky foram denominadas gramáticas transformativas, e estes termos ainda são usados coletivamente, incluindo teorias subseqüentes a esta. Atualmente um número de versões distintas de gramáticas generativas é aplicado na lingüística. A teoria de Chomsky é conhecida como Programa

Minimalista. Entre outras teorias proeminentes estão: a gramática estrutural para expressões, a gramática generativa léxica, a gramática categorial, a gramática relacional e a gramática de três adjunções.

Chomsky afirma que muitas das propriedades de uma gramática generativa surgem de uma gramática universal “inata”. Proponentes de gramáticas generativas argumentam que a maioria das gramáticas não é resultado da função de comunicação e não é aprendida simplesmente pelo ambiente. Neste aspecto, a gramática generativa adota um ponto de vista diferenciado da gramática funcional cognitiva e de outras teorias comportamentais.

Gramáticas livres de contexto podem ser descritas e comparadas com o auxílio da hierarquia de Chomsky, composta por uma série de tipos de gramáticas formais, com poder de expressão crescente. Entre as gramáticas formais mais simples está a gramática regular, de tipo 3; Chomsky afirma que gramáticas regulares não são adequadas para modelar a linguagem humana, pois todas as linguagens humanas permitem a múltipla inclusão de segmentos dentro de outros segmentos.

Em nível mais alto de complexidade estão as gramáticas livres de contexto, de tipo 2. A derivação de uma sentença por uma gramática pode ser dissecada em uma árvore de derivação. Freqüentemente lingüistas trabalhando com gramáticas têm árvores de derivação como seu principal objeto de estudo. De acordo com este conceito, a sentença não é meramente uma seqüência de palavras, ao invés disso é uma árvore com ramos superiores e inferiores conectados através de nodos.

Quando a gramática generativa foi inicialmente proposta, foi amplamente aceita como a maneira possível para formalizar as regras implícitas que uma pessoa “sabe” quando ela conhece a linguagem e produz expressões nesta linguagem. Porém, Chomsky rejeita esta interpretação, afirmando que a gramática para uma linguagem determina o que uma pessoa deve saber de modo a reconhecer expressões, mas não uma hipótese sobre o processo envolvido para o entendimento e a produção de expressões naquela linguagem. Em qualquer caso a realidade é que sentenças produzidas por uma

estrutura gramatical poderão ser rejeitadas. Por exemplo, apesar de inclusões recursivas de termos serem permitidas em uma gramática, o limite aceitável é uma questão empírica e pode variar entre diferentes indivíduos, algo que não pode ser facilmente capturado em uma gramática formal. Conseqüentemente, a influência de gramáticas generativas em psicolingüística empírica tem declinado consideravelmente.

2.2. Gramática de formas

O processo para o desenvolvimento e uso de gramáticas de formas pode ser dividido em três etapas lógicas: (1) a criação e a modificação da gramática de formas, quando o projetista cria as regras e a forma inicial, para então verificar ou alterar restrições espaciais e lógicas; (2) compilar a gramática, quando a gramática em sua forma original é decomposta em elementos primitivos para a verificação sistemática de que cada regra tem um número finito de aplicações; e finalmente (3) a exploração da linguagem de design, produzindo desenhos, inserindo novas restrições, interrompendo o processo generativo, retrocedendo a estados prévios e armazenando os resultados obtidos (Tapia, 1999). O projetista poderá interpretar os desenhos resultantes e reutilizá-los como base para o desenvolvimento de novas gramáticas.

2.2.1. Linguagem e geometria

Linguagem é um sistema de símbolos para a codificação e decodificação de informações e é objeto de estudo de gramáticos. O processo de surgimento de uma linguagem é por muitas vezes desconhecido, em geral incremental e suscetível as mais diversas influências. Atualmente é conhecido um grande número de linguagens criadas artificialmente, devendo então haver a distinção entre linguagens naturais e linguagens artificiais. O termo “linguagem” pode ter diversas interpretações, como as linguagens para comunicação falada ou escrita, porém o termo aqui é aplicado especialmente para descrever linguagens estéticas, compostas por elementos geométricos.

Neste estudo o principal elemento tratado são as linguagens de design, ou vocabulário de design, servindo para determinar um esquema ou estilo estético que guia o projeto e configuração de elementos da geometria. Projetistas por vezes procuram encontrar e explorar uma mesma linguagem de design, percebida em objetos com forma e aparência consistente, nos quais são destacados aspectos como materiais, esquemas de cor, formas, padrões, texturas, etc. A partir da identidade de uma linguagem de design diferentes elementos podem ser relacionados e comparados entre si, apresentando similaridades e disparidades específicas.

A geometria é a matemática que trata de questões como medidas, formas, posições relativas e outras propriedades do espaço, sendo uma das ciências mais antigas da humanidade. Neste estudo a geometria serve como ferramenta matemática para a concepção e manipulação de conceitos diversos, especialmente na composição de formas e regras para derivação sobre formas. A geometria aqui utilizada é a geometria euclidiana, bidimensional ou tridimensional, porém algumas elaborações sobre elementos unidimensionais também estão presente. Alguns relacionamentos com os números naturais e com os números reais podem ser representados como pontos em uma reta, seja discreta ou contínua, e com este conceito alguns exemplos são apresentados.

Como simplificação dos elementos geométricos trabalhados, curvas regulares e irregulares são desconsideradas, sendo a grande maioria de elementos compostos exclusivamente por retas e segmentos de retas. Entretanto, alguns elementos extras são necessários, como pontos, marcadores, especialmente para a coordenação sobre operações de derivação e destaque visual para elementos de interesse. O procedimento para reconhecimento de formas e transformação sobre estas formas limita-se exclusivamente às formas compostas por segmentos de linhas simples.

2.2.2. Reconhecimento de padrões

Atualmente o reconhecimento de padrões conta com um vasto número de métodos para o desenvolvimento de aplicações em diferentes áreas de

atividade (Sá, 2001). A significância dos métodos e das técnicas para reconhecimento de padrões é percebida na emulação de tarefas “inteligentes”. Alguns exemplos de campos de atuação onde o reconhecimento de padrões é aplicado são: fabricação assistida por robôs, sistemas para diagnóstico médico, sistemas para previsão da economia, exploração de recursos naturais e análise de dados fornecidos por satélites. Devido a tais necessidades tem surgido um grande número de metodologias para situações específicas, ligadas a diversas outras disciplinas. Porém, contrário a esta tendência de dispersão, mais recentemente um novo desenvolvimento teórico tem unido muitos dos métodos clássicos para o reconhecimento de padrões, apresentando novas perspectivas.

O reconhecimento de padrões envolve modelos matemáticos de objetos descritos por suas capacidades e atributos. Noções de representação abstrata podem dar senso comum para as idéias de similaridade ou proximidade entre objetos. Os formalismos matemáticos, modelos e operações usadas dependem do tipo de problema a ser solucionado. Neste sentido, o reconhecimento de padrões é a “matemática posta em ação”. Reconhecimento de padrões é uma disciplina científica que trata métodos para descrição e classificação de objetos. Desde o princípio da computação o projeto de algoritmos que emulem a capacidade humana em descrever e classificar objetos tem sido uma tarefa desafiadora, sendo assim uma área fértil para a pesquisa, com múltiplos relacionamentos a outras disciplinas e envolvendo profissionais de diferentes áreas.

Exemplificando, as duas formas apresentadas na figura 1 são passíveis de reconhecimento sobre as três amostras da mesma figura. Observa-se que a primeira forma pode ser identificada na primeira e terceira amostra, porém não está presente na segunda amostra, pois o ângulo entre as linhas desta amostra é diferente. Similarmente, a segunda forma pode ser identificada apenas na amostra de número dois, pois na primeira e na terceira amostras não estão presentes os ângulos existentes na segunda forma. A terceira amostra apresenta maior complexidade, porém isto não invalida a existência de instâncias de cada uma das formas, entretanto maior custo computacional é necessário devido ao grande número de intersecções e possibilidades que devem ser investigadas.

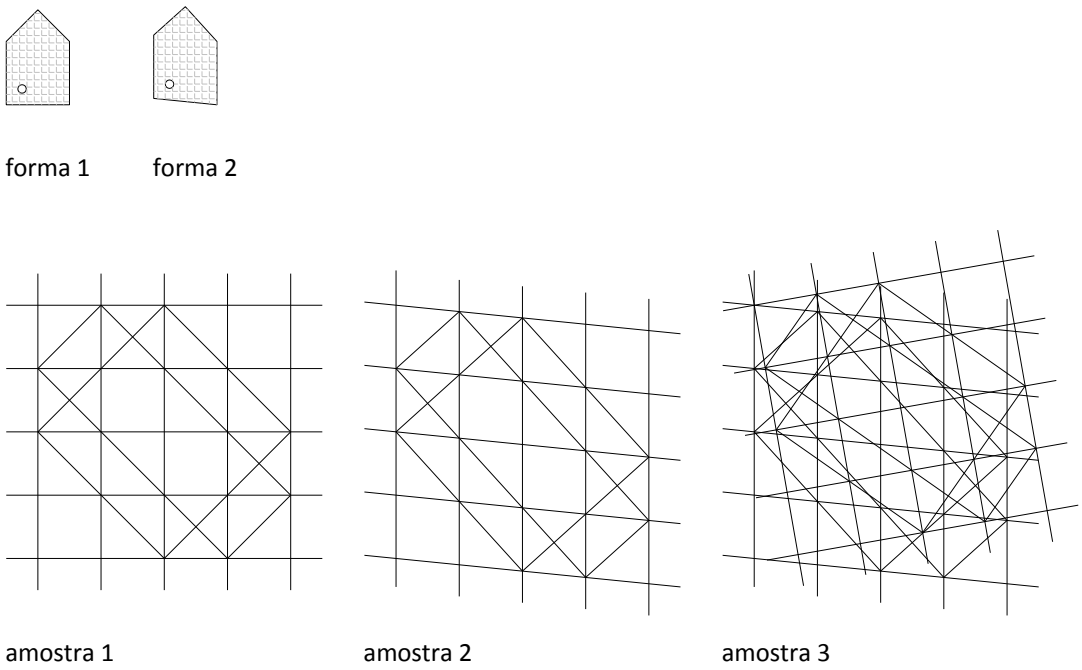


Figura 1: Exemplo para reconhecimento de padrão sobre formas e amostras distintas.

Na figura 1 as formas apresentadas incluem pequenos círculos servindo apenas como marcadores para identificação, não participando do processo de reconhecimento. O mesmo acontece com a textura, servindo somente para a identificação visual de cada instância das formas. Na figura 2 podemos observar o resultado do processo de reconhecimento de padrões, onde foram encontradas 30 instâncias e 60 instâncias da primeira forma na primeira e na terceira amostra. Cada instância reconhecida tem posição, rotação e escala, servindo esse conjunto de parâmetros para identificar unicamente cada instância. Na segunda amostra, como citado anteriormente, nenhuma instância da primeira forma foi encontrada.

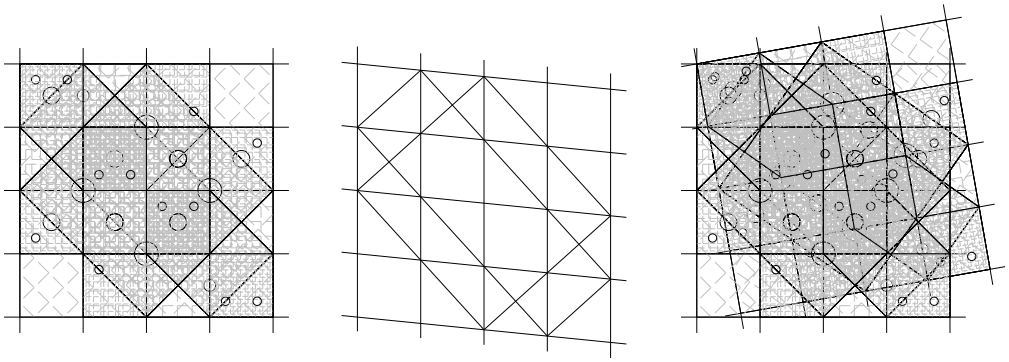


Figura 2: Resultado do processo de reconhecimento de padrões buscando a primeira forma sobre as três amostras apresentadas na figura 1.

O mesmo processo de reconhecimento de padrões, porém buscando a segunda forma sobre as três amostras, tem o resultado apresentado na figura 3, onde é possível observar exatamente 8 instâncias da segunda forma identificadas na segunda amostra. Na primeira e na terceira amostras não há nenhuma ocorrência desta segunda forma, pois por razão similar a descrita para a primeira forma na segunda amostra, os ângulos verificados na forma não estão presentes nestas amostras. Outro diferencial que pode ser destacado é o reduzido número de instâncias identificadas, exatamente por a segunda forma não possuir simetria – neste caso um menor número de possibilidades é possível.

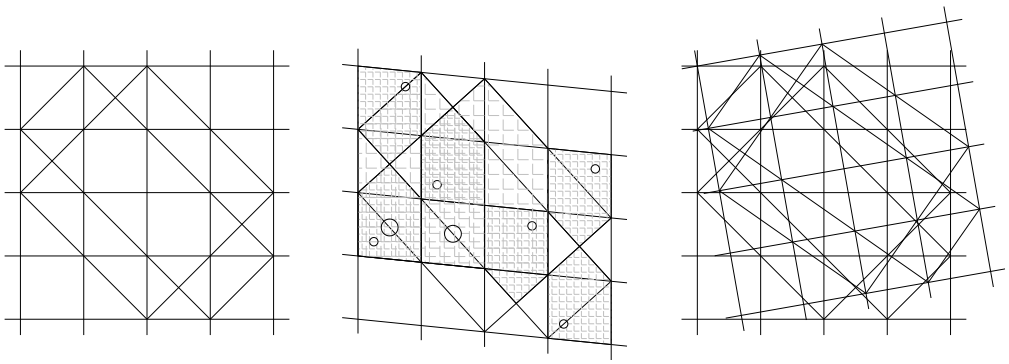


Figura 3: Resultado do processo de reconhecimento de padrões buscando a segunda forma sobre as três amostras apresentadas na figura 1.

O processo de reconhecimento sobre as amostras apresentadas na figura 1 é consideravelmente mais custoso para a terceira amostra, pois além de maior número de elementos e conseqüentemente maior número de intersecções, o processo para o reconhecimento é exaustivo, considerando

cada segmento da forma buscada como possivelmente cada segmento da amostra para a busca. O maior número de intersecções resulta em mais segmentações para cada linha presente, sendo este o maior ponto para o considerável aumento de complexidade computacional para o processo.

Este processo sistemático de reconhecimento de padrões é um passo necessário para a aplicação de gramáticas de formas, pois a partir do processo sistemático de reconhecimento das formas especificadas no lado esquerdo das regras são extraídos os parâmetros de instâncias onde tais regras são aplicáveis. Os exemplos apresentados na figura 2 e na figura 3 têm objetivo didático, exemplificando situações específicas para o processo de reconhecimento de padrões, porém esta sistemática pode inclusive ser aplicada a outras mídias, por exemplo, para o reconhecimento de padrões numéricos ou até harmônicas de áudio, bastando para tanto a devida representação de cada elemento.

Para a avaliação de similaridades entre diferentes elementos recorremos às capacidades e aos atributos de natureza distinta. No processo de discriminação podemos avaliar atributos distintos, adequadamente representados e relacionados. A tarefa de determinar uma classe para um elemento é chamada tarefa de classificação. Do ponto de vista matemático, é conveniente em tarefas de classificação representar padrões como vetores, relacionando as diversas características avaliadas. Considerando o processo de classificação automático, uma decisão razoável é obtida com a distância Euclidiana entre diferentes vetores.

Em geral o processo para a classificação de padrões pode ser impreciso por diversas razoes:

- Os atributos avaliados são inadequados ou insuficientes, ou seja, o problema de classificação provavelmente irá melhorar com o acréscimo de outros atributos;
- As amostras de padrão usadas para projetar o classificador não são suficientes, ou seja, se o objetivo é a discriminação de diferentes objetos, teremos que incluir uma amostra representativa de objetos, abrangendo toda a sua variância;

- O classificador não é suficientemente eficiente para separar classes, ou seja, a mensuração de distância entre vetores não é precisa ou o modelo usado não é apropriado;
- Existe uma intersecção intrínseca nas classes que não pode ser resolvida.

Diretamente relacionado com o processo de inferência existe outro tipo de tarefa, a regressão, que toma como base as experiências observadas anteriormente. Observamos este tipo de processo quando, por exemplo, animais iniciam seu processo migratório baseado em mudanças fisiológicas e seus ciclos biológicos internos. No cotidiano o processo de inferência é uma ferramenta importantíssima para orientar a decisão. Alguns exemplos bastante conhecidos são a manutenção da distância entre veículos em uma rodovia, a previsão de condições climáticas, a previsão de lucro em investimentos e a avaliação para concessão de empréstimos financeiros baseada em variáveis econômicas.

Tarefas de regressão podem ser operadas como uma tarefa de classificação. Podemos dividir o domínio de variáveis dependentes em intervalos suficientemente pequenos e interpretar a solução da regressão como uma solução de classificação, onde a classificação correta corresponde aos valores previstos dentro de uma faixa de classes. Neste sentido podemos considerar a seqüência de valores como um vetor de atributos, permitindo expressarmos a similaridade em termos de distância, agora relacionado aos valores previstos e desejados.

Algumas vezes as tarefas de regressão são também operadas como parte da classificação. Exemplificando, o reconhecimento de formas é por vezes um mero fator usado por arquitetos, dependendo de alguns atributos específicos, como geometria, dimensões e ângulos. Sistemas automáticos para reconhecimento de padrão procuram então regressar aos parâmetros avaliados manualmente por um especialista, de modo a estabelecer um ponto classificatório.

Em ambos os processos de classificação e de regressão a similaridade é a distância avaliada como uma quantidade numérica. Outro tipo de

similaridade relaciona-se com a estrutura de atributos dos objetos. De modo a solucionar a questão, os atributos individuais são descritos como uma seqüência de segmentos conectados sucessivamente. Estes segmentos podem então ser classificados de acordo com seus inter-relacionamentos.

2.2.3. Emergência de formas

A definição explícita de emergência de formas não é consenso de muitos autores, porém o termo é recorrente em diversos artigos e estudos publicados (Gero, Artificial Intelligence in Design '00, 2000) (Gero & Maher, 1992). Emergência em geral refere-se à identificação de formas e posterior derivação computacional através de regras gramaticais. Uma forma emergente não é uma forma pré-definida na gramática, mas sim uma forma que surge, ou é formada, das formas geradas através da aplicação de regras de derivação (Knight, 2003).

Uma gramática de formas é composta por um conjunto de regras para derivação sobre formas, juntamente com uma forma inicial para o início do processo computacional. Uma regra gramatical tem o formato $A \rightarrow B$, significando que uma forma A pode ser substituída por uma forma B. A regra $A \rightarrow B$ pode ser aplicada a forma C se a forma A ou qualquer forma geometricamente equivalente a A, com seus respectivos parâmetros para translação, rotação e escala, estiver contida na forma C. Em outras palavras, a forma A precisa coincidir com a forma C. Formalmente, a regra é aplicável se houver uma transformação t que faz com que a forma A seja parte da forma C. Se essa transformação t existir, então a regra pode ser aplicada a forma C transformando-a na forma C'. A regra é aplicada com a subtração da forma A da forma C, seguida da adição da forma B com a mesma transformação t. Assim a nova forma C' é igual a $[C - t(A)] + t(B)$. Uma computação sobre formas é uma seqüência de formas onde cada forma, com exceção da inicial, é gerada através de uma regra aplicada a forma anterior.

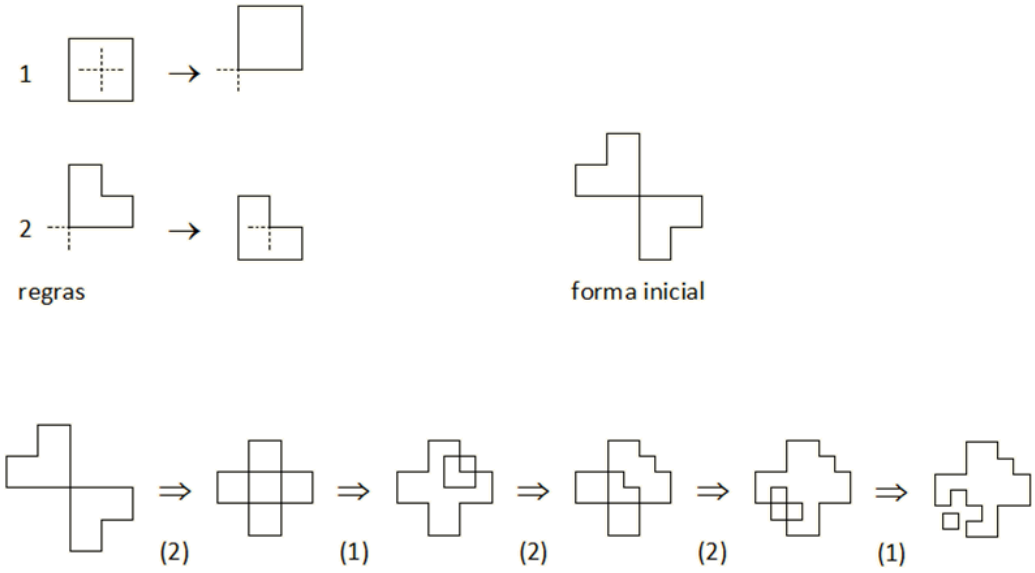


Figura 4: Uma gramática de formas composta por duas regras e uma forma inicial, e a conseqüente computação de formas através da aplicação de regras.

A figura 4 mostra uma gramática de formas composta por duas regras e uma forma inicial, seguida por uma possível computação de formas usando as regras. A primeira regra desloca o quadrado em seu eixo diagonal a distância equivalente a metade do lado do quadrado. A segunda regra desloca a forma *L* também em sua diagonal. Marcas de registro em cada regra mostram a posição das formas dos lados esquerdo e direito da regra relacionadas entre si. A forma inicial é composta por duas formas *L*, estando uma delas com rotação de 180 graus em relação à outra. As duas regras são aplicadas com o reconhecimento da forma quadrada ou da forma *L* do lado esquerdo de cada uma das regras com a forma inicial ou resultante de cada passo de computação. A forma quadrada ou a forma *L* em qualquer uma das regras pode sofrer translação, rotação ou mudança de escala, de modo a coincidir com a forma atual. Se um reconhecimento é detectado, a forma encontrada é então substituída conforme determina a regra. Para as regras desta gramática apresentada como exemplo, a direção para deslocamento conforme determinado pelas regras dependerá das transformações necessárias para o reconhecimento da instância da forma.

A computação apresentada na figura 4 é apenas um dois diversos resultados possíveis usando as duas regras propostas. Diferentes computações são possíveis devido à natureza não determinística das regras,

isto é, a aplicação exige escolhas. Em cada passo da computação existem três escolhas a serem feitas: (1) qual regra aplicar, (2) qual forma coincide com a forma ao lado esquerdo da regra e (3) qual a transformação necessária para o casamento da forma da esquerda da regra com a forma da computação presente. O utilizador das regras, seja humano ou mecânico, procede tais escolhas de modo a produzir uma computação específica. O grau de escolhas determinado pelas regras de uma gramática de formas pode em parte determinar o número de computações possíveis.

Uma forma emergente é uma forma, ou parte de uma forma, em uma computação qualquer que não foi inserida pela aplicação de uma regra em algum passo anterior da computação. Isto é, é uma forma que não é a mesma forma $t(B)$ inserida pela aplicação anterior de uma regra $A \rightarrow B$.

Formas em uma gramática de formas são compostas por elementos básicos de diferentes dimensões – pontos, linhas, planos, sólidos, ou uma combinação destes – unidos em um espaço de dimensão específica – linear, planar, tridimensional ou hiper-dimensional. Formas são compostas por elementos de determinada dimensão arranjadas no espaço de mesma dimensão, constituindo uma álgebra de formas. Por convenção, uma álgebra de formas é denotada por U_{ij}, onde i é a dimensão dos elementos básicos e j é a dimensão do espaço onde esses elementos são arranjados (Stiny, 1992). Por exemplo, a álgebra de formas U_{12} consiste de formas compostas por linhas no plano.

Por formas serem livremente decomponíveis, a emergência que pode ser reconhecida em qualquer passo da computação é possivelmente ilimitada em número ou tipo. Uma forma emergente pode ser a soma de formas inseridas por aplicações consecutivas de regras, ou pode ser parte de uma única forma inserida anteriormente.

Por exemplo, há um número infinito de formas emergentes em cada passo da computação apresentada na figura 4. As duas regras podem ser combinadas com qualquer dessas formas emergentes, tanto quadrados quanto formas *L*, em qualquer passo da computação com exceção do primeiro. A figura 5 indica os quadrados ou formas *L* nas quais uma regra é aplicada.

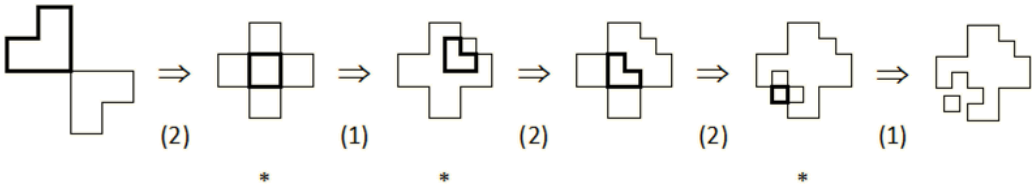


Figura 5: A computação apresentada na figura 4 com as formas onde as regras são aplicadas a cada passo destacadas por linhas espessas. Na linha inferior da figura asteriscos assinalam os passos onde há emergência de formas.

Ao contrário da emergência em gramáticas de formas, a emergência na maioria de outros sistemas computacionais é hierárquica, pois os elementos emergentes são uma composição de elementos primitivos daquele sistema, como padrões em cadeias de símbolos que são na verdade composições de símbolo individuais. A emergência surge através da combinação de primitivas em um nível para a formação de uma entidade emergente, com novas propriedades, no próximo nível. Um autômato celular é um bom exemplo para a emergência hierárquica, gerando padrões em uma grade de células. Um padrão é uma configuração de células em determinados estados, por exemplo, “ligado” e “desligado”. Iniciando com alguma configuração de células “ligadas” e “desligadas”, são aplicadas regras para a transformação de uma configuração em outra. A aplicação recursiva de regras pode gerar interessantes padrões emergentes e seqüências de padrões. Como formas compostas por pontos simples, padrões em um autômato celular são compostos por células. Assim, padrões emergentes são combinações de células – as primitivas de nível mais baixo no sistema. A emergência hierárquica é bem ilustrada no conhecido “*Game of Life*”, um autômato celular no qual as regras produzem padrões interessantes. Um dos mais conhecidos padrão no “*Game of Life*” é o “*glider*”, um padrão de cinco células que mudam ciclicamente aparentando estar em movimento sobre a grade de células. No nível mais simples, as células do autômato interagem para produzir um fenômeno mais complexo, de mais alto nível. Todos os padrões emergentes em um autômato celular têm este aspecto hierárquico, não sendo possível nenhum outro tipo de emergência.

Emergência na maioria dos sistemas computacionais restringe-se a geração e observação. O fenômeno da emergência é resultado de computações, mas não é usado como entrada para outras computações.

Assim, emergência ainda é hierárquica e computações com emergência são também hierárquicas. Entretanto, projetistas aplicam emergência de outros modos, sendo também possível para computação. Um processo de projeto típico é um tipo de computação informal no qual os projetistas transformam desenhos em outros desenhos pela adição e subtração de formas. Em cada passo desse processo o projetista pode reconhecer uma forma emergente inteiramente nova e não prevista, incluindo-a em transformações subseqüentes. O processo de projeto é contínuo, fluído e não hierárquico.

Computação com formas tem esta qualidade, com formas emergentes sendo geradas e reconhecidas de diferentes modos. Formas emergentes podem ser usadas em computações contínuas. Por exemplo, a regra apresentada na figura 6 transforma uma forma composta por um quadrado adicionando outro quadrado.

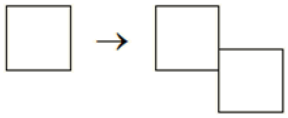


Figura 6: Regra simples para inserção de um quadrado junto a outro quadrado já existente.

Esta é uma computação – uma das muitas – usando a regra. A computação inicial com um quadrado simples:

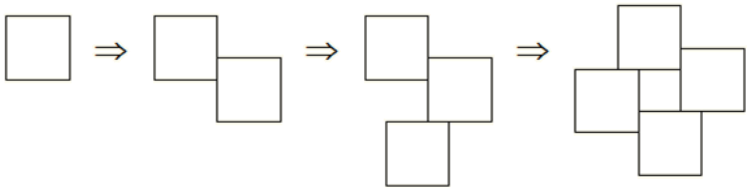


Figura 7: Computação para inserção de repetidos quadrados através da aplicação da regra apresentada na figura 6.

Após três passos da computação emerge outro quadrado, de menores dimensões. A regra pode ser aplicada a este novo quadrado continuando a computação, sem a troca para um diferente nível de computação:

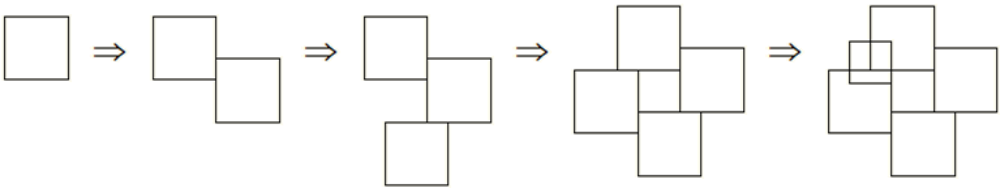


Figura 8: Mesma computação da figura 7 prosseguindo com derivação sobre forma emergente.

Emergência é comumente relacionada com imprevisibilidade e surpresa. É o fenômeno da emergência aparentemente espontânea e não explícita nas regras que torna a emergência computacional interessante e atraente para o projeto criativo. Porém, imprevisibilidade não é essencial para emergência em sistemas computacionais, inclusive para gramáticas de formas. Emergência em gramáticas de formas pode ser classificada como totalmente previsível a totalmente imprevisível. Diferentes aplicações de gramáticas de formas podem exigir diferentes graus de previsibilidade. Podem ser distinguidos três diferentes tipos de emergência em uma gramática de formas: antecipada, possível e não antecipada. Essas classes não são absolutas, sendo definidas apenas em relação ao conhecimento e a observação do autor da gramática.

Com emergência antecipada o autor da gramática cria regras e sabe onde certas formas irão emergir. Com a antecipação da emergência regras especificas podem ser criadas para operar sobre as formas. A emergência antecipada é crucial para a aplicação de gramáticas de formas em processos analíticos. Em um processo de análise uma gramática de formas é desenvolvida para caracterizar formas em uma linguagem ou estilo existente, sendo então aplicada a gramática sobre um conjunto expressivo da linguagem ou estilo em estudo, o chamado *corpus*. A gramática pode produzir elementos da linguagem, já existentes ou totalmente novos. A emergência é necessária em processos analíticos, mas deve ser antecipada, assim novos elementos serão produzidos somente em circunstâncias específicas. As regras deverão produzir todo e qualquer elemento de determinado estilo ou linguagem, exatamente isto e nada mais.

A figura 9 ilustra a emergência antecipada aplicando uma gramática composta por duas regras. A primeira regra concatena dois triângulos eqüiláteros e com algum conhecimento sobre triângulos eqüiláteros é fácil

antecipar que a aplicação recursiva da regra irá produzir um hexágono. A partir desta observação, uma segunda regra é incluída na gramática, uma regra para identificação de hexágonos. Iniciado uma computação a partir de um triangulo simples, após algumas derivações da primeira regra, o hexágono é criado. A segunda regra é então aplicada ao hexágono.

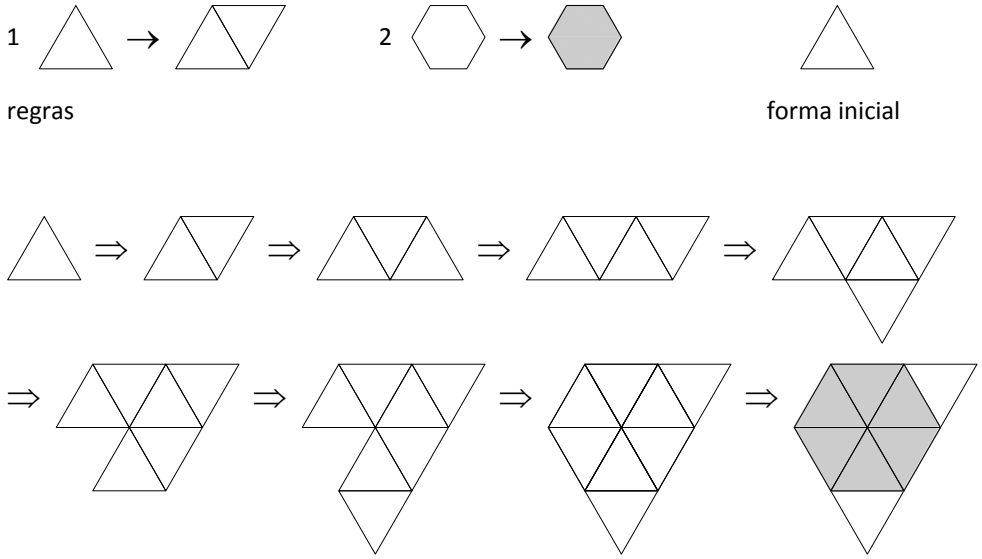


Figura 9: Regras, forma inicial e computação com o conceito de emergência antecipada.

Na emergência possível o autor da gramática define regras e supõe que certas formas irão emergir. Devido a extensão e a complexidade do sistema o autor não pode determinar – formalmente, experimentalmente ou de qualquer modo – quando e onde essas formas irão surgir. São então criadas regras especificas para a identificação das formas desejadas na ocorrência de emergência destas formas, servindo como auxiliares para a identificação precisa das formas emergentes desejadas.

A figura 10 ilustra a emergência possível em uma gramática de duas regras. A primeira regra sobrepõe um triângulo sobre outro triângulo existente. Alguma experimentação sobre esta regra revela a emergência de uma forma de peixe. A nova forma emerge com a computação de quatro passos apresentada. Sabendo com esta forma emerge a partir de triângulos, é simples predizer que um número infinito de formas semelhantes e de diferentes dimensões irá surgir com mais computações. Uma regra é incluída então na gramática para identificar a forma emergente. Esta segunda regra é aplicada

no quarto passo da computação. A primeira regra para triângulos gera uma espécie de malha de triângulos e desta malha diversas outras formas podem ser identificadas e exploradas.

Em aplicações analíticas a emergência pode não ser desejada, portanto a emergência não antecipada é determinante para o projeto de aplicações deste tipo. Este tipo de emergência é recorrente no processo convencional de desenho, particularmente no princípio, em estágios conceituais, quando a exploração livre de novas idéias de desenho é mais importante.

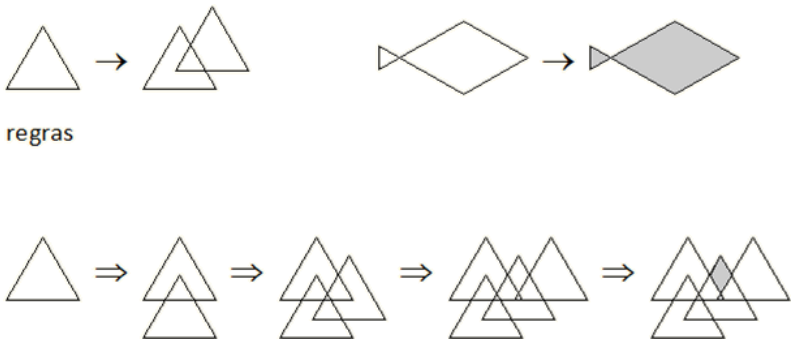


Figura 10: Regras e computação com o conceito de emergência possível.

Conforme a complexidade da gramática em estudo é possível antecipar a emergência de algumas formas, porém trabalhando com linguagens de design de maior complexidade ou com maior número de derivações possíveis esta exploração extensiva pode estar sujeita a erros de concepção e a dificuldade em prever a emergência de formas é crescente.

2.2.4. Teoria das categorias aplicada

O estudo de linguagens de design utiliza amplamente gramáticas de formas para a representação de elementos estéticos, possibilitando processos de análise e de síntese. A análise de elementos representativos de uma linguagem, o chamado *corpus*, requer a decomposição de derivações gramaticais complexas em composições de derivações de grau mais elementar, de modo a obter uma gramática eficiente na produção e verificação daquela linguagem. Já o processo de síntese explora o espaço solução

definidos por uma gramática gerando elementos da linguagem, especialmente considerando a emergência de formas, que amplia as possibilidades generativas da gramática definida.

A aplicação da teoria das categorias (Pierce, 1993) sobre o universo definido por uma gramática de formas busca a utilização das propriedades categoriais para o melhor entendimento do espaço solução e das possibilidades de derivação. Os algoritmos necessários para os processos analíticos sobre linguagens de design podem se beneficiar da propriedade transitiva, verificando relações de equivalência entre derivações complexas e composições de derivações elementares. O processo de síntese para uma gramática de formas definida deve aperfeiçoar a derivação de regras gramaticais, onde a identificação de ciclos na estrutura categorial e o melhor entendimento da emergência de formas podem melhorar substancialmente a eficiência dos processos. Este estudo define formalmente uma categoria correspondente a uma gramática de formas específica, detalhando componentes e a seguir verificando propriedades da teoria das categorias.

Linguagens de design têm inúmeras aplicações na formalização de conceitos de estética, porém são inúmeros os desafios no trato eficiente sobre elementos de maior subjetividade. A formalização de um modelo categorial que permita o estudo matemático e preciso de propriedades e relacionamentos, especialmente com a exploração de propriedades categoriais e suas possíveis aplicações no aprimoramento de algoritmos analíticos e de síntese, pode ter considerável impacto na eficiência dos processos. Com base nas observações deste modelo matemático, novos processos e novas possibilidades para algoritmos computacionais determinísticos ou estocásticos podem ser exploradas. Além disso, o estudo comparativo de modelos computacionais teóricos deve ter forte formalismo, de modo a definir relacionamentos e observar atributos comparativos.

3. GRAMÁTICA DE FORMAS

3.1. Ambiente, restrições e derivações

As formas em uma gramática de formas têm uma analogia direta em desenho: composição de linhas de mesma espessura sobre uma folha de papel; a superfície do papel contém formas em uma área fixa e delimitada. Entretanto também pode ser concebida uma gramática de formas com qualquer número de dimensões, por exemplo, três dimensões para representação de elementos espaciais tridimensionais.

Uma gramática de formas é definida por um conjunto de regras de transição, sendo cada uma das regras composta por duas formas, lado direito e lado esquerdo. O lado esquerdo é considerado para o reconhecimento de formas deriváveis, o lado direito é considerado para a possível derivação em determinada instância.

As partes de cada regra são definidas individualmente, porém estão relacionadas espacialmente e usualmente são representadas com uma marca de referência, representando o ponto origem do sistema. Representado como uma tupla, esse ponto de origem pode ser abstraído, pois computacionalmente não é representativo. Entretanto, durante um mesmo processo de derivações consecutivas, um único ponto de origem deve ser respeitado, pois diferentes ramos de derivações distintas podem ter conseqüências entre si.

O processo de derivação considera os parâmetros de determinada instância da forma representada no lado-esquerdo de uma regra gramatical, então esses parâmetros são aplicados ao lado-direito da mesma regra, produzindo uma alteração no ambiente. Cada conjunto de parâmetros,

representando uma instância de forma específica, é determinado pelo número de dimensões consideradas, sejam dimensões espaciais ou não.

3.2. Vocabulário, regras e derivações

Entre os elementos utilizados neste estudo destacam-se os vocabulários, regras de derivação e as derivações sobre formas, ou seja, a aplicação de regras definidas sobre componentes do vocabulário.

Sendo o vocabulário composto por elementos de linguagens de design, seus atributos são exclusivamente geométricos, entretanto conceitos e semânticas específicas podem ser considerados, neste caso estando explicitamente identificados.

3.2.1. Taxonomia

O estudo de gramáticas de formas é por si só bastante extenso e por essa razão é interessante sua classificação de acordo com características mais relevantes. Tratando com gramáticas de formas, as características observadas nos processos de derivação de regras gramaticais fornecem as características necessárias para classificação.

Cada regra de uma gramática de formas é composta por duas partes: o lado esquerdo da regra representa a forma original, aquela forma que deve ser identificadas inequivocamente de modo a possibilitar a aplicação de uma regra; e o lado direito da regra, representando o objetivo esperado do processo de derivação. Formalmente, o processo de derivação de regras de uma gramática de formas deve depois de identificar a instância de aplicação, substituir a forma do lado esquerdo da regra pela forma do lado direito da regra. Assim, este processo pode resultar na exclusão ou inclusão de elementos geométricos, conforme a diferença entre as formas esquerda e direita contidas na regra.

A figura 11 exemplifica três diferentes regras gramaticais que podem ser classificadas respectivamente como aditiva, subtrativa e mista. Regras aditivas

são aquelas em que há exclusivamente a inserção de elementos geométricos, na primeira regra apresentada na figura 11 a inserção de um novo quadrado deslocado em relação ao quadrado original dá esta característica à regra, sendo uma regra aditiva. Em regras subtrativas, como a segunda regra exemplificada, ocorre somente a subtração de elementos geométricos, como neste exemplo a remoção de um dos quadrados visto no lado esquerdo da regra, que no lado direito não está incluído. Finalmente, regras mistas são aquelas onde a derivação insere e remove elementos geométricos, apresentando então características de regras aditivas e subtrativas simultaneamente. De acordo esta classificação para regras individuais, uma gramática de formas completa é considerada aditiva ou subtrativa se for composta por regras aditivas ou subtrativas, respectivamente. Qualquer gramática composta tanto por regras aditivas quanto por regras subtrativas, ou mesmo por qualquer regra mista, é considerada uma gramática mista.

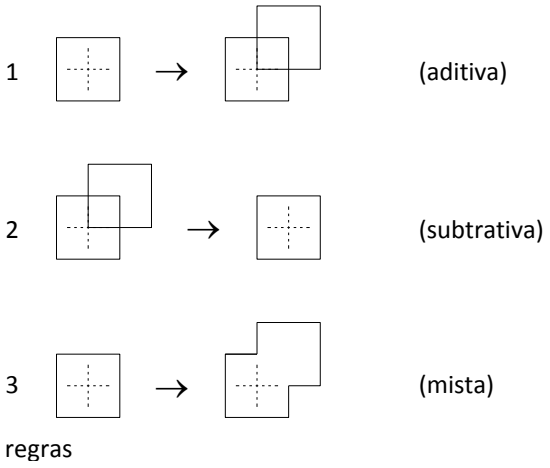


Figura 11: Regras gramaticais de tipo aditiva, subtrativa e mista.

Neste estudo, visando maior simplicidade dois exemplos e elementos apresentados, a grande maioria de regras e gramáticas é aditiva. Especialmente nas considerações sobre o universo definidos a partir de qualquer gramática, ou seja, a linguagem definida pela gramática, a inclusão de regras subtrativas ou mistas produz maior complexidade de entendimento, sendo assim evitada sempre que possível. No decorrer deste estudo são discutidos conceitos de linguagens ditas infinitas bem como características de recursividade sobre derivações gramaticais, onde o trato de gramáticas exclusivamente aditivas facilita enormemente o entendimento. Entretanto,

apesar da taxonomia proposta aqui, os conceitos estudados ainda tem validade para gramáticas de formas de qualquer tipo.

3.3. Ciclos e equivalência

O processo computacional para gramáticas de formas apresenta um desafio, o entendimento sobre derivações de natureza não determinística e o possivelmente espaço de soluções infinito. Mesmo com espaços de soluções finitos alguns algoritmos precisam tratar problemas inerentes como ciclos de derivações e a decomposição de regras gramaticais complexas em conjuntos equivalentes de regras mais simples. A teoria das categorias pode auxiliar a otimização dos processos algoritmos.

3.3.1. Estudo categorial

Este estudo procura desenvolver um modelo matemático de modo a associar gramáticas de formas e teoria das categorias. Esta abordagem pode permitir a utilização das propriedades categoriais de associatividade, transitividade e identidade para a otimização de algoritmos e redução de complexidade.

Uma gramática de formas $\boldsymbol{G}$ é formalmente definida como:

$$\boldsymbol{G} = \langle V_T, V_M, R, I \rangle$$

onde:

V_T	conjunto finito de formas terminais;
V_M	conjunto finito de formas não terminais (marcadores);
R	conjunto finito de regras na forma $u \to v$, onde u e v são formas;
I	forma inicial.

O processo de derivação de regras de qualquer gramática pode ser representado por relações matemáticas entre elementos de desenho. Instâncias de elementos de desenho, identificadas através de parâmetros de instância, tem entre si a relação derivacional, quando existe a possibilidade de derivação de regras da gramática da forma origem da relação produzindo a forma do destino da relação.

O estudo de gramáticas de formas apresenta alguns desafios como derivações não-deterministas e possivelmente um espaço de soluções infinito. Mesmo em espaços finitos, problemas como ciclos de derivações e a decomposição de regras gramáticas complexas em um conjunto equivalente de regras em composição. Neste campo, a Teoria das Categorias oferece as ferramentas.

Uma categoria computacional $\boldsymbol{SG}$, representando uma gramática de formas específica, é definida formalmente como:

$$\boldsymbol{SG} = \langle Obj_{\boldsymbol{SG}}, Mor_{\boldsymbol{SG}}, \delta_0, \delta_1, \iota, \circ \rangle$$

onde:

$Obj_{\boldsymbol{SG}}$	conjunto de objetos – desenhos resultado de possíveis derivações da gramática, determinando os objetos da categoria;
$Mor_{\boldsymbol{SG}}$	conjunto de morfismos – conjunto de todas as derivações n-árias da gramática, determinando os morfismos da categoria;
$\delta_0, \delta_1 : Mor_{\boldsymbol{SG}} \rightarrow Obj_{\boldsymbol{SG}}$	domínio e contra-domínio – mapeamento de origem e destino das derivações gramaticais, determinando o domínio e a imagem da categoria;
$\iota : Obj_{\boldsymbol{SG}} \rightarrow Mor_{\boldsymbol{SG}}$	função de identidade – representa uma derivação zero-ária, determinando a operação categorial de identidade, associando cada objeto a um morfismo $\iota_A : A \Rightarrow A$;

$\circ : (Mor_{\boldsymbol{SG}})^2 \rightarrow Mor_{\boldsymbol{SG}}$	função de composição – composição de regras gramaticais, determinando a função parcial de composição categorial, onde para a gramática $\langle \boldsymbol{ab}: A \rightarrow B, \boldsymbol{bc}: B \rightarrow C, \boldsymbol{ac}: A \rightarrow C \rangle$ a composição dos morfismos $\boldsymbol{ab}$ e $\boldsymbol{bc}$ é associada ao morfismo $\boldsymbol{ac}$.

Com a categoria formalmente definida, as propriedades de associatividade, transitividade e identidade poderão servir como ferramenta de trabalho no estudo, bem como para a definição formal de propriedades específicas para determinada gramática.

3.3.2. Aplicações

Como uma estrutura categorial, a categoria $\boldsymbol{SG}$ mantém todas as propriedades e conceitos da teoria das categorias, alguns deles muito úteis para a implementação e otimização de algoritmos computacionais. Definida para determinada gramática de formas $\boldsymbol{G}$, o comportamento da categoria $\boldsymbol{SG}$ é delimitado pelas derivações gramaticais e pela correspondente linguagem definida. Algumas observações podem ser feitas:

- Endomorfismos correspondem a derivações nulas, isto é, elementos de desenhos anteriores e posteriores idênticos e não há inserção ou remoção de formas – estas derivações normalmente não são incluídas em uma gramática bem formada, porém devem ser consideradas para formar uma categoria computacional respeitando a propriedade de identidade;
- Pode haver morfismos paralelos quando ocorrer redundância na sobreposição de elementos de desenho, porém em uma gramática bem formada não deverá haver equivalência direta entre regras gramaticais, mesmo considerando os parâmetros de transformação isométrica.

Nos demais exemplos e exploração de propriedades todas as gramáticas definidas são mantidas o mais simples possível, não sendo

utilizados marcadores e com as formas iniciais definidas diretamente nos exemplos da linguagem.

Uma das mais importantes propriedades sobre uma categorial é a equivalência, que pode identificar ciclos sobre as derivações, auxiliar os processos de composição, decomposição e transformação de regras gramaticais. Também é possível reduzir a redundância no espaço solução e identificar estados finais ou desejados no desenho.

Para exemplificar algumas propriedades categoriais úteis, três regras gramaticais são definidas na figura 12:

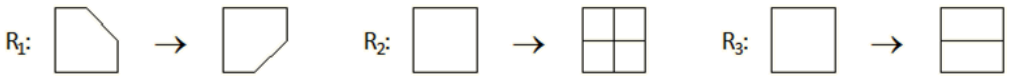


Figura 12: Regras gramaticais para a explanação sobre propriedades categoriais.

Nesta gramática simples, observa-se uma relação de equivalência com uma única regra gramatical, conforme apresentado na figura 13:

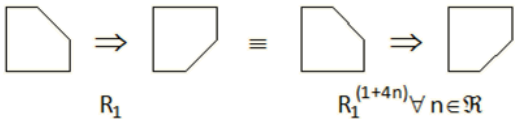


Figura 13: Relação de equivalência para uma regra gramatical única.

Também, observa-se que esta gramática não é mínima, ou seja, existe uma gramática equivalente mais simples. Considerando a relação de equivalência da decomposição (assim como a composição contrária) de uma regra complexa em regras simples, conforme apresentado na figura 14:

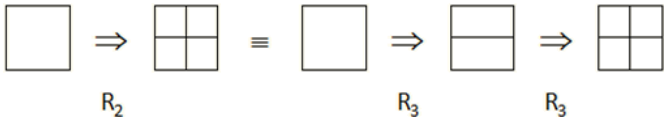


Figura 14: Relação de equivalência para duas regras gramaticais distintas.

Ainda a teoria generativa de (Leyton, 2001) apresenta os conceitos de "*transferability*" e "*recoverability*" permitindo computações de mais alto nível,

como explicação causal, inferência de estados, análise de simetria e percepção visual.

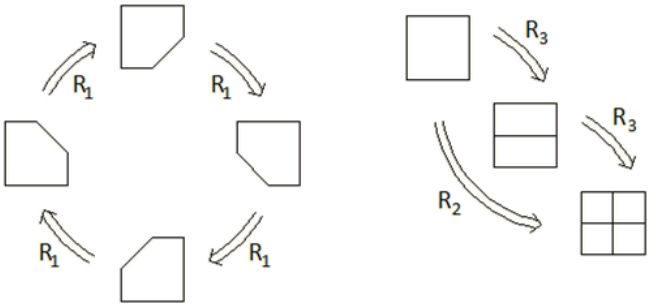


Figura 15: Ciclo de sobre derivações de regra única e relação de equivalência entre regras distintas.

A figura 16 apresenta o diagrama de uma categoria definida a partir de uma gramática composta de quatro regras a, b, c e d, definindo uma linguagem com 10 elementos de desenho distintos – o conjunto de objetos da categoria. Nesta categoria há um total de 24 derivações gramaticais diretas, 10 identidades ou morfismos nulos e 44 composições, totalizando 78 morfismos distintos.

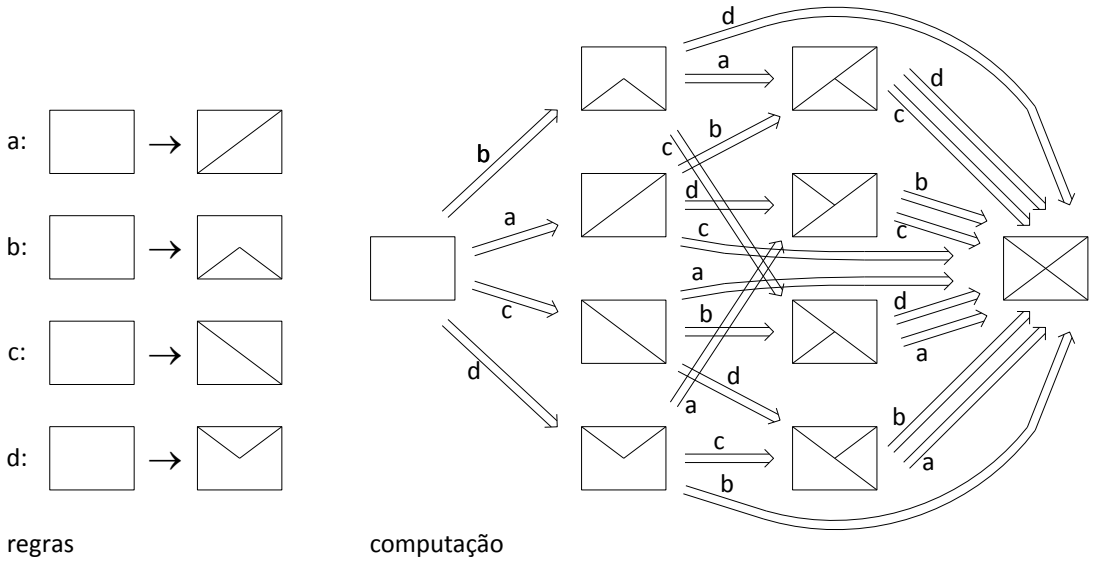
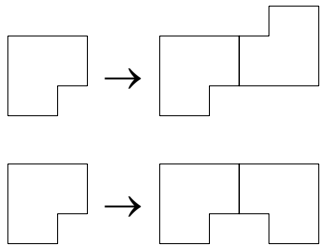


Figura 16: Exemplo de categoria computacional formada a partir de uma gramática de formas, com morfismos identidade omitidos.

Por razão de clareza os morfismos de identidade e são omitidos do diagrama da figura 16, assim como algumas composições. Nesta figura algumas relações de equivalência podem ser assinaladas nas composições de morfismos, entre elas podemos destacar $a^{\circ}b \equiv b^{\circ}a$ e $d^{\circ}b \equiv b^{\circ}d^{\circ}a$, porém

muitas outras existem. A categoria formada a partir desta gramática de formas é uma categoria *pequena*, pois os conjuntos de objetos e de morfismos são ambos finitos.



regras

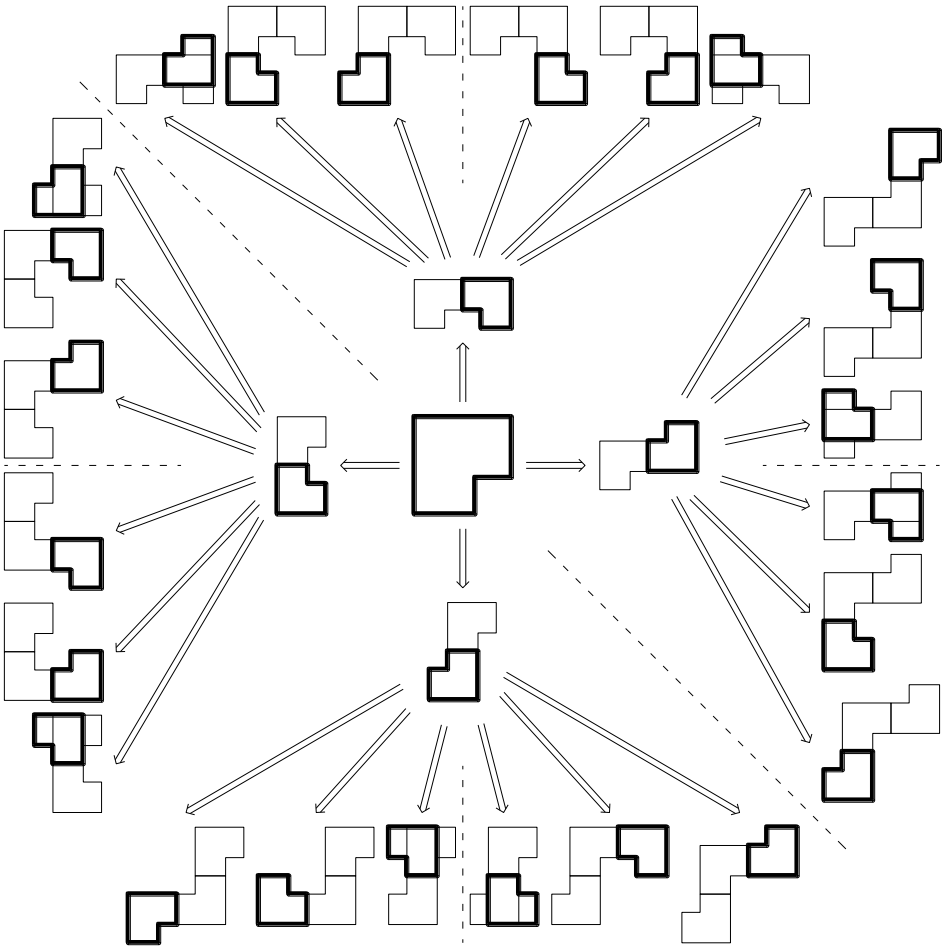


Figura 17: Regras de derivação e diagrama parcial da categoria correspondente, com morfismos identidade omitidos.

O diagrama parcial de uma categoria correspondente a uma gramática de formas é apresentada na figura 17, novamente omitindo os morfismos identidade. Esta gramática composta por apenas duas regras derivacionais produz uma linguagem de infinitos elementos de desenho, conseqüentemente

a categoria correspondente também tem infinitos objetos e morfismos. Muitas simetrias podem ser observadas, algumas delas destacadas no diagrama através de linhas tracejadas. Considerando a aplicação paramétrica de regras de derivação, essas simetrias podem ser decompostas global e localmente, em morfismos categorias correspondentes, o que pode ser extremamente útil para o completo entendimento do espaço solução da gramática.

Também se observa recursividade em algumas derivações compostas, mantendo similaridade geométrica entre os elementos de desenho, os objetos da categoria. A produção de formas pode ser cuidadosamente avaliada deste modo: a forma inicial após uma simples derivação possui semelhança estrutural com a forma produzida por uma longa seqüência de derivações. Os elementos de desenhos da figura 18, produzidos com as mesmas regras gramaticais aplicadas no diagrama da figura 17, destaca três elementos de desenho com similaridade estrutural que podem ser expandidos indefinidamente e ainda assim contidos em uma área finita.

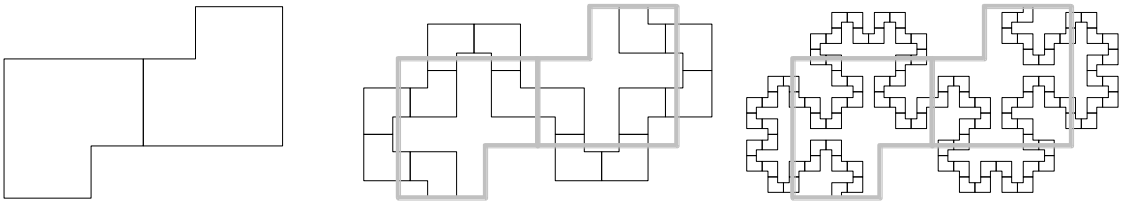


Figura 18: Elementos de desenho produzidos com as regras derivacionais da gramática apresentadas na figura 17 com similaridade estrutural.

Na avaliação da uma categoria definida a partir de uma gramática de formas, a emergência de formas deve ser considerada no surgimento de novas possibilidades de derivação. Quando existem novos morfismos a partir de novos elementos de desenho que não tenham sido inseridos explicitamente por uma regra de derivação – novos elementos do vocabulário são produzidos com a combinação de formas inseridas, de acordo com a linguagem específica da gramática de formas.

Na teoria das categorias objetos isomórficos são essencialmente os mesmos, portanto em qualquer categoria definida a partir de uma gramática de formas objetos isomórficos são aqueles para os quais existem morfismos de

um para outro, isto é, morfismos inversos. Qualquer gramática de formas com regras inversas irá apresentar isomorfismos na categoria correspondente, exatamente para aquelas regras de derivação.

4. IMPLEMENTAÇÕES COMPUTACIONAIS

Gramáticas de formas constituem importantes ferramentas para a análise e desenvolvimento de linguagens de design, porém são poucas as implementações computacionais disponíveis. Muitos têm sidos os avanços com aplicações práticas de gramáticas de formas, especialmente no meio acadêmico (Knight, 2000). O uso de computadores permite ultrapassar os limites relacionados ao número possível de alternativas formais e funcionais e de emergência de formas.

4.1. Implementação AutoLisp

Existem inúmeras implementações computacionais de gramáticas de formas, como GEdit (Tapia, 1999), Shaper 2D (McGill, 2001) e 3D Shaper (Wang & Duarte, 2001), porém com nenhuma delas é possível a construção de uma gramática genérica, sendo importante a possibilidade de produzir e aplicar gramáticas de qualquer natureza, especialmente considerando a emergência de formas nos processos para reconhecimento de padrão. Somente uma implementação com estas características pode servir de ferramenta auxiliar para o estudo de linguagens de design definidas através de gramáticas de formas. O processo de desenvolvimento foi orientado de modo a permitir a utilização do produto de software por profissionais de design experimentados ou não. A experimentação através da construção e aplicação de regras gramáticas, até a definição formal de uma linguagem de design foi considerada de especial interesse.

O primeiro projeto foi desenvolvido sobre a plataforma do AutoCAD utilizando a linguagem de programação AutoLisp, uma variante da linguagem

funcional Lisp. Aproveitando o ambiente do AutoCAD, um padrão acadêmico e da indústria, foram utilizadas as primitivas AutoCAD para elementos geométricos e simbólicos. O produto final é composto por uma biblioteca de funções onde foram encapsulados três comandos customizados, para a construção de formas, para a construção de regras gramaticais e finalmente para a aplicação de regras definidas sobre elementos do ambiente gráfico.

4.1.1. Arquitetura

A estrutura funcional desta primeira implementação é vista na figura 19 faz a ligação entre o usuário e os elementos do AutoCAD, permitindo ainda a construção de aplicações específicas de maior complexidade.

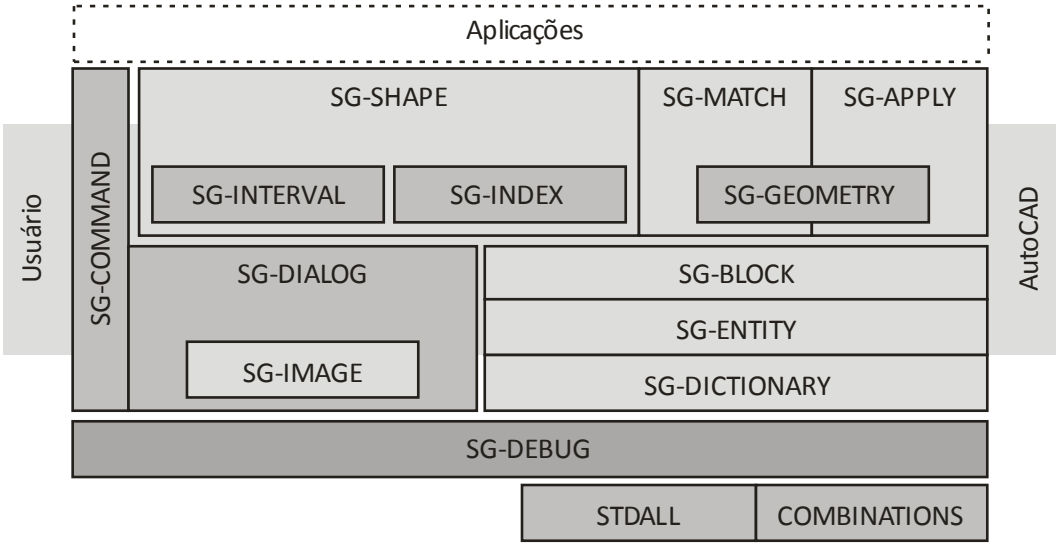


Figura 19: Estrutura arquitetural da primeira implementação, fornecendo uma ponte entre usuário e AutoCAD para a construção e aplicação de gramáticas de formas.

Os principais elementos componentes do sistema são:

SG-SHAPE ..... Compilação de elementos de desenho, produzindo estruturas de dados para representação simbólica e otimizadas para o processo de reconhecimento de padrões;

SG-INTERVAL ..... Implementação parcial de operações intervalares, largamente utilizadas na representação de formas através da parametrização e redução de dimensões sobre elementos gráficos;

SG-INDEX ..... Indexação hierárquica para estruturas de dados;

SG-MATCH ..... Algoritmo principal para o reconhecimento de padrões, oferecendo funções elementares para a identificação de formas contidas em outras formas, produzindo assim parâmetros para instâncias reconhecidas;

SG-APPLY ..... Operações para adição e subtração de formas, trabalhando sobre as estruturas de dados desenvolvidas e refletindo sobre as primitivas de desenho do ambiente;

SG-GEOMETRY ..... Rotinas auxiliares para operações geométricas como translação, rotação e escala;

SG-COMMAND ..... Comandos customizados do AutoCAD para a construção de formas, associação entre formas para a definição de regras gramaticais e posterior identificação, seleção e aplicação de regras construídas sobre elementos de desenho;

SG-DIALOG ..... Diálogos para interface com o usuário;

SG-IMAGE ..... Produção de representações visuais para formas compiladas;

SG-BLOCK ..... Tratamento de primitivas simbólicas do AutoCAD para a estruturação e manipulação

de formas definidas, utilizando a estruturas de blocos disponíveis no ambiente do AutoCAD;

SG-ENTITY ..... Tratamento de entidades do AutoCAD para primitivas de desenho, permitindo a inclusão de atributos de uso específico junto aos documentos em formato padrão;

SG-DICTIONARY ..... Rotinas auxiliares para a manipulação e tratamento de elementos simbólicos e descritivos, representando formas e regras nomeadas pelo usuário;

SG-DEBUG ..... Rotinas auxiliares para a instrumentação de código.

Durante o processo de desenvolvimento foram observados efeitos dos processos computacionais discretos. Assim os módulos para matemática intervalar, para indexação, para reconhecimento de padrões e para geometria incluíram tratamento numérico diferenciado. Foram aplicadas constantes para reduzir os efeitos da imprecisão numérica, na ordem de 10^{-6}. Especialmente no tratamento de múltiplas intersecções e ângulos essa liberdade numérica foi necessária para a aplicação eficiente do sistema. Para citar uma situação limite, matematicamente a equação $3 \times \frac{1}{3} = 1$, porém em sistemas de computador digital, devido a representação discreta de números, este resultado não é válido, podendo-se considerar corresponde a $3 \times \frac{1}{3} \approx 0{,}9999\ldots$, ou seja, uma periódica de valor obviamente diferente de 1. Deste modo a inclusão de salvaguardas para o tratamento dessas idiossincrasias foi necessário e mostrou-se eficiente, permitindo até a experimentação de atributos com maior grau de liberdade sobre o compromisso geométrico.

4.1.2. Compilador de formas

A figura 20 representa o processo de compilação de formas, gerando como resultado uma estrutura de dados otimizada para o reconhecimento de

padrão bem como operações sobre formas. Tendo como entrada um conjunto de primitivas geométricas do AutoCAD, os elementos são decompostos recursivamente, aplicando transformações isométricas sob componentes individuais. A partir de elementos mínimos e individuais são produzidas equações lineares correspondentes aos segmentos de linha, inseridas então em uma estrutura hierárquica indexada e finalmente com redução sobre elementos coincidentes aplicando operações de matemática intervalar. Nestes processos uma constante para liberdade numérica foi aplicada, reduzindo os efeitos colaterais inerentes da computação digital simbólica. O resultado deste bloco operacional é uma estrutura de dados canônica, eliminando qualquer incoerência produzida por possibilidades distintas da construção de elementos gráficos no ambiente AutoCAD.

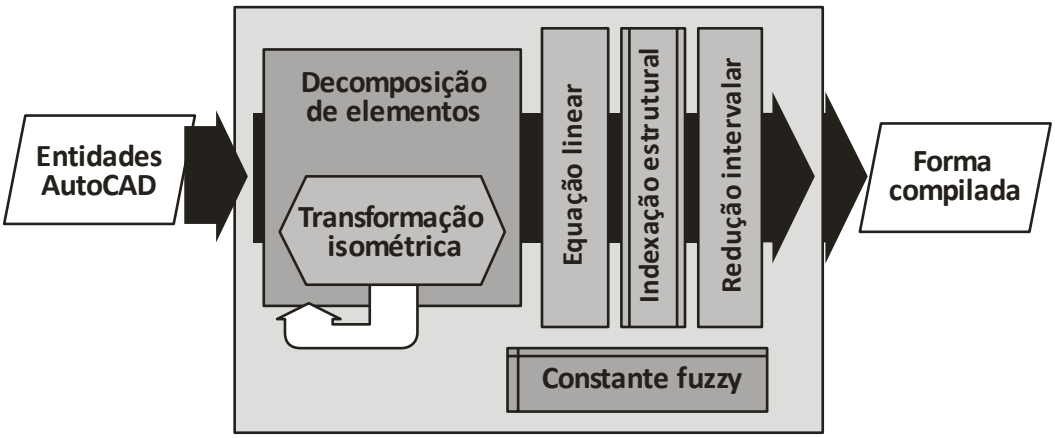


Figura 20: Detalhe do processo para a compilação de formas geométricas, obtendo uma estrutura de dados otimizada para o processo de reconhecimento de padrão bem como operações sobre formas.

4.1.3. Reconhecimento de padrões

O processo para reconhecimento de padrões, apresentado na figura 21 recebe como entrada duas formas previamente compiladas, retornando uma lista de parâmetros para instâncias identificadas. Este processo inicia com a verificação de intersecções de linhas, partindo então para a segmentação e permutação sobre pontos de intersecção, obtendo-se uma coleção de segmentos de linha. Este ponto é especialmente sensível a explosões combinatórias, estando aqui possivelmente o maior custo computacional de

acordo com as características das formas de entras. Aqui possivelmente estará o maior custo computacional, resultado de estruturas de dados para a representação de formas com maior profundidade e conseqüente maior número de intersecções e ângulos distintos. Finalmente, avaliando todo o conjunto de segmentos possíveis produzidos nos passos anteriores, cada segmento da forma objetivo é verifica repetidamente, extraindo quando houver casamento perfeito um vetor de parâmetros de translação, rotação e escala, para cada instância reconhecida.

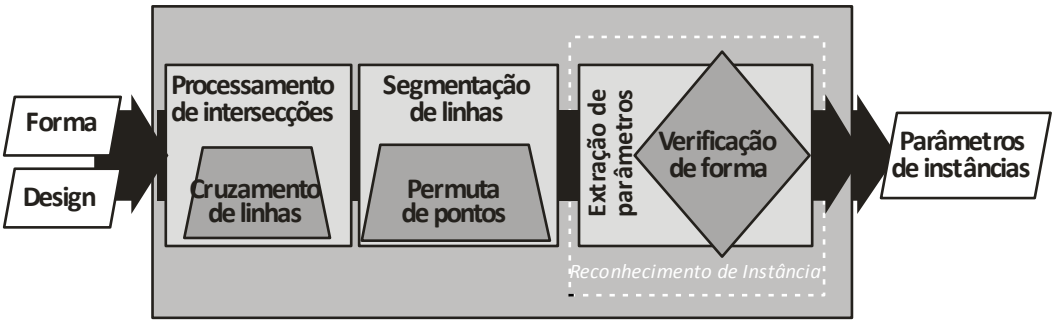


Figura 21: Bloco funcional para o processo de reconhecimento de padrões.

O gráfico na figura 22 compara o desempenho obtido no processo de reconhecimento de padrões sobre objetos de desenho com número crescentes segmentos. Devido à complexidade relacionada com a segmentação de formas, ou seja, o grau de intersecções entre primitivas gráficas distintas, duas amostras foram aplicadas na avaliação. Na primeira amostra, descrita como “pior caso” os elementos de desenho possuíam muitos pontos de intersecção e grande número de ângulos distintos, produzindo alta complexidade para a representação das formas e alta complexidade no processo de busca de padrões, especialmente devido a elevada segmentação e conseqüente explosão de combinações. Na segunda amostra foi reduzido o numero de intersecções de primitivas geométricas assim como a quantidade de ângulos distintos, produzindo estruturas de dados mais enxutas e eficientes. Considerando os dois extremos representados nas amostras distintas, o desempenho geral do algoritmo para reconhecimento de padrões mostrou-se razoável, permitindo a aplicação sobre elementos de alta complexidade e ainda assim mantendo a coerência e estabilidade de resultados.

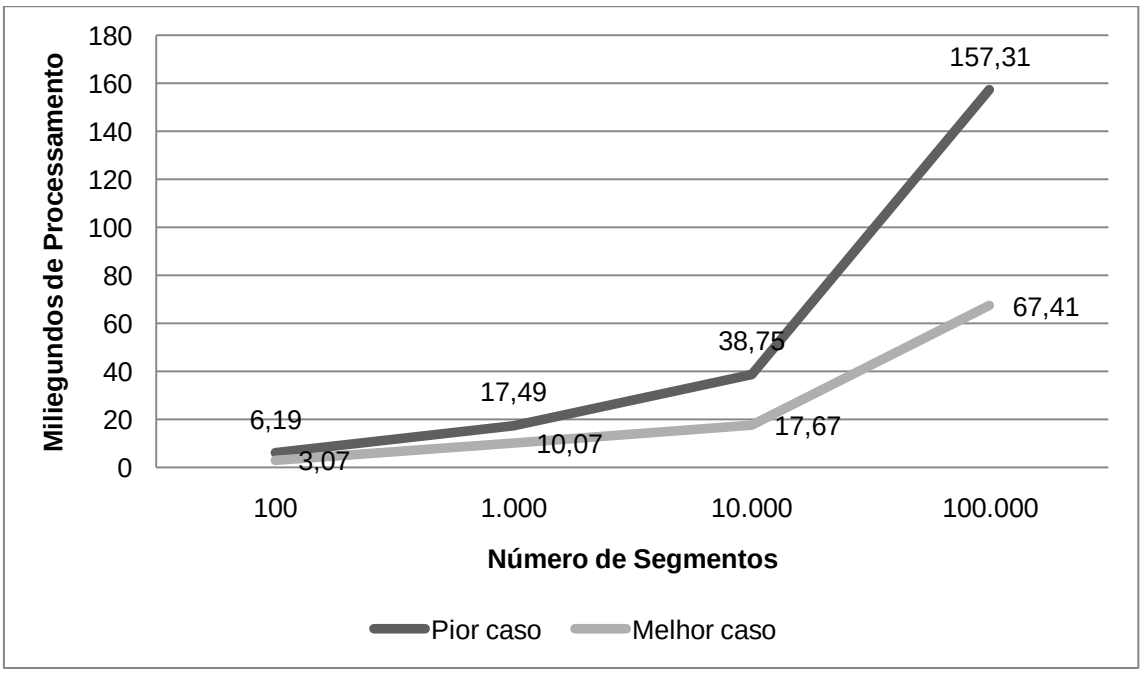


Figura 22: Gráfico comparativo do desempenho do algoritmo para reconhecimento de padrões.

4.1.4. Engenharia de software

Neste projeto inicial, devido à grande complexidade dos processos matemáticos, em sua maioria aplicando processos recursivos da linguagem funcional, uma bateria de casos de testes foi produzida. A metodologia de desenvolvimento orientado a testes (Beck, 2003) permitiu a verificação automática dos diversos componentes do sistema, dando assim maior garantia de correção e eficiência dos processos.

Precedendo o desenvolvimento de cada módulo um conjunto de testes é especificado, servindo também como contrato para cada função a ser desenvolvida e para a verificação automatizada durante o processo de desenvolvimento. A execução individual dos processos para teste de cada módulo aponta possíveis falhas, com este procedimento de teste podendo ser repetido durante o desenvolvimento do sistema e assim garantindo a correção dos módulos.

4.2. Implementação C#

A segunda implementação foi desenvolvida em C# e também sobre a plataforma do AutoCAD. Novamente as primitivas geométricas e simbólicas do ambiente foram exploradas, porém como principal diferencial, esta implementação trata com elementos tridimensionais. O objetivo final do projeto é fornecer um conjunto de ferramentas de trabalho que permitam a construção e aplicação de gramáticas de formas, operando sobre elementos do ambiente gráfico.

Devido à natureza da linguagem de programação escolhida e também dos recursos melhores para o desenvolvimento, foi possível especificar elementos para iteração com o usuário e com o ambiente de maior complexidade, especialmente no projeto de janelas destacáveis, janelas de sobreposição e nas funcionalidades dinâmicas para o ambiente.

4.2.1. Coordenadas de Plücker

Seguindo as mesmas estratégias desenvolvidas na primeira versão computacional desenvolvida, a representação de formas geométricas bem como o processo de reconhecimento de padrões tem como ponto base a redução dimensional dos elementos geométricos. A redução de elementos tridimensionais parte das representações originais no sistema de coordenadas x, y e z reduzindo cada elemento para uma representação canônica e inequívoca das formas, visando também o melhor desempenho nos processo de transformação e reconhecimento de padrões.

Em geometria, as coordenadas de Plücker são um modo de extensão sobre coordenadas homogêneas para vetores de ordem arbitrária (Stolfi, 1989), resultando em uma matemática bastante elegante. Apesar das principais aplicações de coordenadas de Plücker estarem presentes em estudos teóricos, a utilização para métodos computacionais em duas e três dimensões e bastante eficiente e relevante.

Introduzida no Século XIX por Julius Plücker, este sistema de coordenadas é um método de atribuição de seis coordenadas homogêneas para uma linha em espaço de projeção tridimensional, P^3. Este sistema surge naturalmente na geometria algébrica e tem sido demonstrado extremamente útil para a computação gráfica bem como para a cinemática usada para controle de mecanismos robóticos.

Uma linha $\boldsymbol{L}$ em três dimensões do espaço euclidiano é determinada por dois pontos distintos nela contidos, ou por dois planos distintos onde tal linha está contida. Considerando o primeiro caso, com pontos $\boldsymbol{a} = (a_x, a_y, a_z)$ e $\boldsymbol{b} = (b_x, b_y, b_z)$, o vetor de deslocamento de $\boldsymbol{a}$ para $\boldsymbol{b}$ é diferente de zero, pois são pontos distintos, e representa a direção da linha. Isto é, cada deslocamento entre pontos em $\boldsymbol{L}$ é um escalar múltiplo de $\boldsymbol{d} = \boldsymbol{b} - \boldsymbol{a}$. Se uma partícula física de uma unidade de massa mover-se de $\boldsymbol{a}$ para $\boldsymbol{b}$, esta partícula terá um momentum sobre a origem. O equivalente geométrico é um vetor com direção perpendicular ao plano contendo $\boldsymbol{L}$ e a origem e comprimento igual ao dobro da área do triângulo formado pelo deslocamento e pela origem. Tratando os pontos como deslocamentos a partir da origem, o momentum é $\boldsymbol{m} = \boldsymbol{a} \times \boldsymbol{b}$, onde "×" denota o produto vetorial. A área do triângulo é proporcional ao comprimento do segmento entre $\boldsymbol{a}$ e $\boldsymbol{b}$, considerado a base do triângulo; que não é alterada ao mover-se a base através da linha, paralela a ela mesma. Por definição o vetor momentum é perpendicular a qualquer deslocamento ao longo da linha, assim $\boldsymbol{d} \cdot \boldsymbol{m} = 0$, onde "·" denota o produto escalar.

Embora $\boldsymbol{d}$ ou $\boldsymbol{m}$ isoladamente não serem suficientes para determinar $\boldsymbol{L}$, juntos o par a define inequivocamente, para um múltiplo escalar diferente de zero que depende da distância entre $\boldsymbol{a}$ e $\boldsymbol{b}$. Isto é, as coordenadas $(\boldsymbol{d}:\boldsymbol{m}) = (d_x : d_y : d_z : m_x : m_y m_z)$ podem ser consideradas coordenadas homogêneas para $\boldsymbol{L}$, no sentido que todos os pares $(\lambda\boldsymbol{d}:\lambda\boldsymbol{m})$, para $\lambda \neq 0$, podem ser obtidos por pontos em $\boldsymbol{L}$ e somente em $\boldsymbol{L}$, e qualquer desses pares determina uma linha única desde que $\boldsymbol{d}$ seja diferente de zero e $\boldsymbol{d} \cdot \boldsymbol{m} = 0$.

4.2.2. Requisitos

Os requisitos funcionais e não funcionais para este sistema computacional são listados a seguir.

Requisitos funcionais:

a. Definir uma gramática de formas composta por regras de derivação no formato LHS (*left hand side*) e RHS (*right hand side*);
b. Reconhecer padrões geométricos extraindo parâmetros (posição, rotação e escala) de cada instância válida, considerando a emergência de formas;
c. Executar a derivação de uma regra da gramática para determinada instância com a extração e inserção de elementos gráficos – com reconhecimento de padrão, obtenção dos parâmetros de instância, extração e inserção de formas de acordo com a regra definida.

Requisitos não funcionais:

a. Considerar grau de tolerância para imprecisão numérica inerente à computação discreta (para dimensões lineares e angulares);
b. Inserir e remover elementos gráficos de modo ótimo, preservando elementos complexos sempre que possível;
c. Compilar formas geométricas em estruturas de dados eficientes para o algoritmo de reconhecimento;
d. Representar formas geométricas de forma canônica, sem ambigüidade para linhas coincidentes (representação gráfica maximal);
e. Considerar elementos simbólicos (marcadores) para regras de derivação;
f. Permitir definição de regras através de scripts utilizando meta-linguagem a ser definida;

g. Decompor recursivamente elementos gráficos complexos (inserção de blocos – elementos gráficos complexos compostos por outros elementos gráficos e dispostos segundo parâmetros de instâncias) em primitivas (pontos, linhas e curvas);
h. Estender algoritmo para derivação paralela, em três passos distintos: (1) reconhecimento de instâncias válidas, (2) remoção de elementos gráficos, e (3) inserção de elementos gráficos;
i. Representar de forma gráfica regras da gramática definida, utilizando janelas destacáveis padrão do AutoCAD;
j. Identificação dinâmica instâncias para aplicação das regras gramaticais facilitada, utilizando controles comuns do ambiente AutoCAD;
k. Projetar sistema em forma de plug-in para AutoCAD (versões 2004 e superiores);
l. Armazenar e recuperar gramáticas definidas utilizando formato padrão de documentos do AutoCAD.

4.2.3. Engenharia de software

Com métodos ágeis de desenvolvimento e utilizando Scrum (Schwaber, 2004), um *framework* iterativo incremental para gerenciamento de projetos é desenvolvimento ágil de *software*.

A seguir uma descrição das versões, *stories*, tarefas e testes planejados para este projeto:

Comando MSHAPE (Versão 0.8)	Compila um conjunto de entidades gráficas do AutoCAD referenciado por uma palavra-chave. Sintaxe: MSHAPE «shape-name» «selection-set» Estrutura de dados otimizada para operações (união, intersecção, diferença) entre *shapes* e para *matching* de *sub-shapes* em *shapes*; Elementos gráficos como inserção de blocos (INSERT) decompostos recursivamente considerando a transformação aplicada a cada instância; Representação geométrica canônica, sem ambigüidade para

linhas coincidentes;
Operações numéricas com grau de tolerância para imprecisão inerente da computação discreta.

Stories:

Implementação de formas	Implementar classe para formas da gramática. Compilador de formas para conjunto de elementos gráficos; Representação geométrica canônica e sem ambigüidade.
Matching de formas	Implementar *matching* de formas, encontrando *sub-shapes* em *shapes*. Estrutura de dados otimizada para algoritmo; Retorna conjunto de instâncias válidas (posição, rotação, escala).
Comando MSHAPE	Sintaxe: MSHAPE «shape-name» «selection-set» Nome da forma deve ser identificador válido; Seleção de elementos gráficos simples e compostos (INSERT).

Tasks:

Estrutura de dados para formas	Estrutura de dados para formas, otimizada para operações entre formas e para algoritmo de *matching*. Representação geométrica canônica e sem ambigüidade; Considera imprecisão numérica inerente ao modelo computacional discreto; Linhas retas, curvas, pontos, *labels*; Otimizada para operações entre formas e para algoritmo de *matching*.
Compilar formas de elementos gráficos	Compila um conjunto de elementos gráficos para uma forma, utilizando a estrutura de dados própria. Representação geométrica maximal - sem ambigüidade e sem redundância; Decompor recursivamente elementos gráficos compostos (INSERT) procedendo a transformação da instância.
Manipulação de formas	Implementar operações de translação, rotação e escala para formas compiladas.
Operações entre	Implementar operações de união,

formas	intersecção e diferença e predicado para intersecção.
Fracionamento de formas e segmentação	Implementar algoritmos para verificação de intersecções e gerar combinações de segmentação. Intersecções entre qualquer elemento; Combinação de segmentos entre extremos e intersecções de cada linha contínua; Cada combinação de segmento a-b e também b-a.
Verificação de instância válida	Verifica validade de instância para parâmetros de posição, rotação e escala. Obtido de cada combinação de segmento, para verificar possibilidades para cada linha-base; Transforma cada elemento da forma conforme parâmetros; Verifica passo a passo a existência de cada elemento transformado na forma principal; Encerra após verificação completa (T) ou verificar inexistência de elemento (F).
Algoritmo de *matching*	Procede inicialmente com marcadores simbólicos e a seguir com cada classe de elemento gráfico representado nas formas, retornando uma seqüência de instâncias válidas (posição, rotação, escala). Utiliza fracionamento e segmentação para verificar cada possível instância; Retorna um conjunto de instâncias válidas com posição, rotação e escala de cada uma.
Implementar comando MSHAPE	Implementar comando para AutoCAD.

Tests:

Compilador de formas	Verifica resultado do compilador de formas, compilando uma seqüência de formas e a seguir gerando as primitivas gráficas correspondentes, para então verificar a semelhança geométrica. Verificar especialmente ângulos múltiplos de 90; Verificar resultado de sobreposição de linhas e de linhas consecutivas; Verificar decomposição recursiva de elementos compostos (INSERT);

		Verificar decomposição de sólidos 3D entre outros; Verificar tratamento de pontos e de marcadores simbólicos.
	Desempenho do compilador de formas	Verifica complexidade computacional para uma seqüência de operações com crescente número de elementos gráficos e de intersecções.
	Manipulação de formas	Verifica manipulação de formas procedendo a uma seqüência até retroceder a forma original.
	Operações entre formas	Verificar operações de união, intersecção e diferença e verificar predicado de intersecção.
	Fracionamento de formas	Verifica fracionamento de formas em suas intersecções, para um conjunto variado de elementos gráficos. Intersecções em extremidades de linhas e outros; Intersecções entre elementos gráficos de classes diferentes.
	Segmentação de formas	Verifica a segmentação de elementos gráficos em pontos extremos e de intersecção.
	Verificação de instância válida	Verifica a existência e também a inexistência de instâncias válidas em um conjunto de pares de *sub-shape* e *shape*, efetuando a verificação de cada *sub-shape* em cada *shape*.
	Algoritmo de *matching*	Verifica o algoritmo de *matching* para um conjunto variado de pares de *sub-shape* e *shape*, efetuando a busca de cada *sub-shape* em cada *shape*.
	Desempenho do algoritmo de *matching*	Para um conjunto variado de *sub-shape* e *shape*, com crescente número de elementos gráficos e intersecções, verifica a complexidade computacional do processo de *matching* de formas.
	Usabilidade do comando MSHAPE	Verificar usabilidade do comando no ambiente do AutoCAD.
Comando MRULE (Versão 0.8)	Define uma regra para a gramática de formas relacionando dois *shapes* e uma transformação (opcional) para o segundo *shape*, referenciada por uma palavra-chave.	

<table>
<tr><td></td><td>Sintaxe: MRULE «rule-name» «shape-name$_1$» «shape-name$_2$» «position» «rotation» «scale»

- Regra definida no formato left-hand-side → right-hand-side;
- Lado esquerdo (LHS) representa a forma para matching, onde a regra é aplicável;
- Lado direito (RHS) representa a forma final, aplicada após a exclusão do shape do lado esquerdo;
- Transformação aplicada ao segundo shape (RHS), podendo ser nula.

Stories:<table><tr><td>Comando MRULE</td><td>Sintaxe: MRULE «rule-name» «shape-name» «shape-name» «position» «rotation» «scale»

Nome da regra deve ser um identificador válido;
Nomes das formas referenciam formas geradas com o comando MSHAPE;
Posição, rotação e escala são opcionais e utilizam facilidades do ambiente.</td></tr></table>
Tasks:<table><tr><td>Implementar comando MRULE</td><td>Implementar comando para AutoCAD.</td></tr></table>
Tests:<table><tr><td>Usabilidade do comando MRULE</td><td>Verificar usabilidade do comando no ambiente do AutoCAD.</td></tr></table></td></tr>
<tr><td>Comando SGRAM

(Versão 0.8)</td><td>Aplica regra da gramática de formas, encontrando instâncias válidas e procedendo a regra definida para.

Sintaxe: SGRAM «rule-name» «selection-set»

- LHS da regra usado para matching na forma correspondente aos elementos gráficos selecionados;
- Instâncias válidas destacadas para seleção pelo usuário;
- Aplicação regra da gramática para instância selecionada, com remoção do LHS e inserção do RHS.

Stories:<table><tr><td>Comando SGRAM</td><td>Sintaxe: SGRAM «rule-name» «selection-set»

Nome da regra referencia regra da</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>gramática gerada com o comando MRULE;
Seleção de elementos gráficos simples e compostos (INSERT).</td></tr>
<tr><td>Inserção e remoção de formas</td><td>Implementar inserção e remoção de formas nos elementos gráficos do AutoCAD.
Inserção simples conforme representação gráfica da forma;
Remoção sobre um conjunto dado de elementos gráficos, possivelmente todo o documento;
Remoção mínima, para elementos gráficos removidos parcialmente.</td></tr>
</table>

Tasks:
<table>
<tr><td>Inserção de formas</td><td>Inserir formas como elementos gráficos.</td></tr>
<tr><td>Remoção de formas</td><td>Remover elementos gráficos para determinada forma e conjunto de elementos gráficos iniciais.
Remoção mínima, para elementos gráficos removidos parcialmente;
Processa sobre um conjunto de elementos gráficos dado, desconsiderando restante.</td></tr>
<tr><td>Implementar comando SGRAM</td><td>Implementar comando para AutoCAD.</td></tr>
</table>

Tests:
<table>
<tr><td>Inserção e remoção de formas</td><td>Verifica operações de inserção e remoção de formas para um conjunto inicial de elementos gráficos.
Insere uma série de formas e a seguir remove parte a parte;
Verificar remoção mínima de elementos parcialmente removidos.</td></tr>
<tr><td>Usabilidade do comando SGRAM</td><td>Verificar usabilidade do comando no ambiente do AutoCAD.</td></tr>
</table>
</td></tr>
<tr><td>Diálogos para definição de gramática

(Versão 1.0)</td><td>Diálogos customizados para os comandos MSHAPE e MRULE.

Stories:
<table>
<tr><td>Diálogo para formas da gramática</td><td>Diálogo para definição de formas para a gramática.
Seleção de elementos gráficos para compor a forma;
Definição de posição, rotação e escala;</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>Auxiliares para definição de bordas.</td></tr>
<tr><td>Diálogo para regras da gramática</td><td>Diálogo para definição de regras da gramática de formas.
Seleção de formas para LHS e RSH;
Parâmetros de posição, rotação e escala para a forma da direita;
Visualização dinâmica da regra definida.</td></tr>
</table>
</td></tr>
<tr><td>Diálogo para aplicação de gramática

(Versão 1.0)</td><td>Diálogo para comando SGRAM com identificação e seleção de instâncias válidas.

Stories:
<table>
<tr><td>Diálogo para aplicação da gramática</td><td>Diálogo para aplicação de regras da gramática de formas.
Seleção de regra para aplicação;
Visualização da regra escolhida;
Seleção de elementos gráficos para aplicação da regra;
Identificação e escolha de instâncias válidas;
Remoção (LSH) e inserção (RHS) de elementos gráficos.</td></tr>
</table>
</td></tr>
<tr><td>(Versão 1.2)</td><td>Visualização e aplicação da gramática de formas utilizando janelas destacáveis e identificação dinâmica de instâncias válidas.

Stories:
<table>
<tr><td>Janela destacável para gramática</td><td>Janela destacável para visualização e seleção de regras da gramática de formas.
Seleção de regras individuais ou em conjunto;
Botões para operações sobre formas, regras.</td></tr>
<tr><td>Identificação dinâmica de instâncias</td><td>Identifica dinamicamente instâncias válidas para derivação conforme seleção de regras da gramática na janela destacável.
Identifica instâncias válidas com o hover do mouse.</td></tr>
</table>
</td></tr>
<tr><td>(Versão 1.4)</td><td>Definir regras para a gramática de formas utilizando scripts, definidos pelo usuário.

Stories:
<table>
<tr><td>Scripts para regras da gramática</td><td>Definir meta-linguagem para definir regras especiais para a gramática de formas.
No LHS, para o matching, o script deve utilizar predicados geométricos (square, triangle, polygon) para identificar instâncias</td></tr>
</table>
</td></tr>
</table>

			válidas para a regra; No *matching* é feito o *binding* de pontos e outros parâmetros da instância identificada, conforme os predicados utilizados (triangle(A,B,C)); No RHS o script determina operações de remoção e de inserção de primitivas gráficas, utilizando o *binding* efetuado durante o *matching*.

5. APLICAÇÕES E COMPUTABILIDADE

Os processos envolvendo gramáticas de formas têm maior utilidade conforme os ambientes de aplicação, havendo maior relevância para a manipulação de linguagens de design e para o desenvolvimento de processos cognitivos de elaboração e estudo sobre elementos estéticos. Neste capítulo são apresentadas algumas aplicações práticas de gramáticas de formas e alguns problemas específicos de menor relevância, servindo para a comparação entre diferentes modelos computacionais. São também apresentados alguns conceitos percebidos em gramáticas de formas, para a posterior comparação quanto ao conceito de computabilidade entre modelos computacionais simbólicos e não simbólicos.

5.1. Aplicações práticas

5.1.1. Processos de síntese e analíticos

As aplicações mais comuns para uma gramática de formas são para análise e síntese envolvendo linguagens de design, porém, objetivando a comparação com outros modelos computacionais, é possivel conceber um processo derivacional para produção de resultados equivalente a processos algébricos.

O processo de síntese de uma linguagem de design parte de uma gramática de formas definida, incluindo os elementos de vocabulário que compões as regras derivacionais, para a produção de elementos de desenho partencentes àquelas linguagem. Em geral o desenvolvimento de gramáticas

de formas tem o objetivo de produzir uma linguagem de design específica que satisfaça determinadas condições, estéticas ou funcionais. O processo de desenvolvimento envolve o projeto da gramática inicial seguido pela síntese e avaliação dos elementos da linguagem definida. Este processo pode ser repetido indefinidamente, com ajustes sobre as regras derivacionais com o intuito de satisfazer os objetivos propostos. Uma ferramenta computacional se faz necessária de modo a agilizar este processo, por vezes muito custoso e sujeito a erros.

Processos analíticos de linguagens de desing conhecidas procuram produzir uma gramática suficientemente expressiva que permita a produção e validação daquela linguagem. Concorrentemente, a gramática produzida deve ser sucinta e definir de modo fiel e restrito a linguagem de design em estudo. Este processo ainda não foi totalmente desenvolvido, sendo necessário um conjunto de elementos representativos da linguagem de desing, o chamado *corpus*. Inicialmente é definido um elemento de desenho inicial, de certo modo intuitivamente e de acordo com os elementos disponíveis para análise. A seguir são definidas regras individuais que permitam a produção de cada elemento do *corpus*. Considerando este conjunto inicial de regras são verificadas similaridades tanto de componentes dos elementos do vocabulário quanto das próprias regras de derivação, procedendo a decomposição de regras complexas em conjuntos de regras de menor complexidade, buscando a redução do número total e variações deste conjunto de regras. Este processo não é totalmente formalizado, sendo em geral intuitivo e experimental, explorando especialmente similaridades e simetrias de rotação e de escala, sempre considerando a parametrização dos passos de derivação. A gramática final produzida a partir desse processo analítico deverá ser expressiva e ainda restringir com a maior fidelidade possível a linguagem de design sob estudo.

5.1.2. Computação algébrica

Buscando maior semelhança com outros modelos computacionais um exemplo de computação algébrica utilizando gramáticas de formas é aqui apresentado. Considerando os elementos envolvidos, a gramática proposta é

basicamente linear, ou seja, produz elementos em uma reta, compondo linhas auxiliares apenas com a intenção de coordenar o processo derivacional.

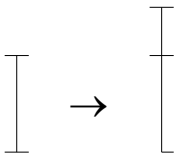
Considerando uma série geométrica de forma geral:

$$\sum_{n=0}^{\infty} z^n$$

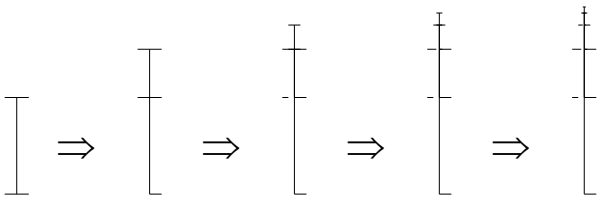
que é convergente para $|z| < 1$, se instanciada para $z = \frac{1}{2}$ corresponde a série:

$$\sum_{n=0}^{\infty} \frac{1}{2^n} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

pode ser computada com uma gramática de formas composta por uma única regra de derivação, conforme apresentado na figura 23. A regra de derivação produz um elemento com exatamente a relação de proporção $1{:}\frac{3}{2}$, ou seja, cada passo derivacional estende o elemento de desenho com uma proporção exata, correspondendo a cada termo da série geométrica a ser computada. O resultado parcial corresponde a extensão vertical do elemento de desenho produzido. Se considerados infinitos passos de derivação a séria irá convergir, exatamente como no processo algébrico.



regra



computação

Figura 23: Gramática de formas e respectiva computação de uma série geométrica correspondente.

Processos semelhantes podem ser criados para séries harmônicas, alternadas e telescópicas, porém para tanto torna-se necessário a utilização de regras parametrizadas. Do mesmo modo, por curiosidade matemática e para ilustrar o conceito, é possível a computação de números irracionais como π e funções da trigonometria ou qualquer outra série numérica. Com os tipos de elementos adequados é possível a computação de funções sobre números complexos ou até hiperdimensionais.

Evidentemente não há argumento válido para computações algébricas através da aplicação de gramáticas de formas, porém o conceito é válido para ilustrar processos equivalentes e possibilidades a explorar.

5.2. Poder computacional comparado

Para a ciência da computação um modelo computacional é um modelo matemático que através de recursos computacionais permite a simulação de um sistema complexo. Podemos considerar os modelos computacionais teóricos clássicos como modelos matemáticos idealizados de dispositivos físicos hipotéticos. Para um dispositivo físico que a partir de entradas específicas resulta em saídas específicas, o correspondente modelo computacional pode ser visto como uma “caixa preta”, computando um determinado conjunto de funções parciais (Boker & Dershowitz, 2005).

Como modelos computacionais clássicos, a gramática generativa de Chomsky e a máquina de Turing são consideradas com poder computacional equivalentes, ou seja, o que pode ser computado com um pode ser computado por outro. Porém, isto não significa que o mesmo custo em recursos computacional é exigido por diferentes modelos, por exemplo, a quantidade de memória dispensada ou o tempo de processamento necessário podem diferir. Na mesma classe de computabilidade encontra-se a gramática de formas, todavia o processo em que as computações são efetuadas é consideravelmente mais eficiente para determinados tipos de problemas. Por exemplo, no desenvolvimento de linguagens de design a gramática de formas tem considerável eficiência, pois resumo em poucos elementos e de fácil

entendimento todos os elementos necessários para produzir, avaliar e validar elementos do universo definido pela gramática.

Em linguagens de design os processos analíticos e de síntese têm especial interesse, pois o conjunto de regras de derivação que forma uma gramática define de modo preciso os elementos contidos na linguagem. Para uma gramática de formas os elementos da linguagem são compostos por geometrias, seja qual for o número de dimensões, e deste modo as regras de derivação são também compostas com o mesmo numero de dimensões. O processo de análise de linguagens de design procura formular a própria gramática através da análise de elementos representativos da linguagem, o chamado *corpus*. Paralelamente, os processos de síntese aplicam regras específicas de uma gramática de formas para a produção de elementos geométricos que são cada um deles parte da linguagem sob estudo.

Especialmente a característica da emergência de formas, própria das gramáticas de formas, tem grande relevância para os processos envolvendo linguagens de design, pois a evolução de elementos da linguagem pode não ser linear, resultando em grande complexidade para a simulação através de um sistema computacional tradicional. Especificamente tratando de elementos de geometria, uma comparação que é valida e pode servir como comparação de domínios é a comparação entre operações de transformação linear (translação, rotação e escala) com elementos fractais – elementos fractais podem ate certo ponto serem produzidos através de transformações lineares tradicionais, porém com custo consideravelmente maior e conseqüente maior complexidade e dificuldade de controle. Da mesma forma, uma gramática de formas pode sintetizar de modo extremamente resumido toda uma linguagem de design. Nos casos especiais em que houver grande incidência de emergência e submergência de formas a equivalência para modelos computacionais tradicionais requer complexidade em escala, pois apesar da possível simulação em qualquer modelo computacional, a complexidade exigida é crescente.

5.3. Emergência e submergência de formas

De acordo com a taxonomia proposta na seção 3.2.1, que classifica regras gramaticais aditivas, subtrativas e mistas, a emergência de formas só poderá estar presente quando houver inserção de elementos no desenho, ou seja, em regras aditivas ou mistas. Ao contrário disto, a submergência de formas somente poderá ocorrer com regras do tipo subtrativas ou mistas, ou seja, quando houver remoção de elementos no desenho.

5.3.1. Emergência de formas

O processo derivacional para qualquer gramática é iniciado com o reconhecimento do padrão explicitado no componente do lado esquerdo da regra (LHS), obtendo assim os parâmetros que identificam unicamente uma instância para possível aplicação daquela regra. No caso de gramáticas simbólicas, como a gramática generativa de Chomsky, se considerado sobre um alfabeto de símbolos, o reconhecimento de uma seqüência de símbolos produz a posição onde a seqüência é encontrada, na forma de um número inteiro que indexa a seqüência específica sobre a totalidade de elementos que compõe o objeto para derivação. Exemplificando, para uma regra simbólica simples na forma $b \rightarrow ab$ aplicada sobre um conjunto o inicial de símbolos $abbc$, o reconhecimento de possibilidades derivacionais é formado pelo conjunto de índices $\{2,3\}$, que considerando a base para indexação iniciando em 1, significa duas instâncias possíveis, nas posições 2 e 3, exatamente as posições dos símbolos b encontrados no conjunto inicial de símbolos. Deste modo, as duas possibilidades de derivação da regra proposta resultam respectivamente nas seqüências $aabbc$ e $ababc$. É importante observar que para qualquer gramática simbólica, qualquer símbolo passível de reconhecimento deverá estar presente ou no conjunto inicial ou então ter sido inserido explicitamente por qualquer derivação de regra. Para uma gramática simbólica regular não é possível o reconhecimento de qualquer símbolo apenas com a combinação de outros símbolos, mas somente se uma regra específica proceder ao reconhecimento de um padrão e a seguir inserir novos símbolos.

Para gramáticas de formas o reconhecimento de padrões é mais complexo e mais rico, considerando-se um padrão válido como parte de elemento do vocabulário ou ainda como a combinação de partes de dois ou mais símbolos distintos. Deste modo, o processo derivacional pode produzir novas instâncias de um determinado padrão.

Na figura 24 uma regra simples determina o padrão de reconhecimento, um quadrado, e o conseqüente processo derivacional, a inserção de outro quadrado adjacente ao original e com deslocamento. Na gramática de formas deste exemplo, o elemento único do vocabulário é o quadrado, e a única regra definida determina o reconhecimento de um elemento e a inserção de outro elemento adjacente ao primeiro. O processo de derivação com consecutivas aplicações desta única regra, apresentado na figura 24 com destaque para as instâncias de aplicação para cada passo derivacional, resulta em uma nova instância passível de reconhecimento, conforme mostrado no quarto passo da derivação. Porém, esta instância, que apesar de ter diferença em escala é simétrica a original e assim válida para derivação, não foi inserida explicitamente por nenhum dos passos anteriores, nem tampouco estava presente nos elementos iniciais. Esta instância surgiu, ou emergiu, da combinação de quatro outros elementos, o elemento inicial e outros três elementos inseridos em cada passo derivacional anterior.

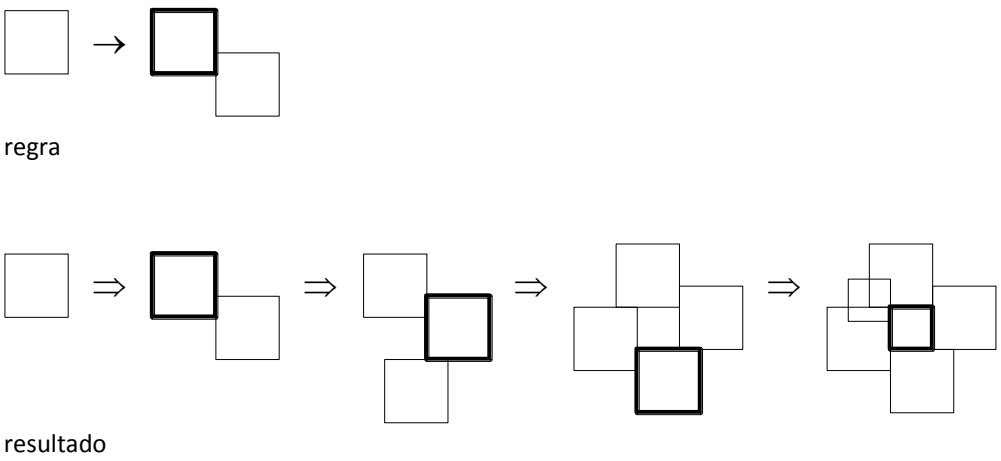


Figura 24: Regra e processo de derivação com destaque para instância de aplicação e emergência de forma.

A característica de emergência de formas é única para gramática de formas, não havendo similar em qualquer outra gramática simbólica. É certo

que podemos desenvolver uma gramática simbólica com regras de derivação específica para que com o devido reconhecimento de determinado padrão novos símbolos sejam inseridos, por exemplo, se a gramática simbólica exemplificada anteriormente for enriquecida com a regra $bbbb \rightarrow bbabb$, inserindo um novo elemento do vocabulário de certa forma semelhante ao símbolo emergente na gramática de formas da figura 24, porém com esta nova regra definida, não é possível afirmar que um símbolo "emergiu" da combinação de outros, mas sim que este símbolo foi explicitamente inserido com a derivação de uma regra definida.

5.3.2. Submergência de formas

Com gramáticas simbólicas a inserção de um símbolo determina que aquele símbolo seja parte do resultado, seja resultado parcial ou final. Somente através da aplicação de uma regra de derivação um símbolo poderá deixar de existir, sendo explicitamente removido através de uma regra de derivação. Por exemplo, para uma gramática simbólica composta pela regras $r_1 := b \rightarrow ab$ e $r_2 := aab \rightarrow ab$, somente com a derivação da segunda regra um símbolo a poderá deixar de existir no conjunto resultante. Não há outra possibilidade com estas regras definidas.

Diferentemente de gramáticas simbólicas, uma gramática de formas pode ser definida de modo que a aplicação de alguma regra derivacional pode fazer com que uma instância do vocabulário que estava anteriormente presente no desenho deixe de estar, ou seja, uma forma submerge durante o processo de derivação.

A figura 25 apresenta uma gramática de formas bastante similar a apresentada anteriormente, porém com uma pequena diferença. O elemento do lado esquerdo da regra não está totalmente presente no lado direito da mesma regra. Ou seja, esta é uma gramática subtrativa, conforme classificado na seção 3.2.1. Considerando que o vocabulário desta gramática é composto tanto pelo quadrado, visto no lado esquerdo da regra, quanto pela forma irregular, visto em composição com outro quadrado no lado direito da regra, o processo derivacional apresentado mostra no seu quarto passo que uma forma

do vocabulário que estava anteriormente presente deixa de estar. É importante ressaltar que esta subtração de formas do vocabulário não é definida explicitamente na regra de derivação, mas sim é resultado de todo o processo derivacional, para este elemento inicial e seqüência de passos específicos.

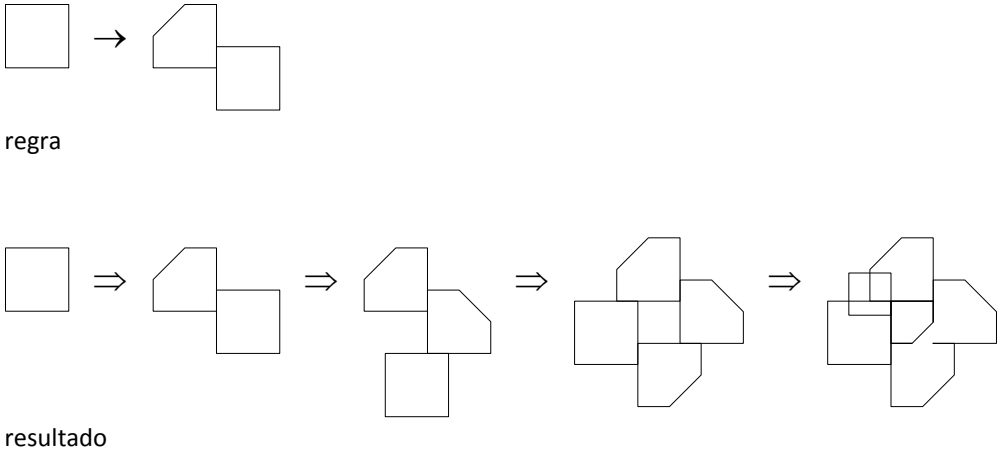


Figura 25: Regra e processo de derivação resultando submergência de formas.

A consideração de elementos de desenho para gramática de formas não pode de modo algum simplificar a existência de formas do vocabulário, tomando a existência de elementos do vocabulário como símbolos atômicos. Cada passo de derivação pode ter efeito não só sobre os elementos determinados pela regra, mas também sobre qualquer outro elemento coincidente a estes, como pode ser visto na figura 25. O passo de número quatro de derivação elimina duas instâncias anteriormente válidas da forma irregular, especificamente as formas inseridas nos passos dois e três, exatamente sobre as formas de quadrado inseridas nos passos um e dois.

Igualmente como que acontece na emergência de formas, uma gramática simbólica pode ser desenvolvida de modo a que elementos do vocabulário sejam subtraídos no processo de derivação, porém esta remoção de elementos deve estar explicitamente definida em uma regra de derivação.

5.4. Finitude, simetria e recursividade

Para caracterizar uma gramática de formas é necessário avaliar a extensão de possibilidades derivacionais – finita ou infinita, as operações onde a simetria pode ser observada dividindo a linguagem de design igualmente em partes simétricas, assim como os processos recursivos. Especialmente a emergência de formas é um fator de grande importância em processos recursivos, pois se bem determinada pode indicar passos infinitos de recursividade.

5.4.1. Finitude

Na matemática “finitude” é a propriedade daquilo que não é infinito ou infinitesimal; relacionado a um número positivo ou negativo diferente de zero; aquilo que é contável e com um número limitado de elementos.

Para facilitar o controle sobre derivações uma gramática de formas pode ser enriquecida com marcadores simbólicos, que deverão ser considerados no processo de reconhecimento de padrões e respeitados conforme especificado nas regras de derivação. Na figura 26 duas regras de derivação são apresentadas e se considerarmos duas gramáticas distintas, cada uma delas composta unicamente por uma das regras, somente uma regra para cada gramática, os seus respectivos resultados sobre um elemento inicial simples são apresentado a seguir. Observa-se que apesar de ser possível um número infinito de derivações, em um ciclo, a linguagem produzida por cada uma das gramáticas é finita. Neste exemplo o processo de derivação pode ser infinitamente repetido, sendo que após exatamente cinco derivações consecutivas sobre o único ou o último elemento inserido, há a sobreposição de elementos sobre os elementos já inseridos, o que não causa modificação no elemento de desenho final. Conseqüentemente, o processo de derivação é infinito, porém os elementos da linguagem de design produzida são finitos.

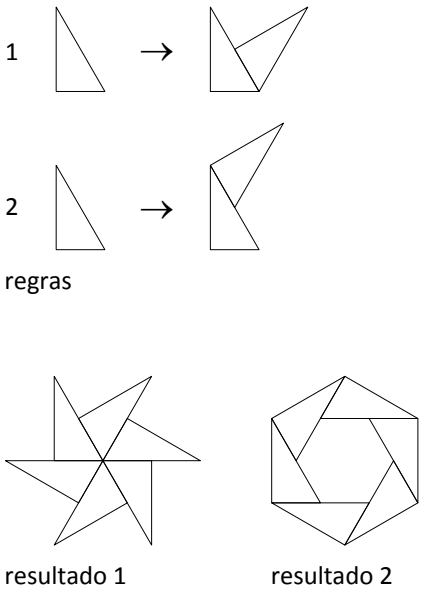


Figura 26: Regras e resultado de derivação finita, porém com possivelmente número infinito de passos.

Porém, se considerarmos uma mesma gramática formada por ambas as regras apresentadas na figura 26, é possível um número infinito de derivações bem como a produção de um número infinito de elementos na linguagem de design definida, conforme apresentado na figura 27. Os resultados 3 e quatro são exemplos possíveis produzidos por tal gramática, porém infinitos outros existem.

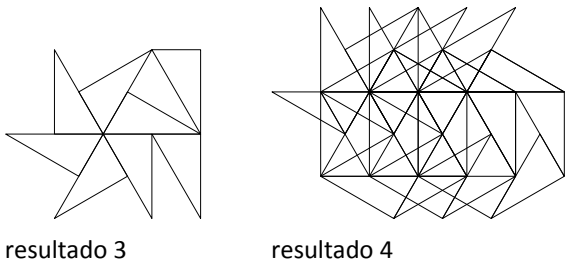


Figura 27: Resultados de derivação para uma gramática composta por ambas as regras apresentadas na figura 26, com número infinito de passos e também número infinitos de elementos na linguagem.

Observando o comportamento geral de gramáticas com relação ao número de derivações e resultados possíveis, é possível determinar uma relação de equivalência com gramáticas simbólicas. Pode ser necessária a extensão dos símbolos empregados na gramática simbólica, de modo a produzir tantos elementos quanto forem necessários para a validade da relação de equivalência. Uma extensão possível e intuitiva é o acréscimo de atributos

para cada símbolo, caracterizando os atributos de instância para cada elemento do vocabulário da gramática de formas. Exemplificando, uma gramática simbólica com infinitas derivações pode ser definida com uma única regra de derivação como $s_i \rightarrow s_i + s_{i+1}$, onde o símbolo s_i representa um único elemento do vocabulário, porém acrescido do índice i, e o operador $+$ representando a concatenação de símbolos. Evidentemente, o processo derivacional sobre esta gramática simbólica é infinito. Considerando o elemento inicial s_0 o processo de derivação resulta em uma cadeia infinita de símbolos na forma $s_0 + s_1 + s_2 + \cdots + s_n$.

5.4.2. Simetria

O conceito de “simetria” tem geralmente duas principais interpretações: a primeira interpretação, mais imprecisa, considera o aspecto harmônico ou estético de proporcionalidade e equilíbrio, possivelmente refletindo em beleza ou perfeição; o segundo significado é preciso e formalmente bem definido, como o conceito de padrão de auto-semelhança que pode ser percebido em sistemas geométricos formais. A noção precisa de simetria pode ser deduzida de acordo com o contexto, e neste estudo o principal interesse é a simetria produzida através de transformações geométricas de escala, reflexão e rotação.

Em gramáticas de formas e conseqüentemente em linguagens de design a observação de simetrias pode facilitar o entendimento tanto do processo derivacional como a concepção do universo definido. Nas regras de exemplo apresentadas na figura 26 não há qualquer simetria, pois o próprio elemento do vocabulário, um triângulo retângulo escaleno, não possui qualquer relação de simetria, nem tampouco as regras derivacionais definidas. Todavia, nos exemplos apresentados na figura 27 pode-se observar diversas relações de simetria, especialmente no resultado 4, onde assim como os elementos do universo definido pela linguagem de design são infinitos, são também infinitas as relações de simetria. Especialmente nestes exemplos, devido a estrutura das regras de derivação onde cada elemento inserido difere do original por um

ângulo de 60 graus, completando seis passos para o retorno a posição original, as relações de simetria também terão esta mesma ordem.

O exame minucioso do universo produzido por uma gramática de formas pode determinar diferentes relações de simetria, podendo estas ser exploradas para o melhor entendimento tanto do processo derivacional repetido quanto da forma geral dos elementos de desenho produzidos. Ao determinarmos quais os princípios gerais das simetrias para uma linguagem de design específica é possível conceber, avaliar e até mesmo validar elementos individuais.

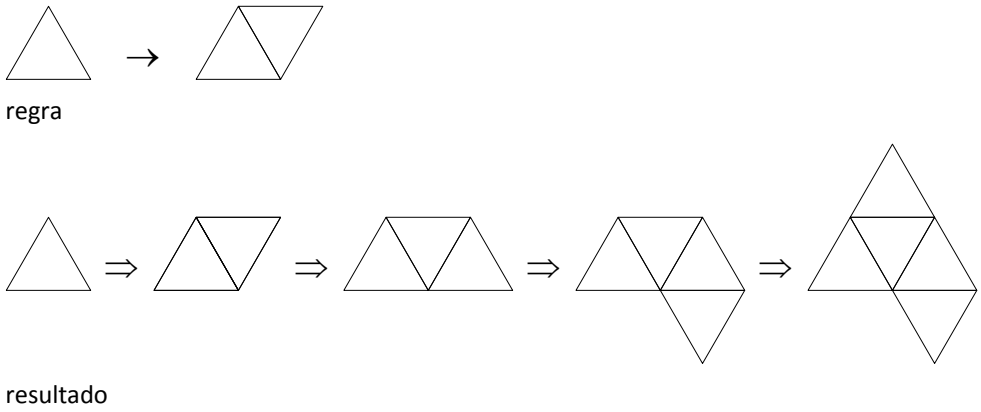


Figura 28: Regra de derivação produzindo simetria de 120 graus e resultado de derivação apresentando diversas relações de simetria em seus elementos.

A figura 28 apresenta uma regra de derivação produzindo a repetição do elemento com rotação de 120 graus. O próprio elemento do vocabulário deste exemplo, um triângulo eqüilátero, possui clara simetria. Neste caso a relação de simetria pode ser verificada tanto no processo derivacional, com três possibilidades simétricas existentes na própria regra de derivação, quanto nos elementos de desenho produzidos. Além da simetria rotacional, percebida em vários níveis nos elementos de desenho produzidos com maior número de derivações, existe também uma relação de simetria de escala, pois com a composição de quatro triângulos o mesmo padrão triangular é produzido, porém com escala dobrada. Na realidade, apenas três triângulos seriam suficientes para a produção do novo triângulo em escala, porém com a única regra de derivação oferecida não seria possível tal seqüência.

O universo de possibilidades contido na linguagem de design definida no exemplo da figura 28 produz evidentemente uma grade triangular de

dimensões infinitas, onde existem também infinitas relações de simetria, tanto de simetria rotacional quanto de simetria em escala.

A consideração de relações de simetria no exame de elementos de desenho produzidos por uma determinada gramática de formas pode reduzir enormemente a complexidade para o entendimento das possibilidades derivacionais. Também o entendimento da linguagem de design definida por uma gramática de formas pode ser facilitado com a avaliação das relações de simetria, tornando possível a rápida avaliação e validação de possíveis elementos de desenho, se contidos ou não no universo definido.

5.4.3. Recursividade

Recursão, na matemática e na ciência da computação, é o método de definição de funções no qual a função sendo definida é aplicada em sua própria definição. O exemplo mais elementar para uma função recursiva é a função fatorial:

$$n! = \begin{cases} 1, n = 0 \\ n \times (n-1)!, n > 0 \end{cases}$$

definida para $n \geq 0$ e indefinida para $n < 0$. Outro exemplo clássico de definição recursiva é a função *fibonacci*:

$$Fib(n) = \begin{cases} 0, n = 0 \\ 1, n = 1 \\ Fib(n-1) + Fib(n-2), n > 1 \end{cases}$$

definida também somente para número positivos. Definições recursivas são especialmente úteis na especificação de elementos infinitos através de elementos finitos, usado do modo geral para descrever um processo repetitivo de auto-semelhança.

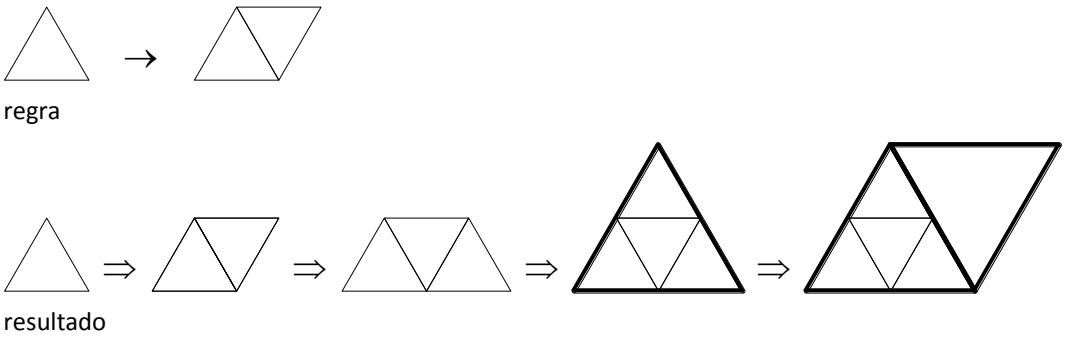


Figura 29: Uma regra simples e o resultado de derivações com destaque no processo recursivo em simetria de escala.

Na figura 29 a mesma regra apresentada na figura 28 é aplicada sobre o mesmo elemento inicial, porém no quarto passo de derivação a regra é aplicada sobre um elemento com simetria de escala dobrada, mostrado em destaque na figura. Este é um caso de simetria de escala onde a operação pode ser repetida recursivamente infinitas vezes, produzindo simetria de escala com potências de 2, sempre com relação ao elemento inicial. Se considerarmos a partir deste mesmo exemplo a regra oposta, transformando a forma trapezoidal no triângulo eqüilátero, a seqüência de derivações será também recursiva, porém agora a relação de simetria de escala será de fator ½, ou seja, potências negativas de 2.

6. CONCLUSÃO

O conteúdo deste trabalho foi desenvolvido durante o trabalho como bolsista de Ciência da Computação no Laboratório para Simulação e Modelagem em Arquitetura e Urbanismo (SIMMLAB) da Faculdade de Arquitetura da Universidade Federal do Rio Grande do Sul. Inicialmente o estudo objetivava a criação de processos generativos para arquitetura, urbanismo e design, porém ficou evidente a relação das gramáticas de formas com outros modelos computacionais teóricos. Este desenvolvimento culminou com apresentação de dois trabalhos no *Design, Computing and Cognitions International Congress* (DCC'08) em Atlanta, EUA no ano de 2008, que foram "*AutoCAD Implementation of Shape Grammars*" e "*Category Theory Applied to Shape Grammars*". Também durante este período foi desenvolvido o currículo da disciplina eletiva ARQ01034, ministrada aos alunos do curso de graduação em Arquitetura e Urbanismos e também a alunos de Pós-Graduação em Design com ênfase em Design e Tecnologia (PGDesign), também da Faculdade de Arquitetura, com participação na elaboração do currículo, dos exercícios práticos propostos utilizando AutoLisp, e na monitoria.

Dado o relacionamento entre gramáticas de formas e processos de derivação, especialmente as gramáticas simbólicas de Chomsky, o processo comparativo foi inevitável. Neste processo a emergência de formas mostrou-se única e sem similar em modelos simbólicos, instigando uma investigação mais aprofundada de similaridades e a formulação de classes comparativas. Para auxiliar o processo de pesquisa foram desenvolvidos diversos algoritmos e ferramentas computacionais, com aplicação direta no estudo envolvendo arquitetura, urbanismo e design, inclusive com aplicações práticas apresentadas em amostra específica na Faculdade de Arquitetura.

A utilização de propriedades e formalismos da teoria das categorias sobre elementos de gramáticas de formas mostrou-se bastante útil para o desenvolvimento de algoritmos, permitindo a exploração mais eficaz sobre o universo composto por linguagens de design específicas. Também no desenvolvimento de algoritmos para reconhecimento de padrões, síntese e análise de elementos de desenho, a aplicação de propriedades categoriais permite otimizar a utilização de recursos e a eficácia dos processos.

No estudo sobre linguagens de design e sobre os processos cognitivos envolvendo elementos estéticos a gramática de formas apresenta qualidades únicas, sendo mais intuitiva e natural na manipulação de elementos de desenho. É possível emular os mesmos resultados através da manipulação de símbolos, por exemplo, utilizando uma gramática generativa de Chomsky, todavia este processo é bastante custoso em termos de recursos computacionais e pode não ser eficiente, especialmente se considerado o processo de desenvolvimento incremental, onde o utilizador observa resultados obtidos e a partir destes resultados direciona o desenvolvimento a fim de alcançar seus objetivos.

A continuidade deste estudo deverá explorar com maior rigor as diferentes classes de modelos computacionais, determinando relações de ordem entre modelos teóricos simbólicos e modelos não simbólicos, com especial atenção às características como a emergência de formas.

Outra área que pode ser grandemente desenvolvida tirando proveito dos resultados obtidos neste estudo é a determinação mais precisa dos processos analíticos e de síntese envolvendo linguagens de design. É natural para nós percebermos diferenças e similaridades em diferentes elementos estéticos, por exemplo, ao observarmos obras arquitetônicas de diferentes autores ou diferentes épocas, porém este processo parece ser bastante intuitivo e com alto paralelismo. Portanto a especificação exata de tais processos mentais é um grande desafio, sendo este também um possível objetivo futuro para continuidade deste estudo, buscando a sistematização e o conseqüente desenvolvimento de sistemas.

REFERÊNCIAS

Beck, K. (2003). *Test-Driven Development by Example.* Addison Wesley.

Berman, G., Doolen, G., Mainieri, R., & Tsifrinovich, V. (1998). The Turing Machine. In *Introduction to Quantum Computers* (pp. 8-10). London: World Cientific.

Boker, U., & Dershowitz, N. (2005). How to compare the power of computational models. *Tel Aviv University* (pp. 3-4). Tel Aviv: School of Computer Science.

Chomsky, N. (1956). Three Models for the Description of Language. *IRE Transactions on Information Theory , 2*, pp. 113–124.

Church–Turing thesis. (2008, 11 4). Retrieved 3 2010, 20, from Wikipedia: http://en.wikipedia.org/wiki/Church-Turing_thesis

Gero, J. (2000). *Artificial Intelligence in Design '00.* Dordrecht, The Nedherlands: Kluwer Academic Publishers.

Gero, J., & Maher, M. (1992). *Modeling Creativity and Knowledge-Based Creative Design.* New Jersey, USA: Lawrence Erlbaum Associates, Inc.

Gips, J. (1975). *Shape Grammars and Their Uses: Artificial Perception, Shape Generation and Computer Aesthetics.* Basel: Birkhäuser.

Knight, T. (2003). Computing with Emergence. *Environment and Planning B: Planning and Design , 30*, pp. 125-155.

Knight, T. (1998). Shape grammars. *Environment and Planning B: Planning and Design* (Anniversary Issue), pp. 86–91.

Knight, T. (2000). *Shape Grammars in Education and Practice: History and Prospects*. (MIT, Editor) Retrieved Feb 24, 2008, from Department of Architecture, School of Architecture and Planning: http://www.mit.edu/~tknight/IJDC/

Leyton, M. (2001). A Generative Theory of Shape. In *Lecture Notes in Computer Science.* Berlin: Springer.

McGill, M. (2001). A Visual Approach for Exploring Computational Design. In *SMArchS thesis.* MIT.

Pierce, B. (1993). Basic Category Theory for Computer Scientists. In *Foundation of Computer Series.* Cambridge: The MIT Press.

Sá, J. (2001). *Pattern Recognition: Concepts, Methods and Applications.* New York: Springer.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Redmond, Washington: Microsoft Press.

Stiny, G. (1992). "Weights". *Environment and Planning B: Planning and Design , 19*, pp. 413-430.

Stiny, G. (1975). Pictorial and Formal Aspects of Shape and Shape Grammars. In *Computer Generation of Aesthetics Objects.* Basel: Birkhäuser.

Stolfi, J. (1989). In *Primitives for Computational Geometry* (pp. 189-198). Palo Alto: Stanford University.

Tapia, M. (1999). A Visual Implementation of a Shape Grammar System. *Environment and Planning B: Planning and Design , 26*, pp. 59-73.

Turing machine equivalents. (2009, 12 23). Retrieved 3 3, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Turing_machine_equivalents

Wang, Y., & Duarte, J. (2001). Automatic Generation and Fabrication of Design. In *Automation in Construction* (Vol. 447). Elsevier.