Aplicação da Complexidade de Kolmogorov na Caracterização e Avaliação de Modelos Computacionais e Sistemas Complexos*

Carlos A. P. Campani[1] e Paulo Blauth Menezes[2]

[1] IFM/ UFPel - CP 354, 96010-900, Pelotas/RS, Brasil
campani@ufpel.tche.br
[2] II/ UFRGS - CP 15064, 91501-970, Porto Alegre/RS, Brasil
blauth@inf.ufrgs.br

Resumo Neste artigo é apresentada uma proposta de aplicação da complexidade de Kolmogorov para a caracterização de sistemas e processos, e a avaliação de modelos computacionais. A metodologia desenvolvida representa uma ferramenta teorética para resolver problemas de ciência de sistemas. Para ilustrar a proposta são apresentadas duas aplicações da metodologia, desenvolvidas pelos autores. A primeira no processo de desenvolvimento de software, e a segunda em modelos de animação gráfica. Ao final é introduzida, de forma breve, uma terceira aplicação do método com o objetivo de caracterizar sistemas dinâmicos de comportamento caótico, o que mostra a potencialidade da metodologia.

1 Introdução

Muitos problemas em ciência de sistemas são difíceis de serem tratados pois a caracterização destes sistemas é muito complexa. Freqüentemente nos deparamos com a necessidade de comparar modelos de sistemas ou avaliar a possibilidade prática de atingir algum determinado resultado. Neste caso enfrentamos a dificuldade relacionada com a nossa limitada capacidade de expressar tais problemas através de algum formalismo matemático expressivo o suficiente para permitir algum tipo de inferência sobre o nosso problema.

Este artigo apresenta uma proposta de caracterização de sistemas e processos, e avaliação de modelos baseada em *complexidade de Kolmogorov*. O artigo é enriquecido com duas aplicações, desenvolvidas pelos autores (parte da abordagem foi publicada em [1]), a primeira em *desenvolvimento de software*, e a segunda em *animação gráfica*, e por uma proposta de aplicação mais abrangente (apresentada na Seção 4.3), que é um resultado adicional que mostra a potencialidade da metodologia.

A metodologia proposta efetua comparações e avaliações que se referem à compressão de dados. A taxa de compressão de dados é um parâmetro importante dentro da nossa proposta, pois caracteriza o *conteúdo de informação* de

* Este trabalho é parcialmente financiado por: CNPq (Projetos HoVer-CAM, MEFIA), UFRGS (Projeto Hyper-Automaton), FAPERGS (Projeto QaP-For) e CAPES/UFPel.

um objeto. O método se baseia na teoria da computabilidade e, neste contexto, um dos conceitos mais importantes é o de *simulação de máquinas.*

Os principais resultados apresentados são:

- Definição do processo de desenvolvimento de software como compressão de dados. Segundo nossa abordagem, o processo de desenvolvimento de software é, em um sentido formal, imprevisível e incompleto;
- Aplicação do método, baseado em simulação de máquinas, para comparação entre modelos de animação gráfica (e em certo sentido avaliação destes modelos), com respeito à compressão de dados;
- Uma proposta de caracterização de sistemas dinâmicos de comportamento caótico baseada em incompressividade e previsão por compressão de dados.

A metodologia aqui desenvolvida representa uma ferramenta teorética para resolver problemas de ciência de sistemas.

2 Complexidade de Kolmogorov

A complexidade de Kolmogorov é uma teoria algorítmica da aleatoriedade que trata da quantidade de informação de objetos individuais, medida através do tamanho de sua menor descrição algorítmica [7,8,9,12]. Os objetos sobre os quais está definido o conceito de complexidade de Kolmogorov são as strings binárias (ou números naturais, já que o conjunto das strings binárias é enumerável). Portanto, a complexidade de Kolmogorov associa a cada string binária (ou número natural) um valor que é a sua "complexidade".

Esta seção do artigo apresenta a definição da complexidade de Kolmogorov e alguns resultados bem conhecidos da área.

A complexidade de Kolmogorov de uma string binária (ou número natural) é o tamanho de sua menor descrição efetiva (computável) relativo a um método de especificação previamente escolhido (máquina de Turing de referência). Objetos e descrições são representados indistintamente como strings binárias ou números naturais.

Definimos $|.|$ como o tamanho de uma string binária (número de dígitos binários). Na Definição 1, $f : \{0,1\}^* \to \{0,1\}^*$ é uma função parcial qualquer. $\{0,1\}^*$ denota o conjunto de todas as strings binárias de tamanho maior ou igual a zero (incluindo a string vazia).

Definição 1. *A complexidade do objeto $x \in \{0,1\}^*$ com respeito a um método de especificação f é definida como*

$$C_f(x) = \min_{f(p)=x} |p| .$$

Se não existe uma descrição p dizemos, por definição, que $C_f(x) = \infty$.

Considere um conjunto de métodos de especificação distintos $f_1, f_2, \ldots, f_n$ que especificam objetos de $\{0,1\}^*$. É fácil construir um novo método f, não necessariamente pertencente a este conjunto, que atribui aos elementos de $\{0,1\}^*$

uma complexidade que excede apenas por uma constante c a complexidade expressa por todos os outros métodos do conjunto [9].

Definição 2. *(Definição de Minoração) Nós dizemos que o método f* minora *(ou* minora aditivamente*) um método g se existe uma constante $c > 0$ tal que, para todo x,*

$$C_f(x) \leq C_g(x) + c\,.$$

Definição 3. *(Método Universal ou Assintoticamente Ótimo) Seja $\mathcal{F}$ uma subclasse das funções parciais sobre o conjunto dos naturais. Uma função f é* universal *(ou* assintoticamente ótima*) para $\mathcal{F}$ se ela pertencer a $\mathcal{F}$ e para toda função $g \in \mathcal{F}$, existe uma constante $c_{f,g} > 0$ que depende apenas de f e g, tal que para todo x, $C_f(x) \leq C_g(x) + c_{f,g}$.*

Definição 4. *(Equivalência de Métodos) Dois métodos f e g são* equivalentes *se para todo x,* $\mathrm{abs}(C_f(x) - C_g(x)) \leq c_{f,g}$*, onde $c_{f,g}$ não depende de x.* abs(.) *denota o valor absoluto de um número.*

A existência de um elemento universal no conjunto dos métodos de especificação permite que este seja usado como método de referência para a definição da complexidade, tornando a medida de complexidade independente do método de especificação escolhido, a não ser por uma constante. Permite assim que a medida da complexidade seja um atributo intrínseco do objeto, o que a torna um conceito objetivo e útil [1].

No entanto, a classe das funções parciais não possui um elemento universal [9]. Além disto, podemos provar que existe um elemento universal na classe das funções parciais recursivas. Esta função é chamada *função parcial recursiva universal* [8,9].

Portanto, a complexidade de Kolmogorov é definida usando-se a máquina de Turing e restringindo-se os métodos de especificação às funções parciais recursivas.

Teorema 1. *(Teorema da Invariância) Existe uma função parcial recursiva u, chamada* universal*, tal que para qualquer função parcial recursiva f e string binária x, e para algum $c > 0$, $C_u(x) \leq C_f(x) + c$ e c depende apenas de f.*

Na prova do Teorema 1 desempenha papel importante o conceito de *simulação de máquinas* e a existência de uma máquina de Turing universal (que simula todas as outras máquinas). Então, podemos provar facilmente o Corolário 1 que define a equivalência entre métodos universais (ou assintoticamente ótimos).

Corolário 1. *(Equivalência de Métodos Assintoticamente Ótimos) Seja u e v duas funções parciais recursivas universais. Então, para todo x,* $\mathrm{abs}(C_u(x) - C_v(x)) \leq c$*, onde c não depende de x.*

Fixando uma função parcial recursiva universal u, chamada *universal de referência* podemos definir $C_u(x) = C(x)$.

Resumindo, podemos afirmar que a complexidade de Kolmogorov de um objeto representa a informação que deve ser fornecida para que o objeto seja computacionalmente (algoritmicamente) construido (ou computado). Assim, a complexidade de Kolmogorov representa quantitativamente a informação imanente neste objeto.

3 Caracterização de Sistemas e Processos e Avaliação de Modelos

A principal contribuição deste artigo é uma proposta de aplicação da complexidade de Kolmogorov como uma ferramenta geral para a caracterização e avaliação de modelos computacionais e sistemas complexos.

A complexidade de Kolmogorov é adequada para este fim, pois uma das características mais importantes dos modelos e sistemas é seu conteúdo de informação.

A idéia geral é que a quantidade de informação caracteriza um processo ou sistema. Eventualmente, podemos estipular níveis máximos de complexidade que são "aceitáveis". Se um processo não é "aceitável" ele não é realizável, pois não existem recursos suficientes alocados para sua realização.

Além de caracterizar estes processos, podemos comparar modelos computacionais e sistemas pelo critério de ser "mais complexo" ou ser "mais simples".

Neste contexto, identificamos como característica central nos objetos sob investigação a quantidade de informação imanente nestes objetos, suprimindo todos os outros aspectos e detalhes do objeto. Assim, consideramos, ao invés de aspectos dinâmicos, apenas os aspectos estáticos relacionados com a construção do objeto (ou computação do objeto), desprezando outros recursos necessários, tal como o tempo de execução, a não ser a informação fornecida para sua computação.

Os seguintes conceitos são fundamentais em nossa abordagem: Incompressividade; minoração e simulação de máquinas; previsão por compressão de dados; não-computabilidade da complexidade; e incompletude dos sistemas formais.

3.1 Incompressividade

O propósito original da complexidade de Kolmogorov era definir seqüências aleatórias formalmente, embasando uma teoria matemática das probabilidades [9].

String binária aleatória Uma string binária é dita *aleatória* se sua complexidade é aproximadamente igual ao tamanho da string.

Assim, definimos as strings simples como sendo aquelas que são *regulares* ou *compressíveis*, e as strings aleatórias ou complexas como sendo aquelas que possuem irregularidade, ou que são *incompressíveis* [8].

Definição 5. *Para uma constante $c > 0$, nós dizemos que a string x é c-incompressível se $C(x) \geq |x| - c$.*

Na Definição 5, a constante c desempenha o papel de "taxa de compressão".

Esta definição de aleatoriedade é muito rígida e exclui qualquer seqüência gerada por um algoritmo de geração de números pseudo-aleatórios de ser uma seqüência genuinamente aleatória, pois sua descrição é muito curta.

É interessante observar que o problema de determinar se uma string é incompressível é um problema indecidível. Suponha que o conjunto das strings incompressíveis é enumerável. Seja f uma função parcial recursiva que enumera as strings incompressíveis. Assim, $f(n)$ denota a primeira string x_0 com complexidade maior que n. Então, $C(x_0) > n$ e $C(x_0) = C(f(n)) \leq C(n)+c = \log n+c$. Assim, $C(x_0) > n$ e $C(x_0) \leq \log n + c$, para qualquer n arbitrário, que é uma contradição.

Esta indecidibilidade mostrar-se-á interessante quando abordarmos o tema "incompletude dos sistemas formais" na Seção 3.5 (obviamente, se não se pode determinar se uma string é incompressível, não se pode determinar formalmente quanta informação é necessária para computa-la, sendo isto uma limitação dos sistemas formais).

Existe uma definição alternativa de seqüência aleatória, de autoria do pesquisador Martin-Löf, baseada em conjuntos nulos efetivos e na existência de um conjunto nulo efetivo, chamado máximo, que contém todos os conjuntos nulos efetivos (teste de aleatoriedade universal) [12]. Não é objetivo deste artigo cobrir tal definição, principalmente porque a definição de Martin-Löf é formalmente equivalente à definição de seqüência aleatória via incompressividade.

3.2 Minoração e Simulação de Máquinas

Como vimos, a minoração (veja Definição 2) traduz que um método de especificação é "mais eficiente" na compressão de dados que outro método.

Seja s_f uma string s descrita pelo método f, e s_g a mesma string descrita por um método g. Então, se para todo s, $|s_f| \leq |s_g| + c$ ("f *minora* g"), significa que *assintoticamente* o método f é mais eficiente na compressão de dados que o método g.

Podemos associar estes métodos com funções parciais recursivas (máquinas de Turing). Assim, se um método f *simula* um método g, então f minora g.

Simulação de máquinas é um conceito central e bem conhecido em teoria da computabilidade. Portanto, o método proposto herda muitos resultados já consolidados e bem conhecidos. Lembremos que a simulação de máquinas induz uma hierarquia de poder computacional e, conseqüentemente, uma herarquia de linguagens (cuja a mais conhecida é chamada hierarquia de Chomsky).

3.3 Previsão por Compressão de Dados

Uma das principais atividades da ciência é a previsão dos fenômenos os quais ela estuda. Esta previsão se constitui em duas etapas. A primeira etapa consta da determinação de uma teoria sobre o fenômeno. Esta teoria deve atender a dois requisitos: ser simples e eficiente na previsão. A segunda etapa é o uso da teoria proposta para antecipar o comportamento futuro do fenômeno.

Podemos entender os fenômenos como seqüências de dígitos binários (strings binárias) e as teorias como programas para a máquina de Turing (funções parciais recursivas) que computam estas strings.

Usaremos probabilidades para expressar nossas inferências, de forma que $\Pr(1|x)$ denota a probabilidade condicional de ocorrer 1 como continuação da string x. Então, a probabilidade expressa pelo menor programa, dentro do espírito de simplicidade que a teoria deve possuir, é uma prévia universal, no sentido que é a forma mais eficiente e precisa de expressar o fenômeno.

Sabemos, pela probabilidade condicional (regra de Bayes), que

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)},$$

com $\Pr(B) > 0$.

Nosso problema pode ser resumido no seguinte: Dada uma string binária x qualquer, desejamos determinar a probabilidade do próximo bit da string ser zero ou um. Aplicando ao nosso caso, com $A =$ "ocorreu 1" e $B =$ "ocorreu x", obtemos

$$\Pr(1|x) = \frac{\Pr(x1)}{\Pr(x)}. \tag{1}$$

Podemos definir **m**, chamada *prévia universal*, que é uma probabilidade prévia baseada no menor programa para a máquina de Turing que computa uma dada string. Esta prévia domina todas as distribuições enumeráveis prévias a não ser por uma constante multiplicativa. Usando esta prévia universal na Equação (1) podemos efetuar *previsão por compressão de dados* [9,10].

O desafio para aplicar este método é encontrar aproximações computáveis da complexidade de Kolmogorov de uma string.

3.4 Não-Computabilidade da Complexidade

Teorema 2. *A função C não é computável.*

Para determinar $C(x)$ devemos executar todos os programas e coletar todos os que param e computam x, escolhendo aquele que tem o menor tamanho. Trivialmente, pelo problema da parada, não podemos decidir se um programa pára ou não [9].

Definição 6. *Uma função f é* semicomputável *se existe uma função (total) recursiva ϕ tal que $\phi(x,t)$ é monotônica em t e $\lim_{t\to\infty} \phi(x,t) = f(x)$.*

Teorema 3. *A função C é semicomputável.*

Isto significa que a função C, embora não possa ser computada com exatidão, pode ser aproximada. Isto tem alguns interessantes desdobramentos como veremos a seguir.

3.5 Incompletude dos Sistemas Formais

Em 1931, Gödel apresentou sua prova que os sistemas formais não poderiam deduzir todas as sentenças verdadeiras de uma teoria. Chaitin [3,4] construiu uma versão informação-teorética da prova de Gödel. Ele mostrou que uma sentença com mais complexidade de Kolmogorov que os axiomas do sistema não pode ser deduzida neste sistema.

Este resultado é uma conseqüência da indecidibilidade das seqüências aleatórias.

Seja o par ordenado (f, a), representando as regras de inferência e os axiomas de um sistema formal em particular. $f(a)$ denota o conjunto de teoremas que pode ser provado usando-se o sistema (f, a).

Teorema 4. *(Teorema de Chaitin) Considere um sistema axiomático (f, a). Suponha que uma proposição da forma "$C(x) \geq n$" está em $f(a)$ somente se $C(x) \geq n$ é verdadeiro. Então uma proposição da forma "$C(x) \geq n$" está em $f(a)$ somente se $n \leq |a| + c$, onde c é uma constante que só depende de f.*

Isto é verdade pois existem infinitas provas com complexidade maior que (f, a). O teorema estabelece um limite de complexidade para o que pode ser provado em um sistema formal.

O teorema provado por Chaitin extende o resultado dado por Gödel e demonstra que os sistemas formais possuem limites inerentemente relacionados com a complexidade de Kolmogorov de seus axiomas.

4 Aplicações da Metodologia Proposta

4.1 Caracterização do Desenvolvimento de Software

Um dos principais objetivos das metodologias de desenvolvimento de software é a obtenção de um código ótimo e programas corretos com o menor custo possível. As métricas e métodos de engenharia de software são importantes para prever esforços e custos do processo de desenvolvimento.

Nós defendemos que o tamanho do programa é um parâmetro importante para o desenvolvimento de software, porque tem influência sobre o gerenciamento do projeto de software. O uso da complexidade de Kolmogorov mostra que o tamanho do programa está relacionado com a medida da complexidade do problema que será solucionado pelo programa em desenvolvimento, independente de outros fatores como linguagem de programação escolhida ou a habilidade do programador (pelo Teorema da Invariância a complexidade é invariante com relação ao método de especificação adotado). A escolha do tamanho do menor programa, entre outras medidas, define uma medida adequada de complexidade, uma *complexidade universal baseada no tamanho dos programas* [1].

Nossa abordagem está relacionada com uma visão do processo de desenvolvimento de software como *compressão de dados.*

Segundo nossa abordagem, podemos argumentar que os métodos formais possuem a propriedade de "incompletude", em um sentido matemático muito

semelhante ao da prova da incompletude da aritmética feita por Gödel em 1931 [1,4]. Esta conclusão está baseada no trabalho de Chaitin sobre a incompletude dos sistemas formais [3].

Um outro importante aspecto do desenvolvimento de software é a previsibilidade. A previsibilidade do desenvolvimento de software é a habilidade de antecipar tempo de desenvolvimento, esforços e custos do processo de desenvolvimento. Um parâmetro fundamental para estimar custos e prever esforços é o tamanho do código fonte [1]. O tempo médio que os programadores gastam para escrever programas pode ser estimado pela produtividade média dos programadores. Nós podemos estimar o número total de linhas de código e usar esta informação para estimar os recursos necessários e prazos para entrega do software.

Nós estamos propondo o uso da complexidade de Kolmogorov como uma métrica para o desenvolvimento de software. Se nós aceitarmos que esta métrica é adequada, por ser independente da linguagem de programação usada e da habilidade dos programadores, nós vamos perceber um problema relacionado com a não-computabilidade da complexidade. Nós sugerimos que o processo de desenvolvimento de software é, neste sentido, formalmente imprevisível [1].

Pela semicomputabilidade da complexidade podemos afirmar que a melhor solução, em relação ao tamanho do código, pode ser apenas aproximada.

Segundo os autores [1], estes resultados (incompletude dos métodos formais e imprevisibilidade do processo de desenvolvimento de software) sugerem que o desenvolvimento de software é, a despeito dos esforços para sua formalização, mais subjetivo e empirico que objetivo e formal. Estes limites apresentados caracterizam o processo de desenvolvimento de software como experimental e baseado em heurísticas como, por exemplo, o desenvolvimento científico em física e química.

Estas conclusões, segundo os autores, sugerem que a engenharia de software é uma área científica não totalmente caracterizada pelo trabalho típico da engenharia, mas também pela metodologia das ciências experimentais [1].

4.2 Comparação e Avaliação de Modelos de Animação Gráfica

Uma aplicação interessante da complexidade de Kolmogorov recentemente desenvolvida, e que ilustra nossa metodologia, é seu uso como uma medida para comparação e, em certo sentido, avaliação de modelos de animação gráfica [2]. Esta comparação/avaliação baseia-se na taxa de compressão de dados obtida em um modelo de animação gráfica. O desenvolvimento de um conjunto de conceitos e métodos para avaliações deste tipo é de muita utilidade para o desenvolvimento de aplicações para Web onde há grande necessidade de recursos multimídia sofisticados e, conseqüentemente, um uso otimizado e econômico do canal de comunicação entre servidor e cliente.

De um modo geral, os formatos de multimídia possuem algoritmos de compressão de dados que melhoram o desempenho da aplicação na troca de arquivos de imagem, som, animações e vídeo. Estas aplicações oferecem recursos de manipulação dos objetos de multimídia tais como rotação, translação, sincronização entre eventos, etc.

O formato mais comum de animação gráfica usado em páginas da Web são as animações GIF (Graphics Interchange Format). Um arquivo de animação GIF é composto por uma seqüência de imagens comprimidas que são apresentadas em seqüência formando uma animação de grande utilidade como recurso para expressão de idéias. Recentemente o surgimento de novos formatos multimídia, tal como o Flash da Macromedia, permitiram maior flexibilidade na construção de páginas Web.

Uma animação é formada por uma seqüência de quadros mostrados na tela em intervalos de tempo. Definimos "animação" como o par (s, t), onde s é uma seqüência $s = s_1 s_2 s_3 \cdots s_n$, cada s_i é um quadro (ou *frame*), e t é uma seqüência de intervalos de tempo $t = t_1 t_2 \cdots t_n$, cada $t_i \in \mathbb{N}$. Desta forma, definimos uma animação como uma string binária "computada" de alguma maneira. Isto representa seu "comportamento dinâmico", isto é, o resultado da execução da animação, mas não nos diz nada a respeito de como armazenar e produzir a animação. Assim, devemos definir um mecanismo que "computa" a animação.

Definição 7. *Um* descritor de animação *é um par $\phi = (\alpha, \Delta)$, onde α é um conjunto indexado de imagens (ou atores) e Δ é uma função, $\Delta : \alpha \mapsto a$, isto é, Δ mapeia α na animação alvo a.*

Por exemplo, no formato GIF a função Δ é muito simples. Todos os quadros estão armazenados literalmente dentro do arquivo GIF e a animação segue seqüencialmente, do primeiro quadro até o último.

Definição 8. *Sejam $s, s' \in \{0,1\}^*$ strings binárias. Nós chamamos δ um* algoritmo de descompressão *e, para todo s' existe algum s, tal que $s' = \delta(s)$, onde s é o código de s', com $|s| \leq |s'| + c$. Nós chamamos γ um* algoritmo de compressão *se, para todo s, $s = \delta(\gamma(s))$, e γ deve ser uma função injetiva. Nós chamamos o par $\Gamma = (\gamma, \delta)$ de* esquema de compressão.

É importante notar que a Definição 8 trata de compressão de dados sem perdas, não incluindo o caso de algoritmos de compressão de dados com perdas. Agora podemos definir formalmente o conceito de "melhor compressão".

Definição 9. *Sejam $\Gamma_1 = (\gamma_1, \delta_1)$ e $\Gamma_2 = (\gamma_2, \delta_2)$ dois esquemas de compressão. Nós dizemos que Γ_1 é* melhor *que Γ_2 se, para todo s e algum $c > 0$, $|\gamma_1(s)| \leq |\gamma_2(s)| + c$.*

Nesta última Definição, nós podemos encontrar a ligação do nosso problema com a complexidade de Kolmogorov. $|\gamma(s)|$ é o tamanho do código de s em um esquema de compressão (γ, δ). Um esquema de compressão Γ_1 é melhor que Γ_2 se e somente se Γ_1 *minora* Γ_2.

Definição 10. *Um* esquema de animação *é um par $G = (\Gamma, \phi)$, onde Γ é um esquema de compressão e ϕ é um descritor de animação.*

Observe que o algoritmo γ é aplicado sobre α, o conjunto de images (ou atores) da Definição 7. Seja G um esquema de animação e seja a uma animação

alvo. Nós definimos $G(a)$ como sendo a animação a executada pelo esquema G, e definimos $|G(a)|$ como o número de bits necessários para armazenar a animação a usando o esquema G. Pode-se mostrar que, para algum $c > 0$,

$$|G(a)| = |\gamma(\alpha)| + |\Delta| + c\,, \tag{2}$$

onde $|\gamma(\alpha)|$ é o número de bits necessários para armazenar todas as images (ou atores) em α usando-se a compressão γ, e $|\Delta|$ é o tamanho da função Δ. O significado de $|\Delta|$ é que nós devemos descrever (ou especificar) a função Δ de alguma forma efetiva. Assim, nós contamos o número de bits desta descrição.

Definição 11. *Sejam G_1 e G_2 dois esquemas de animação. G_1 é* melhor *que G_2 se, para toda animação alvo a e algum $c > 0$, $|G_1(a)| \leq |G_2(a)| + c$.*

Isto é, G_1 é melhor que G_2 se G_1 *minora* G_2 no sentido de minoração da complexidade de Kolmogorov. Nós chamamos G_1 e G_2 respectivamente máquina-G_1 e máquina-G_2 para enfatizar este relacionamento entre os conceitos. Isto é, a máquina-G_1 é melhor que a máquina-G_2 se a máquina-G_1 *simula* a máquina-G_2. Como nós formalizamos o conceito de "melhor animação" como simulação de máquinas, induzido pela complexidade de Kolmogorov, nós devemos definir formalmente as animações como máquinas.

Nós percebemos que a taxa de compressão obtida é acarretada por dois diferentes aspectos:

- O algoritmo de compressão usado para comprimir as imagens usadas na animação;
- O modo como a animação é computada a partir deste conjunto de imagens.

Estas duas partes são representadas na Equação (2) como $|\gamma(\alpha)|$ e $|\Delta|$, respectivamente.

O conceito de simulação de máquinas (minoração) é central em nossa abordagem, permitindo a comparação dos modelos quanto à compressão de dados obtida. O método herda resultados bem conhecidos de simulação de máquinas de teoria da computabilidade (já que o método trabalha construindo provas de simulação de máquinas).

Da mesma forma que a simulação de máquinas induz uma hierarquia de máquinas (e de linguagens), em nosso método ela induz uma hierarquia de métodos de animações.

4.3 Caracterização e Previsão de Sistemas Caóticos

Um dos postulados clássicos da ciência moderna é que todo fenômeno pode ser previsto, desde que se conheça com precisão suas condições iniciais e a sua regra de transformação (lei geral que rege o comportamento do fenômeno). No entanto, muitos fenômenos são de difícil previsão pois são muito sensíveis a estas condições iniciais e a erros de arredondamento de valores reais intermediários nos cálculos matemáticos. Assim, qualquer imprecisão na determinação destas condições e

qualquer arredondamento nos cálculos acarreta uma discrepância muito grande entre a previsão e o fenômeno em si (este efeito é chamado de *efeito borboleta*). Tal comportamento é chamado de *comportamento caótico* [6].

O estudo da teoria do caos pode ser definido como o *estudo qualitativo de sistemas dinâmicos não-lineares, de comportamento aperiódico e instável.*

Um sistema dinâmico pode ser definido em termos de um *espaço de fases*, cujos pontos representam possíveis estados do sistema, e uma *lei de evolução* (ou *mapeamento*) que descreve os estados subseqüentes do sistema [5]. A órbita do sistema é a trajetória, ou conjunto de pontos do espaço de fases, descrita pelo sistema.

As características típicas dos sistemas caóticos são: não-linearidade; determinismo; sensibilidade às condições iniciais; irregularidade e aperiodicidade de comportamento; previsão para longo termo impossível devido aos arredondamentos dos cálculos matemáticos com números reais.

O conceito de caos sugere ausência de organização, uma desordem na qual incerteza e imprevisibilidade predominam. No entanto, a teoria do caos estuda aquilo que podemos chamar de a "ordem da desordem".

O conceito de caos está relacionado com o de complexidade. Sistemas complexos podem apresentar um comportamento caótico, o qual é de difícil descrição em termos simples. Os padrões em um sistema caótico estão presentes, mas não são regulares ou facilmente previsíveis. Isto não significa que, em alguns casos importantes, não possa ser feita uma previsão razoável de curto prazo.

As principais aplicações práticas da previsão e controle (assunto não abordado neste artigo) de sistemas caóticos são, entre outros: asas mais eficientes para aeronaves; melhores turbinas; reações químicas em indústrias químicas; desfibriladores implantáveis; planejamento econômico e previsão do mercado financeiro; e redes de computadores [6].

Representaremos o comportamento de um sistema como uma string binária. Um exemplo de sistema caótico é o *doubling map*. Consideremos a expansão binária de um número real, considerada apenas sua parte fracionária. O espaço de fases deste sistema caótico é formado pelos reais no intervalo $[0;1)$. O mapeamento do sistema é definido pela fórmula $x_{n+1} = 2x_n \mod 1$. Os estados observáveis são os intervalos $[0;\frac{1}{2})$ e $[\frac{1}{2};1)$. Assumimos que o estado inicial é obtido escolhendo-se aleatoriamente um real no intervalo $[0;1)$, de acordo com a distribuição uniforme. Então, o comportamento observável não pode ser previsto melhor que o da seqüência de lançamentos de uma moeda honesta [9]. Observe que a órbita descrita pelo sistema consiste da seqüência de bits da expansão binária do número real escolhido inicialmente.

Embora alguns definam uma seqüência aleatória como uma seqüência caótica [11], estamos propondo uma caracterização de caos significativamente diferente. Devemos considerar o comportamento caótico como sendo um meio termo entre os extremos da reta que une a região de baixa complexidade de Kolmogorov à região de alta complexidade de Kolmogorov. Postulamos que a proximidade com um ou outro extremo é uma boa medida que caracteriza o quão caótico é um sistema. Segundo a nossa abordagem, a proximidade com a região de baixa

complexidade de Kolmogorov representa uma menor irregularidade e aperiodicidade, definindo sistemas dinâmicos lineares ou sistemas dinâmicos não-lineares de comportamento quase caótico. A proximidade com o extremo da região de alta complexidade de Kolmogorov representa sistemas dinâmicos de comportamento muito caótico ou aleatório (não caótico).

Assim, dois aspectos devem ser considerados em nossa caracterização dos sistemas caóticos: irregularidade (aperiodicidade) e previsão (no sentido da previsão por compressão de dados apresentada na Seção 3.3). A primeira está relacionada com o conceito de incompressividade, como visto acima. A segunda está relacionada com o conceito de *segurança*. Uma seqüência é segura se é impossível discerni-la de uma seqüência aleatória gerada por uma moeda honesta (ou seja, ela é *imprevisível*).

Consideremos um sistema caótico simples (gerador de números pseudo-aleatórios) [6]. Ele se baseia em uma generalização do doubling map, $x_{n+1} = cx_n \mod m$, onde $c, m \in \mathbb{N}$. As constantes c e m devem ser escolhidas com critério para que a seqüência gerada seja de boa qualidade. Usualmente o valor inicial da seqüência (semente aleatória) é tomado de algum valor relativamente imprevisível (por exemplo, a hora atual). Sabemos que a seqüência gerada por este algoritmo possui uma pequena complexidade de Kolmogorov, pois sua descrição algorítmica é curta. No entanto, empiricamente é aceito que tais seqüências são de difícil previsão (boa parte das transações comerciais na internet são baseadas em chaves criptográficas geradas de forma muito semelhante).

Nossa abordagem propõe caracterizar tais seqüências pelo quanto são seguras (ou imprevisíveis). A formalização de segurança baseia-se na formalização de previsão por compressão de dados, pois os conceitos são mutuamente contrários. Segundo a nossa abordagem, a dificuldade de prever o comportamento de um sistema caótico é devida à não-computabilidade da complexidade de Kolmogorov (dificuldade de encontrar a descrição mínima).

Tal caracterização apresenta a vantagem de trazer para o campo de sistemas caóticos toda uma série de resultados advindos da área de complexidade de Kolmogorov, que são matematicamente bem fundamentados e expressivos.

5 Conclusão

Apresentamos uma proposta de aplicação da complexidade de Kolmogorov como uma ferramenta para resolver problemas de ciência de sistemas. Nossa abordagem procura caracterizar processos e sistemas, e avaliar modelos computacionais baseado na idéia de compressão de dados (entendida como uma medida do conteúdo de informação).

A metodologia foi aplicada à caracterização do processo de desenvolvimento de software e à comparação de modelos de animação gráfica. Também foi apresentada uma proposta de caracterização de sistemas dinâmicos de comportamento caótico. O método é teorético e mais qualitativo que empírico e quantitativo.

Trabalhos futuros devem cobrir modos de formalizar compressão de dados com perdas (aplicada a formatos como JPEG e MPEG) e meios de aplicar a

metodologia em contextos mais amplos. Finalmente, deveremos em um futuro próximo comparar a metodologia proposta com outras metodologias de ciência de sistemas.

Referências

1. Campani, C. ; Menezes, P. Characterizing the Software Development Process: A New Approach Based on Kolmogorov Complexity. In: *Computer Aided Systems Theory - EUROCAST'2001*, 8th International Workshop on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, feb. 19-23, 2001, Lecture Notes in Computer Science, Springer, v. 2178, p. 242-256, 2001.
2. Campani, C. ; Accorsi, F. ; Menezes, P. ; Nedel, L. Comparing Data Compression in Web-based Animation Models using Kolmogorov Complexity. Submetido para publicação.
3. Chaitin, G. Information-Theoretic Computational Complexity. *IEEE Transactions on Information Theory*, v. 20, p. 10-15, 1974.
4. Chaitin, G. Gödel's Theorem and Information. *International Journal of Theoretical Physics*, n. 22, p. 941-954, 1982.
5. Devaney, R. *An Introduction to Chaotic Dynamical Systems.* Reading: Addison-Wesley, 1989. 336 p.
6. Ditto, W. ; Munakata, T. Principles and Applications of Chaotic Systems. *Commun. ACM*, p. 96-102, nov. 1995.
7. Kolmogorov, A. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*, v. 1, p. 4-7, 1965.
8. Kolmogorov, A. Logical Basis for Information Theory and Probability Theory. *IEEE Transactions on Information Theory*, v. 14, n. 5, p. 662-664, 1968.
9. Li, M. ; Vitányi, P. *An Introduction to Kolmogorov Complexity and its Applications.* 2nd edition, New York: Springer, 1997. 657 p.
10. Solomonoff, R. The Discovery of Algorithmic Probability. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 73-88, aug. 1997.
11. Uspensky, V. ; Semenov, A. ; Shen, A. Can an Individual Sequence of Zeros and Ones Be Random? *Russian Math. Surveys*, v. 45, n. 1, p. 121-189, 1990.
12. Zvonkin, A. ; Levin, L. The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms. *Russian Math. Surveys*, v. 25, n. 6, p. 83-124, 1970.