Sistemas com dados versionados

Thiago Rafael Becker

Dezembro de 2011

Resumo

A semântica de um sistema com dados versionados é estudada a partir de um mapeamento de um grupo de transformações para um grafo direto reflexivo. O grupo de transformações é composto pelas operações adição de caracter e remoção de caracter sobre o conjuntos gerado pelo fecho de Kleene $\Sigma_\perp^\star$. O funtor de mapeamento deste grupo para um grafo reflexivo direto apresenta a estrutura e a semântica de um VCS.

Introdução

Os sistemas com dados versionados (VCS) são sistemas que armazenam um histórico de modificações dos dados que eles contém. Estes sistemas tem diversas funções tais como gerenciamento de documentos, versões de sistemas e configurações destes sistemas para fins de manter histórico de modificações e *rollback* para correção de erros.

O objeto de estudo deste artigo é o cerne da semântica destes sistema, a transição de estados. Para tal, inicialmente vamos definir como são os objetos manipulados por um destes sistemas como um monóide sobre um alfabeto de símbolos, $\Sigma_\perp^\star$. Com essa definição, vamos definir duas operações de adição e remoção de caracteres destes objetos, e provar que estes formam um grupo de transformação, $T_{\Sigma_\perp^\star}$. Por fim vamos definir um funtor $COMMIT$ que mapeia da categoria livremente gerada por $T_{\Sigma_\perp^\star}$ para `RGr`, assim forncendo a estrutura de um grafo para estes sistemas.

Objetos maniulados

Um SCM gerencia estados de objetos. O conteúdo destes objetos é uma sequência de dados pertencentes ao alfabeto $\Sigma_\perp = \Sigma \cup \perp$. Todos os estados de um objeto são definidos pelo conjnto gerado pelo fecho de Kleene sobre o alfabeto $\Sigma_\perp$, $\Sigma_\perp^\star = \langle \Sigma_\perp, \circ, \varepsilon \rangle$. $\perp$ é um elemento especial que indica indefinição das funções parciais sobre o conjunto $\Sigma_\perp^\star$.

Adição e remoção de caracteres

Dois funcionais são definidos sobre $\Sigma_\perp^\star$: *adição de caracter* e *remoção de caracter*. A adição de caracter

$$\oplus_k^c : \mathbb{N} \to \Sigma \to (\Sigma_\perp^\star \to \Sigma_\perp^\star), \oplus_k^c(\perp) = \perp \tag{1}$$

insere o caracter c na posição k, e desloca todos os caracteres a direita de k uma posição à direita. A remoção de caractere

$$\ominus_k^c : \mathbb{N} \to \Sigma \to (\Sigma_\perp^\star \to \Sigma_\perp^\star), \ominus_k^c(\perp) = \perp \tag{2}$$

remove o caracter c da posição k, e move todos os caracteres à sua direita uma poisção para a esquerda. Além disso, a identidade $id : \Sigma_\perp^\star \to \Sigma_\perp^\star$ é definida.

As funções geradas pelos funcionais $\oplus_k^c$ e $\ominus_k^c$ ($\{ger\oplus_k^c\}$, $\{ger\ominus_k^c\}$) formam um grupo de transformação $T_{\Sigma_\perp^\star} = \langle \{ger\oplus_k^c\} \cup \{ger\ominus_k^c\}, \circ, id \rangle$. Para provarmos as propriedades do grupo, definimos o mapeamento $\Sigma_\perp^\star \to \mathbb{N} \cup \{\perp\}$

$$\Gamma = \{\perp_{\Sigma_\perp} \mapsto \perp_{\mathbb{N}\cup\perp}, \varepsilon_{\Sigma_\perp} \mapsto 0_{\mathbb{N}\cup\perp}, c_\Sigma \mapsto n_\mathbb{N}\} \tag{3}$$

tal que o mapeamento é uma bijeção. O mapeamento das operações $\oplus_k^c$ é

$$\Gamma(\oplus_k^c(\sigma)) = \Gamma(\sigma)p_k^{\Gamma(c)}, p_k \in \mathbb{P} \tag{4}$$

e $\ominus_k^c$ é

$$\Gamma(\ominus_k^c(\sigma)) = \Gamma(\sigma)p_k^{-\Gamma(c)}, p_k \in \mathbb{P}, \tag{5}$$

e caso exista $n > 0$ tal que $\Gamma(\ominus_k^c(\sigma))p_k^{-n} \in \mathbb{N}$, então $\ominus_k^c(\sigma) = \perp$. Além disso, se $\Gamma(\ominus_k^c(\sigma)) \notin \mathbb{N}$, então $\ominus_k^c(\sigma) = \perp$.

O mapeamento de $\Gamma(\sigma)$, $\sigma \in \Sigma_\perp^\star$ é tal que

$$\Gamma(\sigma) = \prod_{i \geq 1} p_i^{\Gamma(\sigma_i)}, p_i \in \mathbb{P}, \sigma_i \in \Sigma_\perp^\star \tag{6}$$

Este mapeamento é uma bijeção nas classes de equvalência de $\mathbb{N}$, tal que $\Gamma^{-1}(n) = \Gamma^{-1}(n\prime), n \neq n\prime, n, n\prime \in \mathbb{N}_\perp$, sendo Γ^{-1} o mapeamento inverso a Γ, isto é, o mapeamento de um número para $\Sigma_\perp^\star$. Este mapeamento permite que existam inúmeros números primos entre p_k e p_{k+1} usados para repesentar σ.

Com isso é possível verificar que $\Gamma(\oplus_k^c)$ e $\Gamma(\ominus_k^c)$ formam um grupo de transformação sobre $\mathbb{N} \cup \{\perp\}$, provando a existencia de $T_{\Sigma_\perp^\star}$.

Representação como grafo

Com as informações acima, é possível criar um funtor que implementa a operação *commit*, $COMMIT$. O funtor é tal que

$$COMMIT(\Sigma_\perp^\star) = \sigma_1, \sigma_1 \subset \Sigma_\perp^\star \tag{7}$$

$$COMMIT(\{ger\oplus_k^c\} \cup \{ger\ominus_k^c\} \cup \{id\}) = f : \sigma_1 \to \sigma_1\prime, \sigma_1, \sigma_1\prime \in \Sigma_\perp^\star \tag{8}$$

em que σ_1 é um conjunto unário. $COMMIT$ constrói a estrutura do repositório, e é inversível. Intuitivamente, este funtor toma um conjunto σ_1 e uma composição de morfismos em $\{ger\oplus_k^c\} \cup \{ger\ominus_k^c\} \cup \{id\}$ que transforma o estado do repositório.

O grafo que representa um dado repositório é um subgrafo *Repo* do grafo direto reflexivo

$$\langle COMMIT(\Sigma_\perp^\star), COMMIT(\{ger\oplus_k^c\} \cup \{ger\ominus_k^c\} \cup \{id\})\rangle,$$

e é simples verificar as propriedades compositiva e associativa das setas $COMMIT(\{ger\oplus_k^c\} \cup \{ger\ominus_k^c\} \cup \{id\})$ pelo mapeamento Γ acima. Desta forma, a categoria `Repo`

$$\langle COMMIT(\Sigma_\perp^\star), COMMIT(\{ger\oplus_k^c\} \cup \{ger\ominus_k^c\} \cup \{id\}), \delta_0, \delta_1, id, \circ\rangle$$

é a categoria livremente gerada pelo grafo *Repo*. O objeto inicial de toda a categoria `Repo` é a palavra vazia ε, que representa um repositório vazio.

Conclusões e desenvolvimentos futuros

O funtor $COMMIT$ da categoria livremente gerada de $T_{\Sigma_\perp^\star}$ para `Repo` representa o cerne da semântica dos sistema com dados versionados. O trabalho é incompleto, entretanto: não trata de sistemas que possuam *merging* e *branching*, nem de metadados comumente presentes nestes sistemas.

É possível perceber que a categoria `Repo` apresenta uma estrutura semelhante ao de um reticulado. Analogamente aos conjuntos parcialmente ordenados vistos como categorias, *branching* poderia ser visto como um produto de dois objetos, e *merging* como o coproduto de dois objetos, mas isto são coisas a se estudar.

Metadados podem ser estudados como endomorfismos etiquetados, porém a construção dos funtores para a criação e movimento destes metadados pode ser complexa. Isso talvez possa ser possível através de uma gramática de grafos sobre *Repo*.

Além disso, o estudo pode se estender para o estudos de todos os sistemas versionados e seus funtores, para verificar que tipo de estrutura tem estes.