5
Formalização do Modelo de Dados

5.1
Introdução

Neste capítulo nós formalizamos o modelo de dados *BioConceptual* . A necessidade da formalização do modelo se deve ao fato de que apesar do modelo proposto ser apropriado para a compreensão humana dado que utiliza uma notação gráfica e fácil de ser gerado visto que não exige a compreensão de uma sintaxe complicada. Apesar dessas vantagens, um modelo baseado em representações gráficas, apresentam um semântica que não é precisa dependendo da verificação humanos, tornando-se desta forma impraticável o seu entendimento por máquinas e consequentemente impedindo a automatização de inferências e raciocínio sobre o modelo gerado. Em resumo, um modelo somente gráfico possui a desvantagem de possuir uma expressividade limitada.

Inicialmente apresentamos o metamodelo do *BioConceptual* usando um diagrama de classes UML. Em seguida explicamos sucintamente cada um dos construtores e formalizamos cada um deles utilizando lógica de primeira ordem. Na última seção deste capítulo apresentamos algumas inferências que podem ser realizadas utilizando a formalização proposta.

5.2
Uma visão geral do meta-modelo do BioConceptual

Como definido anteriormente, *BioConceptual* é um modelo orientado a objetos baseado no modelo ODMG. Esta escolha foi justificada pela capacidade dos modelos orientados a objetos em representar objetos complexos e pela sua característica de estar orientado para implementação. Realizamos extensões nesse modelo e alteramos alguns construtores de tal forma a gerar um meta-modelo homogêneo.

Um esquema de dados *BioConceptual* é formado por um conjunto de tipos de percepções e por um conjunto de tipos de construtores definidos respectivamente pelas classes PerceptionType e ConstructType apresentadas na Figura 5.1. As percepções formam uma hierarquia baseada em ligações do tipo "É-UM" e a relação entre uma percepção e os tipos de construtores é realizada através das percepções do nível folha da hierarquia. Cada tipo de construtor *BioConceptual* pode ser um tipo de objeto, um tipo de propriedade, um tipo de domínio ou um tipo de restrição. O uso de elementos mais gerais

com a superclasse ConstructType permite tratar todos elementos do*BioConceptual* de uma forma homogênea. Esta é uma característica importante que permite, por exemplo, definir restrições sobre todos os elementos incluindo atributos e relacionamentos. O relacionamento entre as classes ConstructType e ConstraintType permite que as restrições sejam aplicadas sobre todos os tipos de construtores.

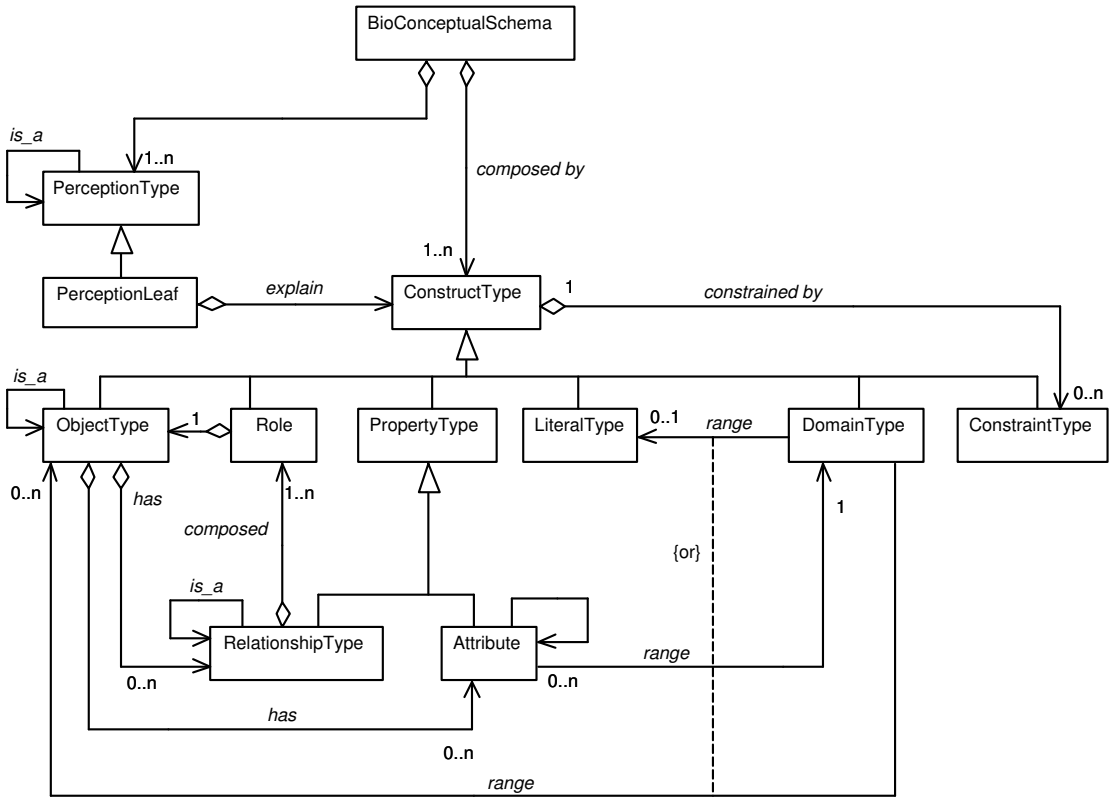

Figura 5.1: *BioConceptual* Tipos de Construtores

5.3 Abordagem para Formalização

Utilizaremos a lógica de primeira ordem para realizar a formalização do modelo *BioConceptual* . A adoção da lógica de primeira ordem se deve ao fato de que ela é uma linguagem que apresenta as seguintes propriedades:

- Apresenta uma semântica precisa;
- É possível realizar uma verificação formal do modelo através de mecanismos de inferência;
- A lógica de primeira ordem pode ser facilmente descrita em uma linguagem que pode ser compreendida por uma máquina, como por exemplo, uma linguagem como Description Logics;
- É uma linguagem de expressividade ilimitada permitindo a representação de todos os elementos do modelo.

A adoção de uma abordagem mista para formalização de modelo de dados, i.e., associar semântica formal para os construtores do diagramas de projeto permite tirar vantagem das metodologias desenvolvidas em engenharia de software para o projeto de aplicações e usar todo o arcabouço desenvolvido para teorias lógicas quando necessário, i.e., compreensão formal, verificação, correção, raciocínio automatizado, inferências sobre o modelo, etc. Além disso, agregamos o valor herdadado dos diagramas conceituais (e.g. facilidade de compreensão humana) com tarefas de raciocínio, as quais são necessárias para verificação formal e manipulação pelas máquinas.

Um importante ponto a destacar é que podemos obter, através dos diagramas conceituais, teorias lógicas de uma forma específica e com expressividade bem comportada. Desta forma, podemos explorar um característica particular desta teoria lógica para simplificar as inferências, alcançando decidibilidade e procedimentos de raciocínio com complexidade computacional adequada.

Sendo assim, usaremos como linguagem gráfica os diagramas do *BioConceptual* representando os diagramas conceituais e como linguagem descritiva a lógica de primeira ordem, a qual servirá para capturar a semântica e o raciocínio do esquema de dados criado. Além disso, pode ser usada a linguagem Description Logics para entender as propriedades computacionais do raciocínio.

5.3.1 Mapeamento para Lógica de Primeira Ordem

Um esquema *BioConceptual* é usado para representar explicitamente as informações de um domínio de interesse, as quais devem ser persistidas em um banco de dados. No caso específico do modelo de dados *BioConceptual* , é adequado para o domínio da biologia molecular. Note que este é exatamente o objetivo de todos os formalismos conceituais, tais como esquemas entidade-relacionamento ou ontologias (agora em voga para devido o estudo da web semântica)

Entendemos desta forma que o meta-modelo do *BioConceptual* representa o nosso alfabeto que será usado para definir um esquema de dados. Cada elemento deste esquema pode ser visto como um predicado lógico que relaciona um elemento do meta-modelo (intensão) com as suas instâncias (extensão). Por exemplo, o conceito Gene que é um tipo de objeto pode ser sintaticamente definido por Gene(x), onde x representa as possíveis instâncias deste conceito.

De forma semelhante, um atributo também será visto como um predicado lógico relacionando o tipo de objeto que ele compõe com o tipo de dado deste atributo. Visto que atributos também podem possuir multiplicidades diferentes de um para um. Desta maneira, os atributos podem ser melhor representados por predicados binários, com assertivas para representar os tipos de dados e multiplicidade. Por exemplo, dado um atributo a de uma classe C com o tipo de dado T e multiplicidade "i..j", nós mapeamos isto em lógica de primeira ordem como um predicado binário $a(x, y)$ com as seguintes assertivas:

– Assertiva para o tipo de dado do atributo:
$\forall$ **x, y.(C(x)** $\wedge$ **a(x,y)** $\supset$ **T(y)**

– Assertiva para a multiplicidade:
$\forall$ **x.(C(x)** $\supset$ **(i** $\leq$ **#{y** | **a(x,y)}** $\leq$ **j)** [1]

Um exemplo mais concreto da aplicação deste mapeamento é apresentado abaixo, onde a definição do tipo de objeto Gene, apresentado na Figura 5.2, é transformado em um conjunto de fórmulas.

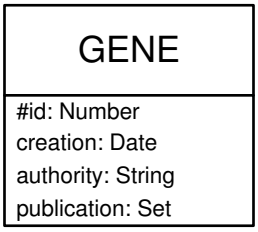

Figura 5.2: Representação gráfica do conceito Gene

Axiomas

$\forall$ **x, y.(Gene(x)** $\wedge$ **id(x,y))** $\supset$ **Number(y)**
$\forall$ **x.(C(x)** $\supset$ **(i** $\leq$ **#{y** | **id(x,y)}** $\leq$ **1)**

$\forall$ **x, y.(Gene(x)** $\wedge$ **creation(x,y))** $\supset$ **Date(y)**
$\forall$ **x.(C(x)** $\supset$ **(i** $\leq$ **#{y** | **creation(x,y)}** $\leq$ **1)**

$\forall$ **x, y.(Gene(x)** $\wedge$ **authority(x,y))** $\supset$ **String(y)**
$\forall$ **x.(C(x)** $\supset$ **(i** $\leq$ **#{y** | **authority(x,y)}** $\leq$ **1)**

$\forall$ **x, y.(Gene(x)** $\wedge$ **publication(x,y))** $\supset$ **Set(y)**
$\forall$ **x.(C(x)** $\supset$ **(i** $\leq$ **#{y** | **publication(x,y)}** $\leq$ **100)**

Relacionamentos entre os tipo de objetos é modelado no *BioConceptual* com tipos de relacionamentos. Como já foi definido, um relacionamento entre dois tipos de objetos define uma relação entre duas ou mais instâncias das classes participantes. Os tipos de relacionamentos modelam propriedades das classes que não são locais, visto que eles envolvem outras classes. Desta forma, um relacionamento entre duas classes é uma propriedade e ambas as classes. Nós podemos representar um relacionamento n-ário entre as classes $C1, ..., Cn$ com um predicado R em lógica de primeira ordem. Desta forma, devemos garantir que os componentes do predicado deve pertencer às classes corretas, como apresentado a seguir:

$\forall\ x_1,...,x_n.\mathbf{A}(x_1,...,x_n) \supset C_1(x_1) \wedge ... \wedge C_n(x_n)$

[1] Esta é uma representação reduzida para a fórmula em lógica de primeira ordem para representar a cardinalidade dos possíveis valores para y

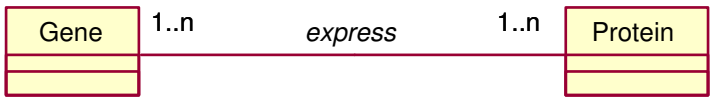

Figura 5.3: Um diagrama mostrando um tipo de relacionamento ligando dois tipos de objetos

A Figura 5.3 ilustra um relacionamento entre dois tipos de objetos Gene e Protein. Este relacionamento de acordo com o nosso mapeamento poderia ser descrito da seguinte forma:

$\forall\, x_1,x_2.\text{express}(x_1,x_2) \wedge \text{Gene}(x_1) \supset \text{Protein}(x_2)$

Da mesma forma que os atributos, relacionamento binários podem possuir cardinalidade associada ao relacionamento.

Os tipos de relacionamento também são mapeados para predicados. As multiplicidades de relacionamentos binários podem ser expressas em lógica de primeira ordem da seguinte maneira:

$$\forall\, x_1.C_1(x_1) \supset (min_1 \leq \#\{x_2 \mid \mathbf{R}(x_1,x_2)\} \leq max_1)$$
$$\forall\, x_2.C_2(x_2) \supset (min_2 \leq \#\{x_1 \mid \mathbf{R}(x_1,x_2)\} \leq max_2)$$

A representação completa relativa à Figura 5.3 será então expressa da seguinte forma:

Axiomas

$\forall\, x_1,x_2.\text{express}(x_1,x_2) \wedge \text{Gene}(x_1) \supset \text{Protein}(x_2)$

$\forall\, x_1.\text{Gene}(x_1) \wedge (1 \leq \#\{x_1 \mid \text{express}(x_1,x_2)\}\ )$

$\forall\, x_2.\text{Protein}(x_2) \wedge (1 \leq \#\{x_1 \mid \text{express}(x_1,x_2)\}\ )$

Um importante elemento para modelagem conceitual é a ligação "É-UM" pois ela permite classificar, especializando ou generalizando as tipo de objetos no esquema de dados. No *BioConceptual* a ligação "É-UM" é modelada usando um pequeno triângulo como símbolo gráfico, o mesmo usado pela UML. A Figura 5.4 apresentada abaixo ilustra o uso desta notação para especificar uma representação de especialização do conceito denominado Gene, o qual é especializado em genes humanos e de rato, respectivamente representados pelas classes Human e Mouse.

A semântica deste tipo de relação é a seguinte: se uma classe C' é subclasse da classe C, então toda instância da classe C' é também uma instância da superclasse C. A capacidade de associarmos mais subclasses a uma classe permite a definição de restrições de integridade sobre as classes envolvidas nessas relações. O uso mais comum de restrições são:

– Disjuntas: determina que diferentes subclasses não podem ter instâncias em comum, i.e., um objeto não pode ser instância de duas subclasses ao mesmo tempo;

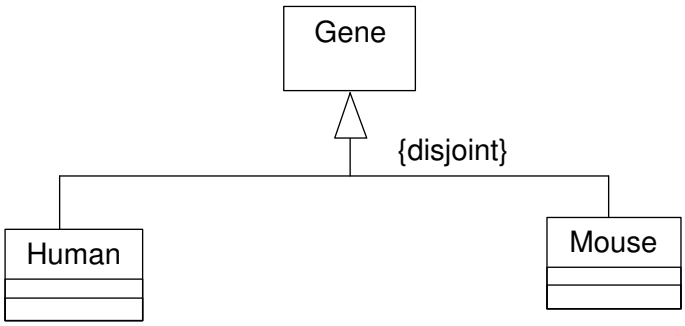

Figura 5.4: Um exemplo de ligação É-UM

– Completa ou cobertura: afirma que toda instância pertencente à superclasse obrigatoriamente deve pertencer a pelo menos uma das subclasses;

Levando em consideração essas características na definição de uma ligação "É-UM", a formalização em lógica de primeira ordem pode ser realizada da seguinte forma:

– Ligação "É-UM": $\forall x.C_i(x) \supset C(x)$ para $i = 1, ..., n$
– Restrição de Disjunção: $\forall x.C_i(x) \supset C_j(x)$ para $i \neq j$
– Completa: $\forall x.C(x) \supset \bigvee_{i=1}^{n} C_i(x)$

Aplicando a formalização acima no exemplo apresentado na Figura 5.4 teremos como resultado os seguintes axiomas:

Axiomas para ligações É-UM
$\forall x.Human_i(x) \supset Gene(x)$
$\forall x.Mouse_i(x) \supset Gene(x)$
$\forall x.Human_i(x) \supset \neg Mouse(x)$

O *BioConceptual* dá ênfase para o construtor *Constraint* o qual pode ser aplicado a qualquer outro construtor. Como havíamos comentado na seção 4.9 do capítulo 4 a linguagem utilizada para este construtor é a lógica de primeira ordem[2]. Desta forma, o construtor Constraint pode ser formalizado através de um predicado e fórmulas lógicas que representem a restrição definida pelo usuário.

Na Figura 5.5 é apresentado um exemplo do uso de duas restrições. A restrição C1 se aplica a um auto-relacionamento sobre a entidade ADNSequence denominado homologue. A segunda restrição denominada C2 se aplica diretamente à entidade ADNSequence. Neste caso, as restrições C1 e C2 seriam mapeadas para lógica de primeira ordem da seguinte forma:

[2]Esta não seria a linguagem ideal para ser usada pelo projetista. No entanto qualquer linguagem que pudesse ser traduzida para a lógica de primeira ordem poderia ser aceita. Dado as limitações de tempo deste trabalho, optamos por não definir uma linguagem amigável para o usuário assim como não repensamos a notação gráfica a ser utilizada.

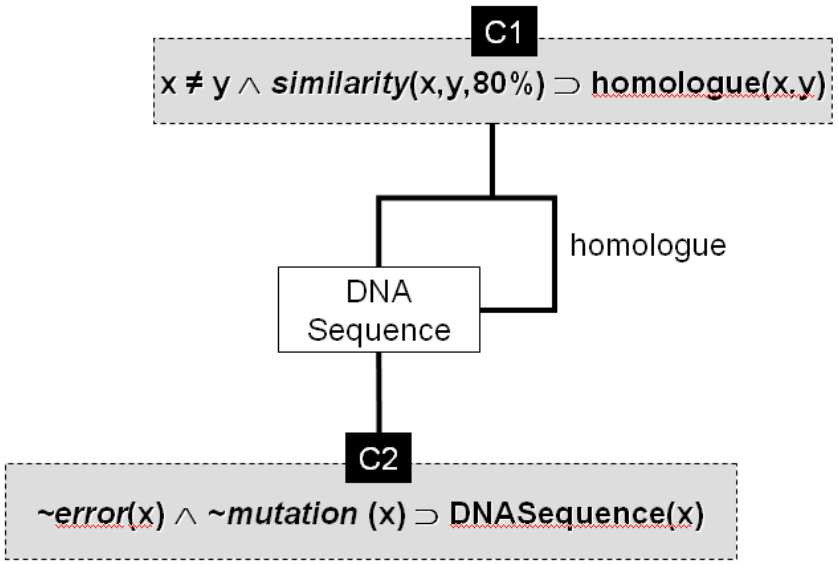

Figura 5.5: Exemplo de uso do construtor *Constraint*

Axiomas para as Restrições C1 e C2

$\forall x, y.(x \neq y \wedge similarity(x, y)) \supset homologue(x, y)$

$\forall x.(\sim error(x) \wedge \sim mutation(x)) \supset ADNSequence(x, y)$

A maioria dos tipos de restrição poderia ser descrita usando o construtor Constraint, no entanto o projetista deve utilizar esse construtor para definir restrições que não sejam possíveis de representar graficamente ou quando a representação gráfica possa poluir o modelo, tornando-o de difícil leitura.

5.4 Realizando Inferências ao Modelo

Apesar de alguns modelos conceituais existentes possuírem ferramentas que auxiliam o projetista a criar seus esquemas, essas ferramentas não apresentam mecanismos para realizar a consistência dos modelos criados. O fato de termos mapeado um esquema do BioConceptual para um conjunto de fórmulas em lógica de primeira ordem, nos permite conferir formalmente as propriedades relevantes dos esquemas gerados. Desta forma podemos garantir a qualidade dos esquemas de acordo com um critério de qualidade definido.

No domínio da biologia, a formalização do modelo é fundamental visto que o projetista que é em geral um cientista da área não possui especialização na área de modelagem de dados. Neste caso, os esquemas criados podem gerar problemas de performance ou mesmo de inconsistência dos dados ao serem implementados em um SGBD.

Algumas inferências interessantes que podem ser realizadas sobre um modelo conceitual podem ser:

- Consistência dos tipos de objetos
- Consistência do esquema em geral

- Consistência da relações de subjugamento (subsumption)
- Equivalência entre classes

Uma tipo de objeto é consistente se o esquema de dados admite uma instanciação na qual o tipo de objeto tem um conjunto não-vazio de instâncias. Intuitivamente, a classe pode ser populada sem violar as condições impostas pelo esquema. Por sua vez, a inconsistência de um tipo de objeto enfraquece a compreensão do esquema e é uma indicação de erro. Um a vez detectado este erro, o projetista pode remover a inconsistência através do relaxamento de condições, ou até mesmo apagar o tipo de dado.

Seja Γ o conjunto de todas as assertivas relativas a um esquema BioConceptual e $C(x)$ o predicado relativo ao tipo de dado C. Então C é consistente se e somente se :

$$\Gamma \models \forall x.C(x) \supset true$$

A fórmula acima determina que deve existir um modelo deΓ cuja extensão não é um conjunto vazio.

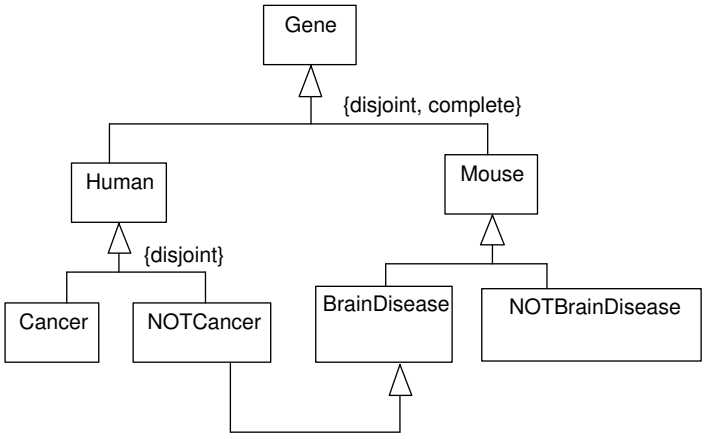

Figura 5.6: Esquema Conceitual apresentando Inconsistências

A Figura 5.4 apresenta um exemplo onde o esquema criado possui uma inconsistência. Realizando uma análise no diagrama apresentado podemos perceber que um gene humano pode ser do tipo Cancer ou NOTCancer, por outro lado, um gene de Mouse pode ser do tipo Obesidade ou NOTBrainDisease. Além disto, existe uma relação de subjugamento (usando ligação "É_UM") entre as classes BrainDisease e NOTCancer. Podemos perceber que a classe NOT_Cancer não poderá possuir nenhuma instância visto que por definição um gene deve ser Human e Mouse e nunca os dois simultaneamente. Porém, no diagrama apresentado, sugere que uma instância de NOTCancer possa ser também uma instância de BrainDisease e consequentemente de Mouse.

A realização de inferências sobre a teoria lógica representada pelo modelo resultaria nas seguintes consequências lógicas:

$$\Gamma \models \forall x.NOTCancer(x) \supset false$$
$$\Gamma \models \forall x.Human(x) \supset Cancer(x)$$
$$\Gamma \models \forall x.Cancer(x) \equiv Human(x)$$

Neste exemplo podemos detectar os quatro tipos de inferências definidas anteriormente que podem ser obtidas através de mecanismos de raciocínio sobre o esquema conceitual.

Na primeira consequência lógica podemos verificar que o tipo de objeto NOTCancer não possuirá instâncias. Na segunda consequência lógica, todas as instâncias de Human são também instâncias de Cancer. Finalmente, deduzimos que o tipo de objeto Cancer é equivalente ao tipo de objeto Human. Neste caso, projetista poderia optar entre redefinir as relações ou eliminar, por exemplo, os tipos de objetos Cancer e NOTCancer.

Apesar da simplicidade do exemplo, não é trivial deduzir todas essas consequências lógicas apenas visualizando o esquema criado. Nos casos onde a hierarquia utilizada é muito grande problemas como este podem ser difíceis de serem detectados.

Em resumo, um esquema conceitual é consistente se ele admite uma instanciação, i.e., se seus tipos de dados podem ser populados sem violar qualquer uma das condições impostas pelos diagramas. Note que a extensão vazia para todos os tipos de objetos não é considerada uma instanciação admissível. Então, o diagrama é consistente se pelo menos uma de seus tipos de objetos admite extensão não-vazia. Quando o diagrama não é consistente, as definições em conjunto são contraditória, desde que ele não permite qualquer tipo de objeto ser populado.

Formalmente falando, seja Γ um conjunto de assertivas em lógica de primeira ordem correspondentes à um esquema BioConceptual. Então o esquema é consistente se somente se Γ é satisfeito, i.e., Γ admite no mínimo um modelo. Lembrando que satisfiabilidade é o mesmo que redutível a seguinte implicação lógica:

$$\Gamma \not\models false.$$

De acordo com os exemplos que vimos anteriormente, podemos concluir que a formalização do modelo nos fornece diversas vantagens em termos de explicitação da semântica inserida no esquema de dados. Desta forma, é possível utilizar a teoria lógica que representa o modelo processamento por uma máquinas. O resultado deste processamento, poderá fornecer informações importantes para elucidar problemas apresentados na modelagem, tais como: inconsistências nos conceitos apresentados, contradições na modelagem e subjugamentos incorretos. Acreditamos que um modelo de dados conceitual deve auxiliar o projetista do banco no momento durante construção do esquema de dados, podendo ser processado por máquinas de forma a garantir a correção do esquema a ser gerado.