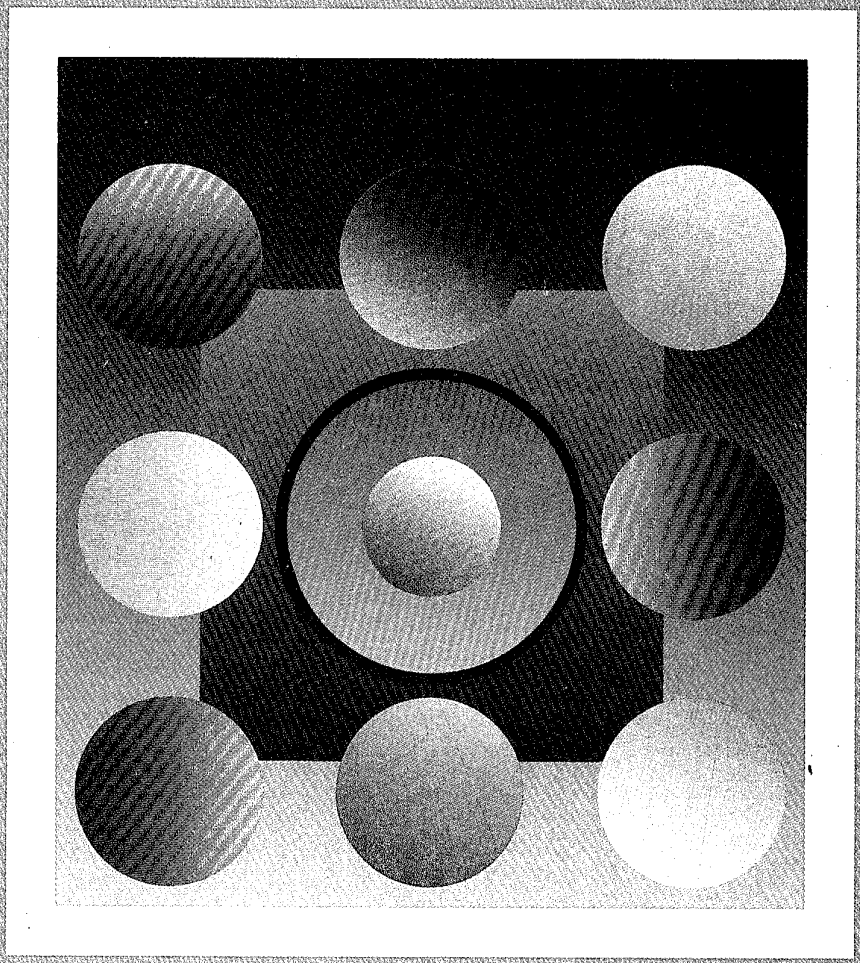
REVISTA
BRASILEIRA
DE COMPUTAÇÃO

RBC

ISSN 0101-0883

Uma publicação da SBC SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

1982, Vol. 2, Nº 3



NESTE NÚMERO

SISTEMAS DISTRIBUÍDOS
BANCO DE DADOS
MONITOR GRÁFICO
ÍNDICES DA RBC

EDITORA CAMPUS

RBC, Rio de Janeiro, V. 2, nº 3, 1982, pp. 151 a 167

UMA TEORIA DE DEPENDÊNCIAS FUNCIONAIS E DE INCLUSÃO

S. P. REGO
M. A. CASANOVA
DEPARTAMENTO DE INFORMÁTICA – PUC/RJ
RUA MARQUÊS DE SÃO VICENTE, 225
CEP 22453 – RIO DE JANEIRO – RJ – BRASIL

SUMÁRIO

Um sistema formal S para dependências funcionais e de inclusão é apresentado. As dependências consideradas são definidas sobre expressões relacionais envolvendo os operadores de projeção, junção natural, união e seleção. A consistência e completude de S é obtida recorrendo-se ao método dos tableaux analíticos. O sistema possui aplicações interessantes ao projeto de banco de dados, como a normalização de esquemas relacionais e integração de subesquemas.

ABSTRACT

A formal system S for functional and inclusion dependencies is presented. The dependencies considered are defined over relational expressions involving projection, natural join, union and selection operators. The soundness and completeness of S are verified using the analytic tableaux method. The system has interesting applications to database design, such as normalization of relational schemas and subschema integration.

1. INTRODUÇÃO

Este trabalho apresenta um sistema formal S baseado em dependências funcionais (abrev. DF) e dependências de inclusão (abrev. DI) definidas sobre expressões relacionais envolvendo os operadores de projeção, junção natural, união e seleção. A classe de DFs a ser considerada compõe-se de expressões da forma $e: X \rightarrow Y$ que indica que Y é dependente funcionalmente de X na relação denotada pela expressão relacional e [Ca]. Da mesma forma, uma DI é da forma $e \subset f$ e indica que a relação denotada por e é um subconjunto da relação denotada por f.

A demonstração da consistência e da completude de S será feita recorrendo-se ao método dos tableaux analíticos [Sm].

O sistema S provê uma base teórica para o desenvolvimento de uma metodologia de projeto de banco de dados que pode eventualmente substituir a normalização. Esta observação é justificada como se segue. Alguns dos trabalhos em normalização dependem de uma teoria de DFs baseada na suposição da relação universal (ver [BBG]), a qual requer que todas as relações em um banco de dados sejam projeções de uma única (universal) relação. Esta suposição ajuda a formalizar a noção de equivalência de esquemas e assegura a adequação das regras de inferência de DFs [Ar]. Contudo, ela não ocorre na prática e é incompatível com alguns dos propósitos de normalização [BG]. As dependências de inclusão substituem esta suposição no sentido de que elas permitem definir seletivamente quais os dados que devem ser duplicados em quais relações. Isto é especialmente útil quando uma relação é dividida em várias pelo processo de projeto. Portanto, a teoria de DFs sob a suposição da relação universal pode ser substituida com sucesso por uma teoria de DFs e DIs.

A organização do trabalho é a seguinte. A seção 2 define DFs e DIs sobre expressões relacionais. A seção 3 descreve o sistema formal S e o método dos tableaux analíticos. A seção 4 prova a consistência e completude de S. A seção 5 apresenta conclusões e direções para pesquisa futura.

2. LINGUAGENS DE DEPENDÊNCIAS FUNCIONAIS E DEPENDÊNCIAS DE INCLUSÃO

Esta seção define uma família de linguagem formais que nós chamamos linguagens de dependências funcionais e de dependências de inclusão (DF-DI). As fórmulas de uma linguagem DF-DI são essencial

mente combinações booleanas de DFs e DIs e fórmulas da forma $e(\bar{c})$ ou $\bar{c} = \bar{d}$, onde e é uma expressão relacional e $\bar{c}$, $\bar{d}$ são tuplas de constantes.

Chamamos de atributos as colunas distintas das relações e denotamos por ATR(r) a sequência de atributos distintos de r. Se X, Y são sequências, X ||Y (ou simplesmente X Y) indica a concatenação de X e Y, COMP(X) denota o comprimento de X e X ⊂ Y indica que todos os atributos de X também ocorrem em Y.

Uma linguagem DF-DI L contém os seguintes símbolos:

Parâmetros:

- nomes de relações: para cada n > 0, um conjunto não vazio de nomes de relações; cada relação r possui uma sequência ATR(r) de atributos (se COMP(ATR(r))=n, diz-se que a relação é n-ária).
- símbolos de constantes: um conjunto de símbolos não vazio, distinto do conjunto acima.

Símbolos lógicos:

- os conectores usuais e símbolos especiais: $\neg, \wedge, \vee, \Rightarrow, (,), [,], \rightarrow, :, \subset$
- igualdade entre constantes: =
- os operadores de relação: *, ∪

Uma expressão relacional de L ou termo ou simplesmente uma expressão, e os atributos de uma expressão são definidos indutivamente como se segue (se uma expressão tem n atributos diz-se que a expressão é n-ária).

- um nome de relação é uma expressão (atômica);
- se e é uma expressão e T é uma subsequência de ATR(e), a projeção e[T] é uma expressão e ATR(e[T]) = T;
- se e e f são expressões então a junção natural $e * f$ é uma expressão onde

$$ATR(e * f) = ATR(e) || \sigma$$

  onde σ é ATR(f) sem os atributos comuns a ATR(e);
- se e e f são expressões onde ATR(e) = ATR(f), então a união ($e \cup f$) é uma expressão e

$$ATR(e \cup f) = ATR(e) = ATR(f);$$

- se e é uma expressão, X é uma subsequência de ATR(e) e $\bar{d}$ uma tupla de constantes, tal que COMP(X) = COMP($\bar{d}$) então a seleção e[X = $\bar{d}$] é uma expressão e

$$ATR(e[X=\bar{d}]) = ATR(e)$$

Uma fórmula atômica de L é da forma $\bar{a} = \bar{b}$, $e(\bar{a})$ uma dependência funcional $e : X \rightarrow Y$ ou uma dependência de inclusão $e \subset \delta$, onde $\bar{a}$ e $\bar{b}$ são tuplas de constantes n-árias, e e δ são expressões n-árias tais que

$$COMP(ATR(e)) = COMP(ATR(\delta)) = n$$

e X, Y são subsequências de $ATR(e)$, $n > 0$. Uma fórmula de L é uma fórmula atômica ou é da forma $\neg P$, $(P \wedge Q)$, $(P \vee Q)$ ou $(P \Rightarrow Q)$, onde P, Q são fórmulas.

Em essência, podemos falar sobre combinações booleanas de DFs e DIs sobre expressões relacionais em L. Nós também incluimos as fórmulas atômicas da forma $\bar{a} = \bar{b}$ e $e(\bar{a})$, as quais afirmam a igualdade de tuplas de constantes e a existência de uma tupla em uma relação denotada por uma expressão, respectivamente. Isto torna-se necessário devido a natureza das regras do sistema formal que desenvolveremos na próxima seção.

Uma estrutura I com domínio D_I para L é uma função atribuindo a cada nome de relação n-ária r de L uma relação n-ária $I(r) \in D_I^n$, $n > 0$, e a cada tupla k-ária de constantes $\bar{a}$ uma tupla $I(\bar{a}) \in D_I^k$, $k > 0$. I é estendido para expressões relacionais de L como se segue (note que I já é definido para as expressões atômicas).

$$I(e[T]) = \{\bar{a}_T \mid \bar{a} \in I(e)\} \tag{1}$$

$$I(e \cup \delta) = \{\bar{a} \mid \bar{a} \in I(e) \vee \bar{a} \in I(\delta)\} \tag{2}$$

$$I(e[X=\bar{d}]) = \{\bar{a} \mid \bar{a} \in I(e) \wedge \bar{a}_X = I(\bar{d})\} \tag{3}$$

$$I(e * \delta) = \{\bar{c} \mid \exists\, \bar{a}\, \exists\, \bar{b}(e(\bar{a}) \wedge \delta(\bar{b}) \wedge \bar{a}_W = \bar{b}_W \wedge \bar{a} = \bar{c}_X \wedge \bar{b} = \bar{c}_Y)\} \tag{4}$$

onde $X = ATR(e)$, $Y = ATR(\delta)$ e W são os atributos comuns a X e Y e onde, dado uma tupla $\bar{d}$ de uma relação r e uma sequência de atributos Z, $\bar{d}_Z$ denota os valores de $\bar{d}$ para os atributos em Z. Se duas tuplas $\bar{d}$ e $\bar{h}$ têm os mesmos valores em todos os atributos da sequência Z em comum, então dizemos que $\bar{d}_Z = \bar{h}_Z$

Estendemos I a uma valorização booleana das fórmulas de L como se segue

1) $I(\bar{a} = \bar{b})$ = Verdadeiro $\Leftrightarrow$ $I(\bar{a}) = I(\bar{b})$, caso contrário $I(\bar{a} = \bar{b})$ = Falso
2) $I(e(\bar{a}))$ = Verdadeiro $\Leftrightarrow$ $I(\bar{a}) \in I(e)$, caso contrário $I(e(\bar{a}))$ = Falso
3) $I(e : X \rightarrow Y)$ = Verdadeiro $\Leftrightarrow$

$\forall \bar{a}\ \forall \bar{b}(\bar{a} \in I(e) \wedge \bar{b} \in I(e) \wedge \bar{a}_X = \bar{b}_X \Rightarrow \bar{a}_Y = \bar{b}_Y)$,
caso contrário $I(e : X \to Y)$ = Falso

4) $I(e \subset \mathfrak{f})$ = Verdadeiro sss $\forall \bar{a}(\bar{a} \in I(e) \Rightarrow \bar{a} \in I(\mathfrak{f}))$, caso contrário $I(e \subset \mathfrak{f})$ = Falso

5) I é estendido para fórmulas não-atômicas usando as regras do Cálculo Proposicional.

O significado das fórmulas de L deve ser claro, nós somente enfatizamos que $e : X \to Y$ é verdadeiro em I sss as colunas Y da relação denotada por e em I são funcionalmente dependentes das colunas X. De maneira semelhante, $e \subset \mathfrak{f}$ é verdadeiro em I sss $I(e)$ é subconjunto de $I(\mathfrak{f})$.

Nós concluimos nossa lista de definições básicas dizendo que um conjunto P de fórmulas é satisfatível sss todas as fórmulas de P são verdadeiras em alguma estrutura de L. Um conjunto P de fórmulas implica logicamente uma fórmula P (P é uma consequência lógica de P) sss P é verdadeira em cada estrutura de L onde todas as fórmulas de P são verdadeiras (escreve-se $P \models P$). Finalmente, P é uma tautologia sss P é verdadeira em todas as estruturas de L (escreve-se $\models P$, uma vez que P é uma tautologia sss P é implicado logicamente pelo conjunto vazio de fórmulas).

3. UM SISTEMA FORMAL PARA DEPENDÊNCIAS FUNCIONAIS E DEPENDÊNCIAS DE INCLUSÃO

Nesta seção apresentamos um sistema formal S, cuja linguagem L é uma linguagem DF-DI. Ainda, nesta seção, apresentaremos um procedimento de prova para S, tal que uma fórmula P de L é consequência lógica de um conjunto P de fórmulas de L se e somente se P é um teorema de P em S. Este resultado é provado na seção 4. Inicialmente, descrevemos o procedimento de prova, uma vez que as regras de S dependem dele. Este procedimento de prova, denominado método dos tableaux analíticos é uma variante dos "tableaux semânticos" de Beth e dos métodos de Hintikka [Sm]. A noção de uma prova em S é uma generalização direta do método dos tableaux para o Cálculo Proposicional [Sm]. Este procedimento formaliza a seguinte estratégia para provar que $P \models P$. Partindo de P e $\neg P$, analisa-se todos os casos possíveis. Se todos os casos conduzem a uma contradição, então $P \cup \{\neg P\}$ é insatisfatível e $P \models P$. Por outro lado, se a análise de algum caso termina sem chegar a uma contradição, então $P \cup \{\neg P\}$ é satisfatível e portanto, $P \not\models P$ não ocorre. Isto tem que ser pro-

vado, uma vez que não é uma propriedade imediata do sistema.

O raciocínio por casos é feito usando regras do seguinte tipo

$$R_i \cdot \frac{P_i}{Q_{i1} | \ldots | Q_{in_i}} \quad \text{onde } P_i \text{ e } Q_{ij} \quad (1 \leq j \leq n_i)$$

são conjuntos finitos de fórmulas. Intuitivamente, R_i quer dizer que a partir de P_i podemos derivar todas as fórmulas em Q_{ij} , para algum $j \in [1, n_i]$. Chamamos P_i o <u>antecedente</u> de R_i e $Q_{i1}, \ldots, Q_{in_i}$, os <u>consequentes</u> de R_i. Uma prova por análise de casos pode ser organizada como uma árvore, onde os filhos de um nó correspondem a casos de ramificação. Uma prova termina quando cada ramo contém uma contradição ou não pode continuar a ser estendido sem repetição, onde cada ramo de uma árvore significa um caminho desde a raiz até uma folha.

Um <u>tableau</u> é um conjunto de elementos, chamados nós, parcialmente ordenados e classificados em <u>níveis</u> como explicado abaixo. A cada nó está associado um conjunto finito de fórmulas.

Cada nó pertence a um único nível, que é identificado por algum número natural. Existe um único nó de nível 0, chamado nó <u>inicial</u> (origem) do tableau. Cada nó de nível n+1 é um sucessor de um único nó, o qual deve ser de nível n. Um nó é uma <u>folha</u> se ele não tem sucessores.

Se um tableau tem pelo menos um nó de nível h mas nenhum outro nó de nível maior que h, dizemos que h é a altura do tableau.

Um <u>ramo</u> de um tableau é uma sequência finita de nós $\theta_0, \ldots, \theta_k$ tais que θ_0 é o nó inicial do tableau, e, para $i=1, \ldots, k$, θ_i é um sucessor de θ_{i-1} , e θ_k é uma folha. Este ramo é dito que <u>termina</u> em θ_k. Claramente, para cada folha existe um único ramo terminando nela.

Uma fórmula pertencendo a um nó de um ramo é dita ser uma fórmula daquele ramo. As fórmulas pertencendo ao nó inicial são as <u>fórmulas iniciais</u> do tableau. Os nós de um tableau são parcialmente ordenados pela relação de <u>ancestralidade</u>, pela qual cada nó é <u>ancestral</u> de seus próprios sucessores, dos sucessores de seus sucessores, etc... Passaremos agora a um conjunto de definições formais, como em [Ca], que serão utilizadas no decorrer de todas as demonstrações que forem apresentadas neste trabalho.

<u>Definição 3.1</u>:

(a) O conjunto de tableaux analíticos para um conjunto P de fórmulas consiste de árvores cujos nós são conjuntos de fórmulas. Definiremos os tableaux analíticos de maneira indutiva como se segue:

(i) A árvore cujo único nó é P é um tableau analítico para P;

(ii) Suponha que T é um tableau para P e seja λ uma folha de T. Assim sendo, a árvore obtida estendendo T pela operação seguinte é também um tableau analítico para P: se existe uma regra R_i com antecedente P_i e consequentes $Q_{i1},\ldots,Q_{in_j}$ tal que todas as fórmulas em P_i ocorrem no ramo terminado em λ, então n_i distintos sucessores $\lambda_1,\ldots,\lambda_{n_i}$ podem simultaneamente ser ligados a λ, onde $\lambda_j \subseteq Q_{ij}$ $(1 \leq j \leq n_i)$

(b) Um conjunto H de fórmulas é um conjunto Hintikka com respeito a um conjunto U de constantes $\delta\delta\delta$.

(i) nenhuma fórmula e sua negação estão em H;

(ii) se existe uma regra R_i, distinta das regras $\neg$PR, DF e DI (introduzidas a seguir), com antecedente P_i e consequentes $Q_{i1},\ldots,Q_{in_j}$ tal que $P_i \neq \emptyset$ e $P_i \subseteq H$, então $Q_{ij} \subseteq H$, para algum $j \in [1,n_i]$;

(iii) se $(\neg e\,[X]\,(\bar{a})) \in H$, então, para qualquer tupla de constantes $\bar{b}$ em U tal que $\bar{b}_X$ é igual a $\bar{a}$, $\neg e(\bar{b}) \in H$.

(iv) se $(e : X \to Y) \in H$, então para quaisquer tuplas de constantes $\bar{a}$ e $\bar{b}$ tais que $\bar{a}_X = \bar{b}_X$, então teremos $Q \in H$ onde $Q \in \{\neg e(\bar{a}), \neg e(\bar{b}), \bar{a}_Y = \bar{b}_Y\}$

(v) se $(e \subset \delta) \in H$, então para qualquer tupla de constantes $\bar{a}$, teremos $Q \in H$ onde $Q \in \{\neg e(\bar{a}), \delta(\bar{a})\}$

(c) Um ramo de um tableau é fechado se e somente se contem uma fórmula e sua negação, caso contrário é um ramo aberto

(d) Um ramo de um tableau é completo se e somente se a união de todos os seus nós é um conjunto Hintikka (em relação ao conjunto de constantes da linguagem).

(e) Um tableau é fechado se e somente se todos os seus ramos são fechados.

(f) Um tableau T é completo se e somente se uma das seguintes situações ocorre:

(i) todos os ramos de T são fechados;

(ii) pelo menos um ramo de T é completo.

(g) Uma prova de uma fórmula P a partir de um conjunto de fórmulas $\mathcal{P}$ é um tableau fechado para $\mathcal{P} \cup \{\neg P\}$. Neste caso, P é um teorema de $\mathcal{P}$ em S (e escreve-se $\mathcal{P} \vdash P$). □

De modo a ilustrar as definições acima, damos um exemplo do Cálculo Proposicional, o qual na verdade é um subsistema de S. As regras que adotamos para o Cálculo Proposicional são as seguintes [Sm]:

$$\text{regras - A} \;.\; \frac{A}{A_1 \;,\; A_2} \qquad\qquad \text{regras - B} \;.\; \frac{B}{B_1 \mid B_2}$$

onde A, A_1, A_2 e B, B_1, B_2 são dados pelas seguintes tabelas:

A	A_1	A_2
$P \wedge Q$	P	Q
$\neg(P \vee Q)$	$\neg P$	$\neg Q$
$\neg(P \Rightarrow Q)$	P	$\neg Q$
$\neg\neg P$	P	P

Tabela 3.1

B	B_1	B_2
$\neg(P \wedge Q)$	$\neg P$	$\neg Q$
$P \vee Q$	P	Q
$P \Rightarrow Q$	$\neg P$	Q

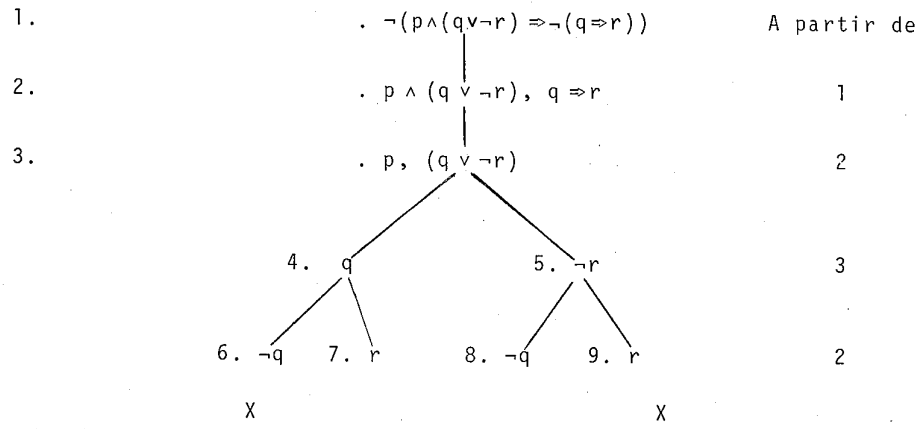
Tabela 3.2

As regras-A indicam que a partir de A, podemos deduzir A_1 e A_2; as regras-B indicam que a partir de B, podemos deduzir B_1 ou B_2. Por exemplo, assumindo $P \vee Q$, podemos ter 2 casos a considerar: P é verdadeira ou Q é verdadeira.

Exemplo 3.1: Seja P a seguinte fórmula do Cálculo Proposicional.

$$p \wedge (q \vee \neg r) \Rightarrow \neg(q \Rightarrow r)$$

Tentaremos provar que P é uma tautologia exibindo um tableau fechado começando com $\neg P$. Indicaremos um ramo fechado com um 'X', sempre que se chegar a uma contradição (alguma fórmula atômica e sua negação). Um ramo em que não apareça 'X' será um ramo aberto.
O tableau abaixo foi construído seguindo a definição 3.1(a) e usando as regras do Cálculo Proposicional. Desta forma, os nós 4 e 5, por exemplo, foram anexados como filhos do nó 3 usando a regra-B $\frac{P \vee Q}{P \mid Q}$ com P substituido por q e Q substituido por $\neg r$, ou seja, se temos $q \vee \neg r$, devemos considerar 2 casos: q ou $\neg r$.



Temos 2 ramos fechados e 2 ramos abertos, sendo que os ramos abertos estão completos, uma vez que cada um deles forma um conjunto Hintikka. Assim, o tableau está completo, embora não esteja fechado, o que nos mostra que já foram aplicadas todas as regras sem que tenhamos chegado a contradições em todos os casos possíveis. Portanto, o tableau referido é uma prova de que P não é uma tautologia.

Por outro lado, os ramos abertos indicam como construir contra-exemplos para P. Para isto será suficiente atribuirmos a q e r o valor verdadeiro ou falso a ambos simultaneamente, pois q e r ocorrem em um dos ramos abertos e ¬q e ¬r ocorrem no outro. Da exposição aqui apresentada, podemos concluir que o tableau considerado, na verdade, indica que P não é uma tautologia. Estas observações serão generalizadas na Seção 3.4 quando expusermos a prova da completude para *S*.

Passamos agora a descrever as regras de *S*. Por uma tupla de constante nova, queremos dizer uma tupla de constantes que não tenha ainda ocorrido no tableau que esteja sendo construido. Se $t = (t_1,\ldots,t_n, t_{n+1},\ldots,t_{n+m})$, então $t_{[1,n]}$ denota $(t_1,\ldots,t_n)$ e $t_{[n+1,n+m]}$ denota $(t_{n+1},\ldots,t_{n+m})$.

Regras DF:

¬DF. $\dfrac{\neg\, e: X \to Y}{e(\bar{a}), e(\bar{b}), \neg \bar{a}_Y = \bar{b}_Y}$ $\bar{a},\bar{b}$ são tuplas de novas constantes tais que $\bar{a}_X = \bar{b}_X$

DF. $\dfrac{e : X \to Y}{\neg e(\bar{a}) \mid \neg e(\bar{b}) \mid \bar{a}_Y = \bar{b}_Y}$

$\bar{a}$, $\bar{b}$ são quaisquer tuplas de constantes tais que $\bar{a}_X = \bar{b}_X$

Regras DI:

$\neg$DI. $\dfrac{\neg e \subset \delta}{e(\bar{a}),\ \neg\delta(\bar{a})}$

$\bar{a}$ é uma tupla de novas constantes

DI. $\dfrac{e \subset \delta}{\neg e(\bar{a}) \mid \delta(\bar{a})}$

$\bar{a}$ é qualquer tupla de constantes

Regras de Projeção:

$\neg$PR. $\dfrac{\neg e[X](\bar{a})}{\neg e(\bar{b})}$

$\bar{a}$,$\bar{b}$ são quaisquer tuplas de constantes com $\bar{b}_X = \bar{a}$

PR. $\dfrac{e[X](\bar{a})}{e(\bar{b})}$

$\bar{a}$ é qualquer tupla de constantes
$\bar{b}$ é uma tupla de novas constantes com $\bar{b}_X = \bar{a}$

Regras de Junção Natural:

$\neg$JN. $\dfrac{\neg(e * \delta)(\bar{c})}{\neg e(\bar{c}_X) \mid \neg\delta(\bar{c}_Y)}$

$\bar{c}$ é qualquer tupla de constantes e $X = ATR(e)$, $Y = ATR(\delta)$

JN. $\dfrac{(e * \delta)(\bar{c})}{e(\bar{c}_X),\ \delta(\bar{c}_Y)}$

Regras de União:

$\neg$UN. $\dfrac{\neg(e \cup \delta)(\bar{a})}{\neg e(\bar{a}),\neg\delta(\bar{a})}$

$\bar{a}$ é qualquer tupla de constantes

UN. $\dfrac{(e \cup \delta)(\bar{a})}{e(\bar{a}) \mid \delta(\bar{a})}$

Regras de Seleção:

$\neg$SE. $\dfrac{\neg e[X=d](\bar{a})}{\neg e(\bar{a}) \mid \neg \bar{a}_X = \bar{d}}$ $\bar{a}$ é qualquer tupla de constantes

SE. $\dfrac{e[X=\bar{d}]\ (\bar{a})}{e(\bar{a}), \bar{a}_X = \bar{d}}$ $\bar{a}$ é qualquer tupla de constantes

Regras de Igualdade:

RI. $\dfrac{}{\bar{a}=\bar{a}}$ SI. $\dfrac{\bar{a}=\bar{b}}{\bar{b}=\bar{a}}$ TI. $\dfrac{\bar{a}=\bar{b},\ \bar{b}=\bar{c}}{\bar{a}=\bar{c}}$

PI. $\dfrac{\bar{a} = \bar{b}}{\bar{a}_1=\bar{b}_1, \ldots\ldots, \bar{a}_n=\bar{b}_n}$ $\neg$PI. $\dfrac{\neg\, \bar{a} = \bar{b}}{\neg \bar{a}_1=\bar{b}_1 \mid \ldots\ldots \mid \neg \bar{a}_n=\bar{b}_n}$

II. $\dfrac{\bar{a} = \bar{b},\ e(\bar{a})}{e(\bar{b})}$ $\neg$II. $\dfrac{\bar{a} = \bar{b}, \neg\, e(\bar{a})}{\neg\, e(\bar{b})}$

$\bar{a}$, $\bar{b}$, $\bar{c}$ são tuplas n-árias de constantes, $n > 0$.

Regras-A e Regras-B:

Como no Cálculo Proposicional.

Considerando as regras de S, de maneira intuitiva, vemos que elas utilizam-se de padrões de raciocínio usados comumente em Matemática. A regra $\neg$DI, por exemplo, estabelece o seguinte: "... Assuma que $\neg e \subset \mathcal{G}$. Então deve haver uma tupla de constantes, ainda não utilizada, que pertence a e, mas que pertence a $\mathcal{G}$. Vamos denotar por $\bar{a}$ esta tupla ...". As regras FD, $\neg$FD e DI também podem ser explicadas de maneira análoga. Para o caso das regras que fazem uso de expressões relacionais, elas exprimem diretamente as definições anteriores. As regras de igualdade, as regras-A e as regras-B já são familiares, exceto as regras PI e $\neg$PI, nas quais nós interpretamos $\bar{a} = \bar{b}$ como $\bigwedge_{i=1}^{n} a_i = b_i$

Podemos aumentar S com regras derivadas, o que nos permitirá obter provas mais curtas.

Um conjunto de fórmulas possível é:

DF'. $\dfrac{e : X \twoheadrightarrow Y,\ e(\bar{a}),\ e(\bar{b}),\ \bar{a}_X = \bar{b}_X}{\bar{a}_Y = \bar{b}_Y}$ DI'. $\dfrac{e \subset \delta,\ e(\bar{a})}{\delta(\bar{a})}$

PR' $\dfrac{e(\bar{a})}{e[X]\ (\bar{a}_X)}$ JN'. $\dfrac{e(\bar{c}_X),\ \delta(\bar{c}_Y)}{(e * \delta)\ (\bar{c})}$

onde X = ATR(e)
Y = ATR(δ)
c̄ é qualquer tupla de constantes

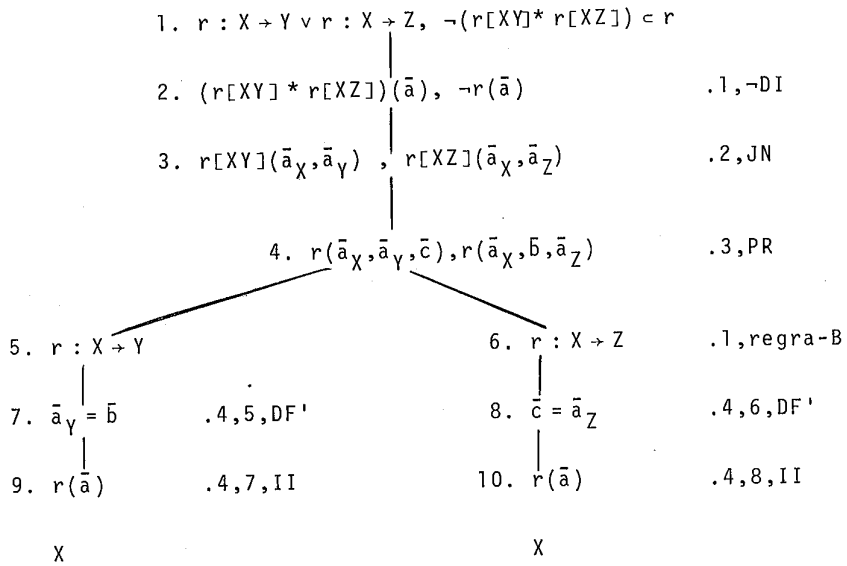
UN'. $\dfrac{(e \cup \delta)\ (\bar{a}),\ \neg e(\bar{a})}{\delta(\bar{a})}$ SE'. $\dfrac{e(\bar{a}),\ \bar{a}_X = \bar{d}}{e[X = \bar{d}]\ (\bar{a})}$

Estas regras podem ser derivadas diretamente a partir das regras ¬DF, ¬DI, ¬PR, ¬JN, ¬UN e ¬SE respectivamente. A fim de ilustrar o uso de *S*, apresentaremos alguns exemplos, os quais servirão para demonstrar a expressividade das linguagens DF-DI.

Exemplo 3.2: Apresentamos agora, uma prova formal em *S* da segunda parte do teorema 1 de [Ri]. Este resultado diz que, dada uma partição das colunas de um nome de relação r em X, Y, Z, se $r : X \twoheadrightarrow Y$ ou $r : X \twoheadrightarrow Z$ ocorre, então a junção natural de r [XY] e r [XZ] em X̄ é um subconjunto de r. Formalmente, o que queremos mostrar é que a seguinte fórmula é válida:

$$(r : X \twoheadrightarrow Y \vee r : X \twoheadrightarrow Z) \Rightarrow (r[XY] * r[XZ]) \subset r$$

Como veremos, na demonstração apresentada abaixo, obtivemos um tableau fechado e portanto terminamos a prova da segunda parte do teorema de [Ri]. □



4. CONSISTÊNCIA E COMPLETUDE DO SISTEMA S

Provamos nesta seção que S é consistente e completo. A consistência significa que $P \vdash P \Rightarrow P \models P$ ocorre e a completude significa que $P \models P \Rightarrow P \vdash P$. Uma vez que $P \models P \;\&\&\; \models P_1 \wedge \ldots \wedge P_n \Rightarrow P$ e $P \vdash P \;\&\&\; \vdash P_1 \wedge \ldots \wedge P_n \Rightarrow P$, onde $P = \{P_1,,,P_n\}$, nós podemos assumir sem perda de generalidade que P é vazio. Nós também assumimos que o conjunto de constantes da linguagem L usado por S é infinito (o que assegura que não esgotamos as constantes durante a prova).

Teorema 4.1: S é consistente.

Esboço da Prova: A prova da consistência de S é feita por indução na altura do tableau. Basta mostrar que se um ramo de um tableau é satisfatível (i.e, se o conjunto de fórmulas pertencendo ao ramo é satisfatível) e se esse ramo é estendido, (ou estendido e ramificado por qualquer das 26 regras de S), então o novo ramo (ou pelo menos um dos novos ramos no caso de haver ramificação) é também satisfatível. □

Para provar a completude de S nós temos que mostrar que se P é uma tautologia, então existe um tableau fechado para ¬P(i.e., que $\models P \Rightarrow \vdash P$). Nós realmente provamos que se P é uma tautologia, então

todo tableau completo para ¬P fecha. Ou, equivalentemente, se existe um tableau aberto e completo para ¬P, então ¬P é satisfatível e, portanto, P não é uma tautologia. Este resultado é obtido como se segue. Recorde que um tableau T é completo e aberto ⁂ algum ramo aberto β de T forma um conjunto Hintikka. Nós provamos que, de fato, qualquer conjunto Hintikka é satisfatível. Portanto, β é satisfatível e, uma vez que β começa com ¬P, então ¬P é satisfatível.

Lema 4.1: Qualquer conjunto Hintikka é satisfatível.

Prova: Seja H um conjunto Hintikka para L. Nós construimos uma estrutura I para L onde todas as fórmulas em H são verdadeiras. Nós primeiro definimos um conjunto E de classes de equivalência de constantes.

Seja U o conjunto de constantes de L e defina $\rho=\{(a,a)\mid a \in U\} \cup \{(a,b)\mid "a=b" \in H\}$. Por construção e uma vez que H é um conjunto Hintikka (usando as regras de igualdade), ρ é uma relação de equivalência. Nós tomamos E como o conjunto de classes de equivalência de ρ. A classe de equivalência de uma constante a é designada por a^0. I é construido como se segue. O domínio de I é E; para cada constante a, $I(a) = a^0$; para cada nome de relação n-ária r, $n > 0$, $I(r) = \{(a_1^0, \ldots, a_n^0) \in E^n \mid r(a_1,\ldots,a_n) \in H\}$. Considere agora I estendida a uma valorização booleana para as fórmulas de L. Nós mostramos que cada fórmula P em H é verdadeira em I por indução no grau de P (o número de ocorrências de $\rightarrow$, $\subset$, ¬, $\vee$, $\wedge$, $\Rightarrow$ e os operadores relacionais em P, sendo que $\rightarrow$ e $\subset$ tem peso 2).

Base: suponha que P tem grau 0.

Então P ou é $r(\bar{a})$ ou $\bar{a} = \bar{b}$, onde r é um nome de relação e $\bar{a}$, $\bar{b}$ são tuplas de constantes. Se P é $r(\bar{a})$ então, por construção de I, $\bar{a}^0 \in I(r)$. Portanto, P é verdadeira em I. Se P é $\bar{a} = \bar{b}$, então por construção de I, $a^0 = b^0$. Portanto P é verdadeira em I.

Passo de indução: Suponha que todas as fórmulas em de grau menor que i são verdadeiras em I e seja $P \in H$ uma fórmula de grau i.

Podemos resumir todos os casos a 3 esquemas.

Esquema do caso 1: P é $\neg e : X \rightarrow Y$, $e[X=\bar{d}](\bar{a})$, $\neg e \subset \delta$, $(e * \delta)(\bar{c})$, $e[X](\bar{a})$, $\neg(e \cup \delta)(\bar{a})$ ou o antecedente de uma regra-A. Então existe uma ocorrência de uma regra-B cujo antecedente é P e cujo consequente é $Q = \{Q_1,\ldots,Q_n\}$, onde cada Q_i tem grau menor que P. Uma vez que H é um conjunto Hintikka, cada Q_i está em H. Por hipótese

de indução, cada Q_i é verdadeiro em I. Assim, em cada caso específico, isto implica que P é verdadeiro em I.

Provamos em detalhe apenas o caso em que P é a DF negada $\neg e : X \twoheadrightarrow Y$. Uma vez que H é um conjunto Hintikka (usando a regra ¬DF) existem tuplas de constantes $\bar{a}$ e $\bar{b}$ tais que $e(\bar{a})$, $e(\bar{b})$, $\bar{a}_X = \bar{b}_X$ e $\neg\bar{a}_Y = \bar{b}_Y$ estão em H. Por hipótese de indução e uma vez que estas fórmulas tem grau menor que $\neg e : X \twoheadrightarrow Y$, elas são verdadeiras em I.

Portanto, por construção de I, $\bar{a}^0 \in I(e)$, $\bar{b}^0 \in I(e)$, $\bar{a}^0_X = \bar{b}^0_X$ e $\bar{a}^0_Y \neq \bar{b}^0_Y$. Portanto, $\neg e : X \twoheadrightarrow Y$ é verdadeira em I. Isto conclui o esquema de caso 1.

Esquema de caso 2: P é $\neg(e * f)(\bar{c})$, $(e \cup f)(\bar{a})$, $\neg e[X=\bar{d}](\bar{a})$ ou o antecedente de uma regra-B. A prova é análoga a do esquema de caso 1.

Esquema de caso 3: P é $\neg e: X \twoheadrightarrow Y$, $e \subset f$ ou $\neg e[X](\bar{a})$. A prova é análoga a do esquema de caso 1. □

De modo a usar o Lema 4.1 para obter a prova da completude de S, devemos garantir que todo ramo de um tableau que não feche torna-se um conjunto Hintikka. O procedimento dado na seção 3 permite construir tableaux com ramos abertos infinitos que não são conjuntos Hintikka . Isto ocorre porque:

(i) podem ser aplicados regras redundantemente para introduzir fórmulas já derivadas;

(ii) as regras ¬DF, DF, ¬DI, DI, ¬PR e PR podem ser aplicadas repetidamente para gerar fórmulas que diferem somente nas tuplas de constantes utilizadas;

(iii) a regra SI pode ser aplicada usando qualquer tupla de constantes.

Estes problemas são evitados refinando o procedimento para construir tableaux [Re].

O procedimento refinado para construir tableaux processa-se como na definição 3.1(a), exceto que:

(i) as regras nunca são aplicadas redundantemente;

(ii) são usadas tão poucas constantes quanto possível.

Existe uma outra importante característica do procedimento. Quando as regras DI, DF e ¬PR são aplicadas, as constantes são selecionadas a partir daquelas usadas no ramo que está sendo estendido, não a partir do conjunto de todas as constantes. Portanto, as condições de término garantem que, se o tableau não fecha, existe um ramo aberto θ que forma um conjunto Hintikka com respeito ao conjunto de todas as constantes ocorrendo em θ, mas não neces-

sariamente com respeito ao conjunto de todas as constantes.

Mas isto não afeta a prova do Lema 4.1 e abre a possibilidade de construir conjuntos Hintikka finitos.

Por um tableau sistemático terminado, queremos dizer um tableau construido pelo procedimento refinado, o qual é finito ou infinito, mas não pode ser mais estendido pelo procedimento refinado.

Agora, chegamos finalmente ao teorema da completude para S.

Teorema 4.2:

(a) Todo tableau sistemático terminado e aberto tem um ramo que é um conjunto Hintikka.

(b) Se uma fórmula P é uma tautologia, então todo tableau sistemático terminado que comece com $\neg P$ deve fechar.

(c) O sistema S é completo.

Prova

(a) Segue pela definição do procedimento refinado para construir tableaux.

(b) Suponha que exista um tableau sistemático terminado que comece com $\neg P$ que não é fechado. Então, por (a) ele contém um ramo aberto β que forma um conjunto Hintikka H. Pelo Lema 4.1, H é satisfatível. Uma vez que, $\neg P \in H, \neg P$ é também satisfatível. Portanto, P não é uma tautologia.

(c) Suponha que P é uma tautologia. Por (b), existe um tableau fechado para $\neg P$. Portanto, $\models P \Rightarrow \vdash P$. □

5. CONCLUSÕES

Apresentamos brevemente um sistema formal para dependências funcionais e de inclusão, tomadas sobre expressões relacionais, e não apenas sobre relações como usual [Ar, BBG]. Este sistema permite abordar certos problemas de projeto de banco de dados de forma natural, como explorado em [RC,Re].

A consistência e completude do sistema foi obtida através do método dos tableaux analíticos. Este método torna-se bastante interessante neste caso pois explora a sintaxe das fórmulas para obter provas mais concisas do que aquelas que seriam obtidas, por exemplo, usando uma formulação de primeira ordem para o conjunto de dependências em questão.

Além disto, com certas modificações, o método dos tableaux analíticos pode ser usado para mostrar a decidibilidade do problema da satisfatibilidade para certos conjuntos de fórmulas. Este últi-

mo ponto e a aplicação do sistema ao problema de normalização [BBG] são questões que merecem investigações futuras.

6. BIBLIOGRAFIA

[Ar] Armstrong,W.W. "Dependency Structures of Data Base Relationships", Proc.IFIP'74, North Holland (1974).

[BBG] Beeri,C., Bernstein,P.A., Goodman,N. "A Sophisticate's Introduction to Data Base Normalization Theory". Proc. 5th Int. Conf. on Very Large Databases (1978).

[BG] Bernstein,P.A., Goodman,N. "What Does Boyce-Codd Normal Form Do?" Proc. 6th Int.Conf.on Very Large Data Bases (1980).

[Ca] Casanova,M.A. "A Theory of Data Dependencies over Relational Expressions". PUC-Technical Report DB 108104 (Oct. 1981).

[RC] Rego,S.P., Casanova,M.A. "Aplicações de uma Teoria de Dependências Funcionais e Dependências de Inclusão". Proc. do 15º Congresso Nacional de Informática (Out. 1982).

[Re] Rego,S.P. "Uma Teoria de Dependências Funcionais e Dependências de Inclusão sobre Expressões Relacionais". Dissertação de Mestrado, PUC/RJ (1982).

[Ri] Rissanen,J. "Independent Components of Relations". ACM TODS 2,2 (Dec. 1977).

[Sm] Smullyan,R.M. "First-Order Logic". Springer-Verlag (1971).