

Mètodes de validació d'esquemes
de bases de dades deductives

Carles Farré

Report LSI-96-51-R

Mètodes de validació d'esquemes de bases de dades deductives

Carles Farré

Universitat Politècnica de Catalunya

Dept. de Llenguatges i Sistemes Informàtics

Pau Gargallo 5. E-08028 BARCELONA

e-mail: farre@goliat.upc.es

RESUM: La validació d'esquemes està esdevenint un dels principals problemes de l'enginyeria de bases de dades. Per validació entenem el procés de comprovar si un esquema de base de dades descriu correctament i adequada les necessitats i requeriments dels usuaris. En aquest article presentem i analitzem sis mètodes diferents que tracten la validació d'esquemes en el camp de les bases de dades deductives. Tots aquests mètodes defineixen i comproven algunes propietats desitjables/indesitjables que un esquema de base de dades hauria/no hauria de satisfer: satisfactibilitat de l'esquema, factibilitat d'un predicat, assolibilitat d'una fórmula o redundància de les especificacions de restriccions d'integritat.

ABSTRACT: *Schema validation is becoming one of the most important problems in database engineering. By validation we mean the process of checking whether a database schema correctly and adequately describes the users' intended needs and requirements. In this paper we present and discuss six different methods handling schema validation in the deductive database field. All these methods define and check some desirable/undesirable properties that a database schema should/should not satisfy: schema satisfiability, liveness of a predicate, reachability of a formula and redundancy of integrity constraint specifications.*

1. INTRODUCCIÓ	4
2. NOTACIÓ I MARC CONCEPTUAL	6
2.1 Lògica de primer ordre.....	6
2.2 Bases de Dades Deductives (BDD).....	8
2.3 Propietats de l'esquema de base de dades deductiva.....	9
2.3.1 Satisfactibilitat (<i>State-satisfiability</i>).....	9
2.3.2 Factibilitat d'un predicat (<i>Liveliness</i>).....	11
2.3.3 Assolibilitat d'una fórmula (<i>Reachability</i>).....	11
2.3.4 Redundància d'una restricció d'integritat.....	11
3. MÈTODES DE VALIDACIÓ	13
3.1 El mètode de comprovació de la satisfactibilitat de Kung [Kun84, Kun85].....	13
3.1.1 Plantejament.....	13
3.1.2 Delimitació de l'àmbit d'aplicació del mètode.....	14
3.1.3 Funcionament del mètode (<i>The Tableaux Approach</i>).....	14
3.1.4 Discussió del mètode.....	16
3.2 El mètode de comprovació de la satisfactibilitat de Bry et al. [BDM88].....	17
3.2.1 Plantejament.....	17
3.2.2 Delimitació de l'àmbit d'aplicació del mètode.....	17
3.2.3 Funcionament del mètode.....	18
3.2.4 Discussió del mètode.....	20
3.3 El mètode de comprovació de la satisfactibilitat i la factibilitat de Marqués i Casamayor [MC93].....	21
3.3.1 Plantejament.....	21
3.3.2 Delimitació de l'àmbit d'aplicació del mètode.....	22
3.3.3 Funcionament del mètode.....	22
3.3.4 Discussió del mètode.....	25
3.4 Factibilitat i P-assolibilitat de predicats en Levy et al. [LMSS93].....	27

3.4.1 Plantejament.....	27
3.4.2 Delimitació de l'àmbit d'aplicació del mètode	28
3.4.3 Funcionament del mètode	28
3.4.4 Discussió del mètode	31
3.5 El mètode de comprovació de la redundància relativa de restriccions d'integritat de Gupta et al. [GSUW94].....	32
3.5.1 Plantejament.....	32
3.5.2 Delimitació de l'àmbit d'aplicació del mètode	33
3.5.3 Funcionament del mètode	33
3.5.4 Discussió del mètode	34
3.6 Validació d'esquemes de Bases de Dades Deductives en Decker et al. [DTU96].....	35
3.6.1 Plantejament.....	35
3.6.2 Delimitació de l'àmbit d'aplicació del mètode	36
3.6.3 Funcionament del mètode	36
3.6.4 Discussió del mètode	39
4. CONCLUSIONS	40
Referències.....	41

1. Introducció

En aquest treball pretenem fer una anàlisi d'alguns dels mètodes proposats per validar bases de dades deductives durant els darrers anys. El que ens interessa saber és quins problemes volen afrontar, com els formalitzen, quines eines utilitzen, amb quines dificultats es troben i com se'n surten.

Però, què entenem per validació?. Heus aquí dos botons:

- Validació és el procés de comprovar si un esquema de base de dades descriu correctament i adequada les necessitats i requeriments dels usuaris [ABC82].
- Validació és una tasca informal i basada en la intuïció per la qual el dissenyador experimenta amb l'esquema de la base de dades, estudia els efectes d'algunes accions i compara llurs resultats amb les seves expectatives [LMN+93]

O sigui, per un costat tenim requeriments, necessitats i expectatives, i per l'altre tenim l'esquema de la base de dades. Es tracta, doncs, de confrontar el coneixement informal del que és o ha de ser el sistema i un esquema de base de dades especificat en algun llenguatge formal com, per exemple, la lògica de primer ordre. Com que no hi ha cap procediment automàtic que converteixi una especificació informal de l'usuari en un esquema amb tots els ets i uts, encara menys hi ha eines automàtiques de validació d'esquemes. Com diu la segona definició, el dissenyador/validador s'ha de guiar per la seva intuïció.

Quan hom no disposa d'altra cosa que la intuïció, el que s'acostuma a fer és el prototipatge: a partir de l'esquema (provisional) es va "farcint" la base de dades amb tot d'informació més o menys realista sobre el domini d'aplicació. Ja en aquesta fase podrien aparèixer problemes, com ara que la bases de dades no admet cap dada, simplement perquè hi ha definida una restricció d'integritat que es viola contínuament. Un cop construït el prototipus, es comença a simular el comportament "normal" de la base de dades, per després comprovar la seva robustesa davant dels casos més excepcionals.

Els mètodes que veurem en aquest treball comproven que l'esquema de base de dades deductiva compleixi unes determinades propietats. Podem entendre aquestes propietats com unes mesures de la "bondat" de l'esquema, quelcom més tangible que la intuïció pura. Aquests mètodes no fan evidentment la validació per si sols, són simplement unes eines de suport que en un moment donat poden ajudar al dissenyador a prendre decisions sobre l'esquema. Aquests mètodes pràcticament només són útils als dissenyadors que han fet l'esquema o, dit d'una altra manera, són els que en

poden treure més profit, perquè són ells els que han interpretat els requeriments dels usuaris i aplicacions, saben quines decisions de disseny s'han pres i quin esquema s'ha implementat.

Sovint els autors no es posen d'acord en diferenciar el que és validació del que és verificació. Com a regla general direm que validació és tot allò que confronta un model (l'esquema) amb un altre (els requeriments), mentre que la verificació comprova la correctesa del model en ell mateix. Però això tampoc serveix de gaire i algunes de les propietats que veurem cauen d'un costat o de l'altre segons l'autor. Per exemple, [TAL92] distingeix dos tipus de verificació, la sintàctica i la semàntica. La primera comprova que el model construït hagi utilitzat correctament les construccions, la "gramàtica", del llenguatge d'especificació usat. En canvi, la segona comprova que no hi hagin contradiccions o redundàncies en el model "...i està molt a prop del procés de validació" [sic]. Sota aquest enfocament, les propietats d'esquema de base de dades deductiva que veurem són verificació semàntica.

No ens hem proposat, doncs, traçar la línia divisòria entre validació i verificació, simplement hem considerat aquelles propietats d'esquema que han estat objecte d'estudi en la literatura sobre el tema i que hom ha cregut que serien útils per fer intentar comprovar a priori si la base de dades dissenyada compleix algun dels propòsits per la que fou concebuda.

Encara que ens hem centrat exclusivament en el context de les bases de dades deductives, la validació òbviament no és exclusiva d'aquest camp. Intel·ligència artificial, enginyeria del software, sistemes d'informació, etc, han desenvolupat les seves pròpies línies de recerca, sovint convergents, sobre la validació. Concretament, del camp de la modelització conceptual de sistemes d'informació prové el primer mètode que hem inclòs en aquest treball, [Kun84, Kun85], ja que el seu plantejament és clarament aplicable al món les bases de deductives i ha estat àmpliament referenciat pels autors posteriors. Darrerament també han aparegut en aquest camp dos treballs, [FW95] i [CTUF96], on les propietats de validació que es tenen en consideració són pràcticament les mateixes que veurem. Ambdós mètodes funcionen de manera semblant: defineixen les propietats de validació de manera declarativa en termes d'un objectiu d'un problema de planificació. Aquest enfocament és similar al darrer dels mètodes que analitzarem, [DTU96].

En la secció següent introduïrem la notació i el marc conceptual de les bases de dades deductives, així com les propietats de validació que han estat objecte d'estudi dels diferents mètodes. A continuació analitzarem sis mètodes apareguts des de 1984 fins avui en dia, tots dins del camp de les bases de dades deductives menys, com hem dit abans, el primer. Finalment, les conclusions.

2. Notació i marc conceptual

La notació i la majoria de les definicions s'han pres o s'han adaptat de [Das92], pel que fa a la formalització de la lògica de primer ordre, i de [DTU96], per tot el que fa referència a la formalització de les bases de dades deductives i de les propietats de validació d'un esquema de bases de dades deductives.

2.1 Lògica de primer ordre

Constants: $a, b, c, a_1, b_1, c_1, \dots$

Variables: $x, y, z, x_1, y_1, z_1, \dots$

Termes:

1. Una variable o una constant és un terme.
2. Si f és un símbol de funció n -ari i t_1, \dots, t_n són termes llavors $f(t_1, \dots, t_n)$ és un terme.
3. Qualsevol expressió és un terme si es pot veure com una aplicació de les dues condicions anteriors

Si P és un símbol de predicat n -ari i t_1, \dots, t_n són termes llavors $P(t_1, \dots, t_n)$ és una fórmula atòmica o àtom o literal positiu. Un literal negatiu és una fórmula de la forma $\neg A$, on A és un àtom. Un àtom $P(t_1, \dots, t_n)$ està instanciat si t_1, \dots, t_n són constants.

Fórmules de la lògica de primer ordre:

1. Un àtom és una fórmula
2. Si F és una fórmula llavors $\neg F$ és una fórmula
3. Si F és una fórmula i x és una variable llavors $\forall x F$ i $\exists x F$ són fórmules.
4. Si F i G són fórmules llavors $F \rightarrow G, F \leftarrow G, F \wedge G, F \vee G$ són fórmules.
5. Qualsevol expressió generada a partir de les quatre condicions anteriors és una fórmula.

Una ocurrència d'una variable x en una fórmula F es diu que està lligada si $\forall x$ o $\exists x$ ocorren en F o x està dins de l'àmbit d'influència de $\forall x$ o $\exists x$ en F . En cas contrari, hom diu que l'ocurrència de x és lliure en F . Una fórmula tancada és una fórmula sense cap variable lliure.

Una clàusula és una fórmula de la forma:

$$A_1 \vee \dots \vee A_m \leftarrow L_1 \wedge \dots \wedge L_n, \quad m \geq 1, n \geq 0$$

on $A_1 \vee \dots \vee A_m$ és la capçalera de la clàusula i $L_1 \wedge \dots \wedge L_n$ és el cos. Cada A_i és un àtom i cada L_j és un literal. Tota variable que ocorre en $A_1, \dots, A_m, L_1, \dots, L_n$ s'assumeix que està quantificada universalment en tota la fórmula.

Un fet és una clàusula de la forma:

$$A \leftarrow, \quad \text{on } A \text{ és un àtom.}$$

Una clàusula definida és una clàusula de la forma:

$$A \leftarrow B_1 \wedge \dots \wedge B_n, \quad n \geq 0, \text{ on } A \text{ i } B_i \text{ són àtoms.}$$

Una clàusula normal és una clàusula de la forma:

$$A \leftarrow L_1 \wedge \dots \wedge L_n, \quad n \geq 0, \text{ on } A \text{ és un àtom i } L_i \text{ és un literal.}$$

Una clàusula disjuntiva és una clàusula de la forma:

$$A_1 \vee \dots \vee A_m \leftarrow B_1 \wedge \dots \wedge B_n, \quad m \geq 1, n \geq 0, \text{ on } A_i \text{ i } B_j \text{ són àtoms}$$

Un objectiu normal o denegació és una fórmula de la forma:

$$\leftarrow L_1 \wedge \dots \wedge L_n, \quad n \geq 1$$

on $L_1 \wedge \dots \wedge L_n$ és el cos de la denegació i cada L_i és un literal. Tota variable que ocorre en L_1, \dots, L_n s'assumeix que està quantificada universalment en tota la denegació.

2.2 Bases de Dades Deductives (BDD)

Un esquema de base de dades deductiva **S** és una tupla (**IDB,IC**) on

- **IDB** (Base de Dades Intensional) és un conjunt finit de clàusules. Els predicats que apareixen en els àtoms de la capçalera de les clàusules són els predicats derivats o IDB. Els predicats que apareixen en els cossos de les clàusules però no en les capçaleres són els predicats bàsics o EDB.
- **IC** és un conjunt finit de fórmules tancades anomenades restriccions d'integritat.

Per regla general considerarem que les clàusules de **IDB** són (regles deductives) de la forma:

$$A_1 \vee \dots \vee A_m \leftarrow L_1 \wedge \dots \wedge L_n \quad m \geq 1, n \geq 1$$

de manera que no permetrem que hi hagin fets sobre predicats derivats en IDB

Altres propietats que sovint s'exigeix que ha de complir **IDB**:

- Sense funcions: no es permet l'ocurrència de símbols de funcions en les clàusules
- Permissible: qualsevol variable que aparegui en una clàusula ha d'ocórrer en un literal ordinari positiu del cos de la clàusula.
- Estratificat: No poden haver negacions sobre predicats que siguin directament o indirecta recursius.

Datalog és un tipus de IDB constituït per un conjunt finit de clàusules disjuntives (sense negació) i sense funcions.

Habitualment les restriccions d'integritat es representen en forma de denegació. Qualsevol fórmula tancada F pot representar-se en forma d'una o més denegacions sense pèrdua de generalitat: s'expressa com $\leftarrow \neg F$ i llavors es transforma en un conjunt equivalent de clàusules que s'inclou dins **IC**, tal com descriu [Llo87].

Un estat de base de dades deductiva **D** és un tripleta (**IDB,IC,EDB**) on **EDB** és un conjunt finit de fets bàsics instanciats, és a dir, àtoms instanciats de predicats bàsics. **D** també rep els noms estat, estat de **S** o base de dades deductiva.

Intuitivament, el contingut de $\mathbf{D} = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB})$, denotat $\mathbf{IDB}(\mathbf{EDB})$, és el conjunt dels fets bàsics de \mathbf{EDB} més les extensions calculades dels predicats derivats aplicant les regles deductives.

Intuitivament, una fórmula F és certa en $\mathbf{D} = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB})$ si és pot deduir a partir del contingut de \mathbf{D} : $\mathbf{IDB}(\mathbf{EDB}) \mapsto F$.

Un estat de base de dades deductiva $\mathbf{D} = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB})$ és consistent o vàlid si totes les restriccions d'integritat de \mathbf{IC} són certes, satisfetes, en \mathbf{D} . Altrament, si una restricció d'integritat $I \in \mathbf{IC}$ no és satisfeta en \mathbf{D} direm que I es violada i que \mathbf{D} és inconsistent.

Les transicions entre estats de la BDD es realitzen mitjançant actualitzacions: una actualització \mathbf{U} d'un estat $\mathbf{D}_i = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB}_i)$ és un conjunt bipartit $\mathbf{U} = \mathbf{Ins} \cup \mathbf{Del}$ de fets bàsics de manera que $\mathbf{U}(\mathbf{D}_i) = \mathbf{D}_{i+1} = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB}_{i+1})$, on $\mathbf{EDB}_{i+1} = (\mathbf{EDB}_i - \mathbf{Del}) \cup \mathbf{Ins}$.

2.3 Propietats de l'esquema de base de dades deductiva

2.3.1 Satisfactibilitat (*State-satisfiability*)

Un esquema de base de dades deductiva $\mathbf{S} = (\mathbf{IDB}, \mathbf{IC})$ és satisfactible si existeix un estat $\mathbf{D} = (\mathbf{IDB}, \mathbf{IC}, \mathbf{EDB})$ consistent.

Cal remarcar la referència explícita a la necessitat de que hi ha d'haver estats consistents per considerar l'esquema satisfactible. Aquesta visió de la satisfactibilitat difereix de la visió "clàssica" de la lògica, que consideraria que \mathbf{S} seria satisfactible si els conjunt $\mathbf{IDB} \cup \mathbf{IC}$ de fórmules tingués un model. Intuitivament, un model d'un conjunt de fórmules en la lògica de primer ordre és un conjunt d'àtoms instanciats que fan que les fórmules siguin certes. Així, per exemple, la clàusula $P(x) \leftarrow \neg P(x)$ té el model $\mathbf{M} = \{P(a)\}$. En canvi, una fórmula del tipus $P(x) \wedge \neg P(x)$ no té cap model. Aplicant aquests conceptes als esquemes de BDD, ens trobaríem amb models que contenen fets sobre predicats derivats. Aquests fets "derivats" podrien no ser deduïbles a partir dels fets bàsics del model i, per tant, ens trobaríem amb la paradoxa de tenir informació derivada sense "fonament".

Exemple 2.3.1:

$$\mathbf{IDB} = \{ \text{Boss}(x) \leftarrow \text{Works_for}(x,x) \wedge \neg \text{Subordinate}(x) \\ \text{Subordinate}(x) \leftarrow \text{Works_for}(x,y) \}$$

$$\mathbf{IC} = \{ \leftarrow \text{Top_manager}(x) \wedge \neg \text{Boss}(x) \\ \leftarrow \neg \text{Top_manager}(\text{josemaria}) \}$$

A simple vista hom veu que no pot haver cap estat consistent amb aquest esquema (al cap i a la fi, tothom que treballa per un mateix és també subordinat d'un mateix).

En canvi, $\mathbf{M} = \{ \text{Top_manager}(\text{josemaria}), \text{Boss}(\text{josemaria}) \}$ és un model de $\mathbf{IDB} \cup \mathbf{IC}$.

Com a conseqüència de tot això tenim que qualsevol estat consistent és un model de $\mathbf{IDB} \cup \mathbf{IC}$, però no tots els models de $\mathbf{IDB} \cup \mathbf{IC}$ representen estats consistents de la BDD. Només en el cas de que $\mathbf{IDB} = \emptyset$, els dos enfocaments produirien resultats equivalents.

En [BM86] els autors constaten que hi casos en que els esquemes són (model-)satisfactibles, però llurs models són infinits. Això es degut a la presència d'axiomes d'infinitud en les restriccions d'integritat,

$$\forall x \exists y \text{ Works_for}(x,y)$$

$$\forall x \neg \text{Works_for}(x,x)$$

$$\forall x \forall y \forall z \text{ Works_for}(x,y) \wedge \text{Works_for}(y,z) \rightarrow \text{Works_for}(x,z)$$

és un exemple típic. Donat que en realitat, en el món de les BDD, els estats només contenen informació finita, els autors proposen que els esquemes no sols han de ser "només" satisfactibles sinó que cal que siguin també finitament satisfactibles, és a dir, amb models finits. La satisfactibilitat finita, però, és una propietat semidecidible, en el sentit de que un algorisme que comprovi aquesta propietat en un esquema acabarà i retornarà cert si l'esquema és finitament satisfactible, però en cas contrari pot no tenir una terminació finita.

La satisfactibilitat tal com l'hem definida al principi és "finita", ja que està en funció dels possibles estats consistents i tot estat consistent és finit. Amb tot, continua essent una propietat semidecidible.

2.3.2 Factibilitat d'un predicat (*Liveliness*)

Un esquema pot ser satisfactible, però només en l'estat buit, és a dir, en aquell estat on no hi ha cap fet bàsic. En aquest cas, la satisfactibilitat no aportaria gran cosa. En canvi, hom pot estar interessat en saber si determinats predicats derivats són "utilitzats" algun cop, és a dir, si hi ha algun estat de la BDD on hi ha fets d'aquests predicats.

Un predicat n -ari P és factible en $S = (\text{IDB}, \text{IC})$ si existeix un estat consistent $D = (\text{IDB}, \text{IC}, \text{EDB})$ en el que hi ha almenys un fet sobre P : $P(a_1, \dots, a_n) \in \text{IDB}(\text{EDB})$, on a_1, \dots, a_n són constants.

2.3.3 Assolibilitat d'una fórmula (*Reachability*)

Es pot entendre la assolibilitat com una extensió de l'anterior propietat. Al dissenyador no només li pot interessar que certs predicats siguin factibles, sinó que ho siguin també combinacions de dos o més predicats alhora. En general, el dissenyador podria voler comprovar si determinades condicions més o menys complexes sobre els predicats de l'esquema es compleixen en algun estat consistent de la BDD. Llavors, el que caldria veure és si aquestes condicions expressades com fórmules són assolibles en algun estat.

Una fórmula F és assolible en $S = (\text{IDB}, \text{IC})$ si existeix un estat consistent $D = (\text{IDB}, \text{IC}, \text{EDB})$ on F és certa: $\text{IDB}(\text{EDB}) \models F$.

2.3.4 Redundància d'una restricció d'integritat

Les restriccions d'integritat són propietats que hom vol que es satisfacin en cada estat de la BDD. Això implica que cada cop que es fa una actualització, un canvi d'estat, s'han de comprovar de nou. Encara que hagin tècniques optimitzades que només tenen en compte aquelles restriccions d'integritat susceptibles de ser "afectades" per l'actualització, els tests de consistència són un factor crític en tot sistema de gestió de base de dades (SGBD). Per tant, el dissenyador ha de tenir cura a l'hora de definir quines restriccions d'integritat vol incorporar a l'esquema. Amb les propietats que hem vist fins ara el dissenyador pot adonar-se de si alguna restricció entra en conflicte amb la resta de l'esquema, provocant que aquest sigui insatisfactible o que algun predicat no sigui mai factible. Ara, del que es tracta és de saber si hi ha restriccions d'integritat supèrflues, per decidir després de quines es pot prescindir, " Descarregant" així al SGBD de comprovacions inútils.

Una restricció d'integritat $I \in \mathbf{IC}$ és absolutament redundat si és certa en cada estat de $\mathbf{S} = (\mathbf{IDB}, \mathbf{IC})$

Una restricció d'integritat $I \in \mathbf{IC}$ és relativament redundat si és certa en cada estat consistent de $\mathbf{S}' = (\mathbf{IDB}, \mathbf{IC} - \{I\})$

Creiem que es convenient diferenciar els dos tipus de redundància. En el cas de l'absoluta, clarament aquestes restriccions es poden eliminar de l'esquema sense por de perdre consistència. En canvi, això no està tant clar en el cas de la redundància relativa perquè poden haver-hi subconjunts de \mathbf{IC} mútuament redundants i la decisió de quin descartar a vegades no és trivial.

Hom pot dubtar que el test de redundància formi part de la validació de l'esquema, ja que el fet de que hi hagi redundàncies en les restriccions d'integritat no "allunya" l'esquema dels requeriments i necessitats dels usuaris, ni provoca errors de consistència quan la BDD ja està funcionant. De fet, la comprovació i eliminació de redundàncies ho podríem entendre com una optimització de l'esquema, ja que eliminant les restriccions redundants es milloraria la *performance* de la BDD. Sigui com sigui, les redundàncies introdueixen prou "soroll" en l'esquema com per interferir i complicar el procés de validació i, per tant, no està de més disposar de tècniques que permetin identificar-les.

3. Mètodes de Validació

3.1 El mètode de comprovació de la satisfactibilitat de Kung [Kun84, Kun85]

3.1.1 Plantejament

L'autor proposa un "marc temporal per a l'especificació i la verificació de models conceptuals de sistemes d'informació". De fet, més que una metodologia, Kung introdueix un llenguatge d'especificació formal amb les eines necessàries per "verificar" el models que s'especifiquen amb aquest llenguatge.

Un model conceptual consta de tres components:

1. Un conjunt de restriccions "estàtiques" (*static constraints*), que és el conjunt de restriccions d'integritat que han de satisfer els estats "legals" del model.
2. La descripció de les operacions del model, on per cada operació s'ha d'especificar:
 - la precondició: la condició que han de satisfer els estats en els que s'executarà l'operació
 - la postcondició: la condició que s'ha de complir en el nou estat resultant de l'execució de l'operació
 - una "asserció temporal", que expressa una condició que han de satisfer tots els estats del model anteriors a l'execució de l'operació
3. Un conjunt de restriccions temporals, que expressa les condicions que tota seqüència "permissible" d'operacions ha de satisfer.

Dins de l'àmbit de discussió del nostre treball, només ens ocuparem del mètode que utilitza Kung per "verificar" les restriccions "estàtiques", i que ha estat referenciat per la majoria dels autors posteriors com un dels primers intents d'establir un mètode per determinar la satisfactibilitat d'un conjunt de restriccions d'integritat amb els mitjans de la lògica de primer ordre.

L'objectiu del mètode de "verificació" de les restriccions estàtiques és doble: comprovar la satisfactibilitat del conjunt de restriccions i alhora crear un cert nombre d'estats "prototips" que satisfacin les restriccions estàtiques i que serveixin per una posterior validació de les operacions definides en el model.

3.1.2 Delimitació de l'àmbit d'aplicació del mètode

$S = (\emptyset, IC)$: el contingut estructural de la base d'informació dels models conceptuals especificats sota aquest paradigma s'expressa en les restriccions estàtiques. No hi ha predicats derivats i els predicats bàsics són els que apareixen en les restriccions estàtiques..

Les restriccions estàtiques són fórmules tancades sense símbols de funcions. Per exemple, les restriccions “tot empleat guanya més de 20.000\$” i “tot manager és un empleat”, s'escriurien:

$$sc_1: (\forall x)(\forall y)(E(x, y) \rightarrow y > 20.000)$$

$$sc_2: (\forall x)(\exists y)(M(x) \rightarrow E(x, y))$$

Tanmateix, l'autor ja s'adona que hi ha casos intractables en el que el seu mètode no és decidible, és a dir, no troba una solució en temps finit. El problema principal és la introducció de noves constants. Suposem que tenim una restricció que ens diu “tota persona x treballa per a una persona y ”: $(\forall x)(\exists y) Works_for(x, y)$. Per trobar un estat que satisfés aquesta restricció, el mètode de Kung introduiria dues constants a_0 i a_1 i obtindria un estat “provisional” $\{Works_for(a_0, a_1)\}$. Però tampoc no es compliria la restricció perquè a_1 no treballa per ningú. Llavors s'introduiria una nova variable a_2 , però tornariem a estar igual que abans. La solució que proposa Kung en aquest cas no és limitar el tamany dels dominis (dominis finits), sinó reintroduir les constants que ja existeixen. En aquests casos, doncs, l'autor recomana substituir el quantificador existencial, especialment quan està darrera d'un d'universal, per un nou quantificador, $\exists!$, que reintrodueix les constants preexistents. Per tant, la nova restricció estàtica seria $(\forall x)(\exists! y) Works_for(x, y)$ i $\{Works_for(a_0, a_0)\}$ un estat “legal”.

3.1.3 Funcionament del mètode (*The Tableaux Approach*)

El mètode funciona amb la presumpció de que les restriccions estàtiques no tenen $(\forall x) \dots (\exists y)$ o $(\exists! x) \dots (\exists y)$, que obliguin a la introducció recursiva de noves constants .

El mètode es basa en el procediment de demostració de teoremes mitjançant “tableaux” (taulers) analítics. A partir del conjunt inicial de fórmules (en aquest cas, les restriccions estàtiques) es van generant noves fórmules aplicant regles de transformació sintàctica, tot construint un arbre de derivació. Si en totes les branques generades apareix una fórmula que contradiu alguna de les anteriors, llavors el conjunt inicial de fórmules és insatisfactible. Altrament, si després d'haver esgotat totes les transformacions possibles encara resta alguna branca “oberta”, el conjunt de fórmules inicial és satisfactible

3.2 El mètode de comprovació de la satisfactibilitat de Bry et al. [BDM88]

3.2.1 Plantejament

Donat un esquema de BDD $S = (IDB, IC)$, els autors proposen un mètode per comprovar la satisfactibilitat finita tot construint un conjunt finit de fets, un prototip de BDD, que satisfaci totes les restriccions d'integritat.

Com hem dit, el problema de comprovar la satisfactibilitat finita és semidecidible i, per tant, el mètode que presenten pot no acabar mai si IC conté "axiomes de infinitud".

3.2.2 Delimitació de l'àmbit d'aplicació del mètode

IDB és un conjunt finit estratificable de clàusules normals permissibles.

IC és un conjunt finit de fórmules tancades, sense símbols de funcions i amb quantificació restringida, és a dir, les (sub)fórmules quantificades només poden ser d'una d'aquestes dues formes:

$$\begin{aligned} \exists x_1 \dots \exists x_n [A_1 \wedge L \wedge A_m \wedge Q] \\ \forall x_1 \dots \forall x_n [\neg A_1 \vee L \vee \neg A_m \vee Q] \end{aligned}$$

on A_1, \dots, A_m són àtoms, cada variable x_i ocorre almenys en un A_j , i Q és cert o fals o alguna subfòrmula on totes les x_i que hi apareixen són lliures.

Les restriccions d'integritat amb quantificació restringida són independents del domini. Això implica que no es puguin expressar restriccions d'integritat del tipus "tothom treballa per algú", $\forall x \exists y \text{ Works_for}(x, y)$, que afecten a tot el domini de la BDD. En canvi, la restricció "tothom que treballa per a un mateix té algú altri treballant per ell" que ja havíem vist en l'exemple 3.1.2, sí que és independent del domini:

$$\forall x [\neg \text{Works_for}(x, x) \vee \exists y (\text{Works_for}(y, x) \wedge y \neq x)].$$

3.2.3 Funcionament del mètode

Com ja hem dit, el mètode intenta construir un conjunt finit de fets instanciats, un prototip de BDD, per tal demostrar la satisfactibilitat de S . Per construir el prototip, els autors combinen dos tècniques:

1. la “reparació” de les restriccions d’integritat violades mitjançant la inserció de nous fets en el prototip en construcció i
2. la detecció de quines restriccions d’integritat són violades quan s’insereix un nou fet en el prototip

Inicialment, el prototip de BDD no conté cap fet. En molts casos ja es satisfaran totes les restriccions d’integritat, sobretot quan aquestes només expressen dependències funcionals o multi-avaluades.

Però si hi ha restriccions violades llavors es posa en marxa tot el procés de reparació/detecció de violacions/reparació/detecció de violacions/reparació/... fins que no es detecten noves violacions o no es pot reparar una violació... o no s’acaba mai. El procediment funciona sota “backtracking”: en molts casos hi més d’una alternativa per reparar una restricció, se n’escull una i es continua, però més endavant hi ha una violació irreparable i s’ha de tornar enrera, desfer les insercions fetes des d’aquell punt i escollir una altra alternativa. Cal esgotar totes les possibilitats abans de concloure que l’esquema és insatisfactible.

Cada reparació, inserció d’un nou fet en el prototip, pot induir noves violacions de restriccions d’integritat. Per evitar tornar a comprovar de nou totes les restriccions, el mètode aplica una tècnica que només té en compte les restriccions “rellevants”, les que són “potencialment” violables per la reparació. Per determinar aquestes restriccions rellevants cal tenir en compte no només el fet que s’ha inserit com a reparació, si no també els possibles fets “induïts” per aquest, és a dir, els nous fets derivats resultants d’avaluar **IDB** sobre el prototip actualitzat.

Per reparar una restricció violada, es miren quins són els literals positius de la restricció que encara no estan en el prototip, els candidats a possibles reparacions. Aquests literals poden estar instanciats totalment o parcial, o no estar-ho gens. Com en el cas de [Kun84, Kun85], abans d’introduir una nova constant, s’intenta aprofitar les que ja apareixen en el prototip, però, a diferència d’aquell mètode, si aquesta solució falla es genera una constant nova.

Exemple 3.2.1:

$$\text{IDB} = \emptyset$$

$$\text{IC} = \left\{ \begin{array}{l} \forall x [\neg \text{Works_for}(x,x) \vee \exists y (\text{Works_for}(y,x) \wedge y \neq x)], \\ \exists x \text{Works_for}(x,x) \end{array} \right\}$$

1. S'inicialitza el prototip $\mathbf{D}_0 = \emptyset$
2. Es comproven les restriccions d'integritat avaluant-les contra \mathbf{D}_0 . La segona restricció, $\exists x \text{Works_for}(x,x)$, es viola.
3. Es genera un reparació $\mathbf{U} = \{\text{Works_for}(a,a)\}$ que actualitza el prototip $\mathbf{D}_1 = \mathbf{D}_0 \cup \mathbf{U}$.
4. Es Comproven quines restriccions són "rellevants" per la actualització $\mathbf{U} = \{\text{Works_for}(a,a)\}$. La primera restricció té un literal, $\neg \text{Works_for}(x,x)$, unificable amb \mathbf{U} .
5. Es genera una "instància", I , de la restricció rellevant resolent la restricció amb la reparació \mathbf{U} , $I = \exists y (\text{Works_for}(y,a) \wedge y \neq x)$.
6. S'avalua I contra \mathbf{D}_1 i es comprova, que, efectivament, es viola.
7. Es repara I inserint un àtom del tipus $\text{Works_for}(y,a)$.
 8. Abans de generar una nova variable per a y , es mira si en el prototip existeix un àtom unificable amb el que estem buscant. Existeix, $\text{Works_for}(a,a)$, però si es substitueix y per a , es provoca en una contradicció ($a \neq a$).
 9. Es genera una nova constant, b , tal que $\mathbf{U}' = \{\text{Works_for}(b,a)\}$.
10. Final: no hi ha cap restricció rellevant respecte $\mathbf{U}' = \{\text{Works_for}(b,a)\}$. \mathbf{S} és satisfactible perquè hem aconseguit generar un prototip on es satisfan les restriccions d'integritat. $\mathbf{D}_2 = \{\text{Works_for}(a,a), \text{Works_for}(b,a)\}$.

3.2.4 Discussió del mètode

Els autors assenyalen que llur mètode és consistent i complet per la satisfactibilitat finita. Cal fer notar, però, que aquesta satisfactibilitat finita es vista des del punt de vista de la teoria de models, en el sentit que apuntàvem en l'apartat 2.3.1.

Això queda palès en el fet de que quan es fan reparacions de restriccions violades, s'insereixen aquells àtoms instanciats que unifiquen amb els literals positius de la restricció, que són responsables de la seva violació al no estar presents encara en el prototip. Aquest àtoms instanciats poden ser tant a fets EDB com predicats IDB instanciats, malgrat que aquests no siguin derivables aplicant llurs regles de deducció sobre els fets EDB. Per tant, el prototips que construeix el mètode són models de $IDB \cup IC$, però no necessàriament estats consistents de la BDD.

3.3 El mètode de comprovació de la satisfactibilitat i la factibilitat de Marqués i Casamayor [MC93]

3.3.1 Plantejament

Els autors defineixen un esquema de BDD com una tupla $(\mathbf{BP}, \mathbf{CI})$ on

- \mathbf{BP} és un conjunt $\{Bp_1(x_{11}, \dots, x_{1n}), \dots, Bp_m(x_{m1}, \dots, x_{mp})\}$ d'esquemes de predicats bàsics, on cada x_{ij} és una variable.
- \mathbf{CI} és un conjunt de clàusules disjuntives positives ($\mathbf{CI} = \mathbf{IDB} \cup \mathbf{IC}$).

A partir d'aquí es defineixen un seguit de propietats sobre l'esquema de BDD $\mathbf{S} = (\mathbf{BP}, \mathbf{CI})$:

1. \mathbf{S} és un esquema consistent $\Leftrightarrow \mathbf{CI}$ és consistent
2. \mathbf{S} és un esquema no trivialment consistent $\Leftrightarrow \{\exists x_{11}, \dots, x_{1n} Bp_1(x_{11}, \dots, x_{1n}) \vee \dots \vee \exists x_{m1}, \dots, x_{mp} Bp_m(x_{m1}, \dots, x_{mp})\} \cup \mathbf{CI}$ és un conjunt consistent de clàusules.
3. \mathbf{S} és un esquema fortament consistent $\Leftrightarrow \{\exists x_{11}, \dots, x_{1n} Bp_1(x_{11}, \dots, x_{1n}) \wedge \dots \wedge \exists x_{m1}, \dots, x_{mp} Bp_m(x_{m1}, \dots, x_{mp})\} \cup \mathbf{CI}$ és un conjunt consistent de clàusules.
4. Sigui \mathbf{P} un subconjunt no buit de \mathbf{BP} . \mathbf{S} és un esquema \mathbf{P} -consistent $\Leftrightarrow \bigcup_i \{\exists x_{i1}, \dots, x_{ij} Bp_i(x_{i1}, \dots, x_{ij}) \mid Bp_i(x_{i1}, \dots, x_{ij}) \in \mathbf{P}\} \cup \mathbf{CI}$ és un conjunt consistent de clàusules.

La primera propietat interpreta la consistència de l'esquema des del punt de vista "clàssic" de la lògica. La segona propietat va més enllà i vol que al menys hi hagi un fet instanciat d'un predicat bàsic de l'esquema o, el que és el mateix, vol obtenir un estat vàlid no buit.

Si $\mathbf{P} = \{Bp_i(x_{i1}, \dots, x_{ij})\}$, un sol predicat bàsic, llavors la \mathbf{P} -consistència de \mathbf{S} seria equiparable a la factibilitat del predicat Bp_i . La consistència forta i la \mathbf{P} -consistència amb més d'un predicat podrien veure's com a casos particulars de la assolibilitat d'una fórmula F .

Per a comprovar aquestes propietats els autors no construeixen un prototip de BDD, sinó que a partir de l'esquema i de la propietat que es vol validar sobre aquest, es construeix una teoria de primer ordre, un conjunt de clàusules disjuntives, que es verifica mitjançant un demostrador de teoremes.

3.3.2 Delimitació de l'àmbit d'aplicació del mètode

Cl és un conjunt de clàusules disjuntives positives $A_1 \vee \dots \vee A_n \leftarrow B_1 \wedge \dots \wedge B_m$ on cada A_i, B_j és un àtom i $n \geq 0, m \geq 0, n+m \geq 1$.

Malgrat que, des d'un punt de vista lògic, les clàusules normals i les disjuntives positives són sintàcticament equivalents, com ara

$$C_1 = \text{Female}(x) \leftarrow \text{Person}(x) \wedge \neg \text{Male}(x) \qquad C_2 = \text{Female}(x) \vee \text{Male}(x) \leftarrow \text{Person}(x)$$

semànticament són diferents ja que no hi pot haver cap predicat bàsic de **BP** que aparegui en la capçalera d'una clàusula.

Sota aquest formalisme, les restriccions d'integritat són un cas particular de clàusula disjuntiva:

$$\leftarrow B_1 \wedge \dots \wedge B_m, \text{ on } m \geq 1.$$

Tampoc res no impediria tenir clàusules del tipus $A_1 \vee \dots \vee A_n \leftarrow$ on $n \geq 1$, el significat de les quals, en un context de BDD i tenint en compte que $A_1 \dots A_n$ són predicats derivats, no s'explica.

3.3.3 Funcionament del mètode

El procediment de validació de cadascuna de les propietats consta de dues fases. En la primera, a partir de l'esquema i de la especificació de la propietat es construeix un nou conjunt de clàusules disjuntives positives, de tal manera de que si es demostra que aquesta conjunt és consistent, llavors l'esquema aconsegueix la propietat. La segona fase és, precisament, comprovar que aquest conjunt és consistent i per fer-ho s'utilitza un mètode de refutació que intenta demostrar la inconsistència del conjunt de clàusules i que no és altra cosa que un demostrador de teoremes.

Els autors disposen de dos demostradors de teoremes basats en el principi de resolució lineal anomenats SLT-ALL i SLT-POS, que són extensions de la SLD-resolució per poder tractar amb clàusules disjuntives positives. SLT-POS és una versió pensada per funcionar més eficientment quan hi ha poques clàusules que tenen disjuncions en la capçalera, mentre que SLT-ALL és recomanable pel cas més general. Ambdós mètodes són complets i consistents per demostrar la (in)consistència d'un conjunt de clàusules.

Exemple 3.3.1:

$$\mathbf{BP} = \{ \text{Parent}(x_1, x_2), \text{Male}(x_3), \text{Female}(x_4) \}$$

$$\begin{aligned} \mathbf{CI} = \{ & \text{Sibling}(x, y) \leftarrow \text{Parent}(z, x) \wedge \text{Parent}(z, y) \\ & \text{Brother}(x, y) \vee \text{Sister}(x, y) \leftarrow \text{Sibling}(x, y) \\ & \leftarrow \text{Brother}(x, y) \wedge \text{Female}(x) \\ & \leftarrow \text{Sister}(x, y) \wedge \text{Male}(x) \\ & \leftarrow \text{Brother}(x, x) \\ & \leftarrow \text{Sister}(x, x) \} \end{aligned}$$

Suposem que $\mathbf{P} = \{ \text{Parent}(x_1, x_2) \}$ i que volem comprovar la \mathbf{P} -consistència de l'esquema anterior.

Segons la definició donada més amunt, l'esquema serà \mathbf{P} -consistent ssi $\{ \exists x_1, x_2 \text{Parent}(x_1, x_2) \} \cup$

\mathbf{CI} és consistent:

1. A partir d'aquesta definició es genera una teoria \mathbf{T} :

- se "skolemitzen" els esquemes de predicats bàsics substituint cada variable diferent per una constant diferent
- es substitueix cada $\leftarrow B_1 \wedge \dots \wedge B_m$ per $\text{Inconsistent} \leftarrow B_1 \wedge \dots \wedge B_m$, on Inconsistent és un predicat que no pertany a l'esquema.

$$\mathbf{T} = \{ \begin{array}{l} \text{C1. } \text{Parent}(a, b) \\ \text{C2. } \text{Sibling}(x, y) \leftarrow \text{Parent}(z, x) \wedge \text{Parent}(z, y) \\ \text{C3. } \text{Brother}(x, y) \vee \text{Sister}(x, y) \leftarrow \text{Sibling}(x, y) \\ \text{C4. } \text{Inconsistent} \leftarrow \text{Brother}(x, y) \wedge \text{Female}(x) \\ \text{C5. } \text{Inconsistent} \leftarrow \text{Sister}(x, y) \wedge \text{Male}(x) \\ \text{C6. } \text{Inconsistent} \leftarrow \text{Brother}(x, x) \\ \text{C7. } \text{Inconsistent} \leftarrow \text{Sister}(x, x) \} \end{array}$$

2. Es demostra la consistència de \mathbf{T} per refutació: si $\mathbf{T} \cup \{ \leftarrow \text{Inconsistent} \}$ té èxit, llavors

\mathbf{T} és inconsistent. En cas contrari, \mathbf{T} és consistent i l'esquema és \mathbf{P} -consistent.

$$\begin{array}{l}
\leftarrow \text{Inconsistent} \\
| \quad \text{C7} \\
\leftarrow \text{Sister}(x,x) \\
| \quad \text{C3} \quad \{y/x\} \\
\text{Brother}(x,x) \leftarrow \underline{\text{Sibling}(x,x)} \\
| \quad \text{C2} \quad \{y/x\} \\
\text{Brother}(x,x) \leftarrow \underline{\text{Parent}(z,x) \wedge \text{Parent}(z,x)} \\
| \quad \text{C1} \quad \{x/b, z/a\} \\
\text{Brother}(b,b) \\
| \quad \text{C6} \quad \{x/b\} \\
\text{Inconsistent} \\
| \quad \leftarrow \text{Inconsistent} \\
\quad \square
\end{array}$$

En aquest exemple, el procediment deriva la clàusula buida i, per tant, l'esquema no és \mathbf{P} -consistent respecte a $\mathbf{P} = \{\text{Parent}(x_1, x_2)\}$.

Un procediment anàleg seguiríem per comprovar la consistència no trivial de l'esquema de l'exemple 3.3.1. En aquest cas, la nova teoria que es generaria seria:

$$\mathbf{T}' = \{ \quad \begin{array}{l}
\text{C}'1. \quad \text{Parent}(a,b) \vee \text{Male}(c) \vee \text{Female}(d) \\
\text{C}'2. \quad \text{Sibling}(x,y) \leftarrow \text{Parent}(z,x) \wedge \text{Parent}(z,y) \\
\text{C}'3. \quad \text{Brother}(x,y) \vee \text{Sister}(x,y) \leftarrow \text{Sibling}(x,y) \\
\text{C}'4. \quad \text{Inconsistent} \leftarrow \text{Brother}(x,y) \wedge \text{Female}(x) \\
\text{C}'5. \quad \text{Inconsistent} \leftarrow \text{Sister}(x,y) \wedge \text{Male}(x) \\
\text{C}'6. \quad \text{Inconsistent} \leftarrow \text{Brother}(x,x) \\
\text{C}'7. \quad \text{Inconsistent} \leftarrow \text{Sister}(x,x) \quad \}
\end{array}$$

T' és consistent \Rightarrow l'esquema és no-trivialment consistent \Rightarrow l'esquema és consistent. En canvi, l'esquema 3.3.1 no és fortament consistent, ja que hem demostrat que existeix almenys un subconjunt $P \subseteq BP, P \neq \emptyset$, tal que l'esquema no és P -consistent.

3.3.4 Discussió del mètode

Com hem vist, els dos primers mètodes de validació que havíem estudiat fins ara només s'havien preocupat de la satisfactibilitat/consistència de l'esquema de BDD. Una de les aportacions [MC93] ha estat ampliar el ventall de propietats que caldria validar i proporcionar un mètode comú per a tractar-les de manera uniforme. És evident que les quatre propietats esmentades estan estretament relacionades entre elles, però no deixa de ser un encert introduir propietats tant interessants com la consistència no trivial i la P -consistència.

Pel que fa al mètode en si, és consistent i complet, ja que ho són els demostradors de teoremes que utilitzen com a procediments de refutació i, també, pel fet que la skolemització dels esquemes de predicats bàsics preserva la consistència (o inconsistència) de la teoria generada.

Precisament aquest punt de la skolemització dels esquemes de predicats bàsics ens hauria de sobtar una mica. Hem vist en els mètodes precedents que la introducció de noves constants per substituir variables era un veritable handicap en determinats casos. En canvi en [MC93] sembla que solucionin la qüestió de forma expeditiva: per cada variable diferent, una constant distinta. En alguns casos, però, si introduíssim la igualtat la cosa no aniria tant bé. Heus aquí un exemple senzill:

$$BP = \{Works_for(x_1, y_2)\}$$

$$CI = \{\leftarrow Works_for(x, y) \wedge x \neq y\}$$

En aquest cas, el mètode trobaria que l'esquema no és no-trivialment consistent, quan $\{Works_for(a, a)\}$ podria ser un estat no buit perfectament vàlid.

La introducció de relacions d'igualtat/desigualtat podrien distorsionar alguns resultats, però, en canvi, quan es tracta de comprovar "només" la consistència de l'esquema, que era precisament el únic que comprovaven els mètodes precedents, no s'utilitzen en cap moment els esquemes de predicats bàsics ni, per tant, la skolemització. És més, el procediment de refutació de CI , que és tal com es comprova la (in)consistència de l'esquema, en cap moment necessita instanciar cap variable.

Tot això és en bona part possible gràcies al fet de que les restriccions d'integritat, tal com han estat definides, no poden tenir literals negatius en els cossos de les clàusules. No es poden definir, doncs, restriccions com les que expressen dependències entre de predicats bàsics (exemple: $\forall xy$ $Mother(x,y) \Rightarrow Female(x)$, on $Female(x)$ i $Mother(x,y)$ són predicats bàsics). Evitant aquest tipus de restriccions d'integritat, els autors s'estalvien molts problemes, però perden potència i expressivitat. Un corol·lari de tot això és que la major part de les vegades els esquemes basats en el formalisme de [MC93] seran consistents, i, per tant, satisfactibles en l'estat buit.

3.4 *Factibilitat i P-assolibilitat de predicats en Levy et al. [LMSS93]*

3.4.1 Plantejament

Els autors no tenen un concepte d'esquema de BDD tal com els que hem vist fins ara. L'ur punt de vista està enfocat principalment a tot el que fa referència a les consultes contra la BDD. Contra la BDD es poden definir tantes consultes com es vulgui. Una consulta **P** és un programa -conjunt de clàusules- on hi ha un predicat IDB que és el predicat de consulta *Q* (*query predicate*). Els resultat d'una consulta **P** contra una estat de la BDD **D**, **P(D)**, i és el conjunt de tots els fets resultants d'aplicar **P** sobre **D** i que es poden unificar amb *Q*. Consultes diferents tenen predicats IDB distints.

Sota aquest paradigma els autors defineixen dues propietats:

1. Un predicat IDB *R* d'una consulta **P** és satisfactible si existeix un estat de la BDD **D** en el qual **P(D)** obté almenys una instanciació de *R* (un fet sobre *R*).
2. Sigui $W(\alpha_1, \dots, \alpha_n)$ un àtom d'un predicat EDB o IDB on cada α_i pot ser una variable o una constata. L'àtom $W(\alpha_1, \dots, \alpha_n)$ és P-assolible (*query-reachable*) si existeix un estat de la BDD **D** en el que una instanciació completa d'aquest àtom participa en el càlcul del predicat de query *Q*.

Per integrar aquest enfocament en el nostre concepte d'esquema de BDD només caldria considerar les diferents consultes com a subconjunts disjunts de l'esquema total de la BDD. D'aquesta manera, la propietat que els autors anomenen satisfactibilitat és clarament equiparable al concepte de factibilitat d'un predicat en un esquema de BDD, però aplicant-ho només als predicats IDB. La segona propietat es més específica i la podríem entendre com una forma d'estudiar les interrelacions entre predicats i/o fets.

3.4.2 Delimitació de l'àmbit d'aplicació del mètode

Els autors, prenent com a base els programes datalog -conjunts finits de clàusules definides sense funcions-, estudien les propietats anteriors en funció de dues possibles “extensions” al datalog: la negació i els predicats aritmètics de comparació (\neq , $=$, $<$, \leq). Els resultats són els següents:

- La satisfactibilitat d'un predicat IDB i la P-assolibilitat d'un àtom són problemes decidibles pel que fa a consultes datalog amb predicats de comparació i negació restringida als predicats EDB.
- Idem amb respecte a consultes datalog amb predicats EDB unaris i negació estratificada tant per predicats EDB com IDB.
- La satisfactibilitat d'un predicat IDB i la P-assolibilitat d'un àtom són problemes indecidibles quan s'apliquen a extensions del datalog més amplies.

En aquest enfocament centrat en consultes no hi ha, o no es tenen en consideració, restriccions d'integritat ($IC = \emptyset$).

3.4.3 Funcionament del mètode.

Els autors proposen un mètode per comprovar la P-assolibilitat que van adaptant a mesura que van ampliant el datalog amb les extensions que volen estudiar, fins arribar a les conclusions que hem vist en l'apartat anterior. La satisfactibilitat d'un predicat la podem considerar un sub-producte del mètode ja que es pot comprovar en les primeres fases.

En aquest context sense restriccions d'integritat i fortament orientat a les consultes, no és d'estranyar que aquest mètode tampoc intenti construir un prototip de BDD, encara que en la definició de les propietats que es volen validar s'expliciti que almenys hi d'haver “un estat de la BDD \mathbf{D} ” on aquestes es compleixin.

En línies generals, el mètode intenta construir un arbre de derivació (*query-tree*) que partint del predicat de consulta Q arribi als predicats EDB que, en darrera instància, són la base dels possibles resultats de Q . Per fer-ho s'hauran de “desgranar” els càlculs intermedis: els predicats IDB i llurs regles de derivació. Les dificultats apareixen quan hi ha recursivitat, negació i predicats de comparació.

Exemple 3.4.1:

$P = \{$ C1. $Works_for(x,y) \leftarrow Boss(x,y)$
C2. $Works_for(x,y) \leftarrow Boss(x,z) \wedge \neg Staff(x) \wedge \neg Staff(z) \wedge Works_for(z,y)$
C3. $Q(x,y) \leftarrow Works_for(x,y) \wedge Staff(y) \wedge \neg Boss(x,y) \}$

Predicats IDB: $Q(x,y)$ i $Works_for(x,y)$

Predicats EDB: $Boss(x,y)$ i $Staff(x)$

Predicat de consulta: $Q(x,y)$.

Construcció del *query-tree*:

1. Generació dels *EDB-labels*: A cada predicat IDB li associen els predicats EDB necessaris per calcular-lo. Però només podem associar-li aquells predicats EDB les variables dels quals apareixen també en el predicat IDB. Per trobar els *EDB-labels* d'un predicat IDB cal tenir en compte els *EDB-labels* dels altres predicats IDB que puguin aparèixer en el cos de les seves regles de definició:

C1: es genera el *EDB-label* $L1 = [Works_for(x,y), \{Boss(x,y)\}]$

C2: utilitzant L1 es genera $L2 = [Works_for(x,y), \{\neg Staff(x)\}]$

utilitzant L2 (C2 és recursiva) es té $L3 = [Works_for(x,y), \{\neg Staff(x)\}]$.

L2 i L3 són isomorfs: no es generaran més *EDB-labels* per a $Works_for(x,y)$

C3: utilitzant L1 es genera $L4 = [Q(x,y), \{Boss(x,y), Staff(y), \neg Boss(x,y)\}]$.

L4 és inconsistent: no es té en compte.

utilitzant L2 es genera $L5 = [Q(x,y), \{\neg Staff(x), Staff(y), \neg Boss(x,y)\}]$

2. Rescriptura de P substituint els predicats IDB i llurs regles fent totes les combinacions possibles amb els *EDB-labels* consistents que s'han trobat (L1,L2 i L5):

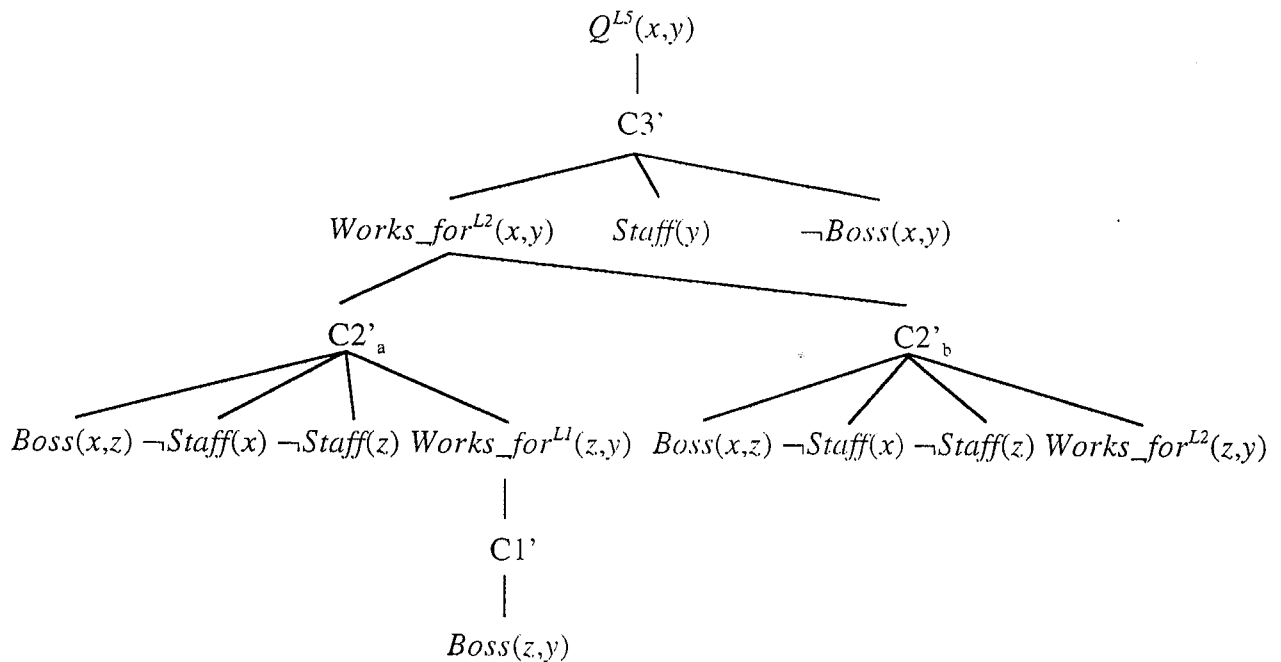
$P' = \{$ C1': $Works_for^{L1}(x,y) \leftarrow Boss(x,y)$

C2'_a: $Works_for^{L2}(x,y) \leftarrow Boss(x,z) \wedge \neg Staff(x) \wedge \neg Staff(z) \wedge Works_for^{L1}(z,y)$

C2'_b: $Works_for^{L3}(x,y) \leftarrow Boss(x,z) \wedge \neg Staff(x) \wedge \neg Staff(z) \wedge Works_for^{L2}(z,y)$

C3': $Q^{L5}(x,y) \leftarrow Works_for^{L2}(x,y) \wedge Staff(y) \wedge \neg Boss(x,y) \}$

3. Generació del *query-tree* per a P' : a partir del predicat de consulta generarem un arbre de tal manera que per cada predicat IDB generem tants subarbres com regles de derivació el defineixin. L'objectiu es que a les fulles de l'arbre apareguin els predicats EDB, però poden haver-hi també predicats IDB si llurs subarbres ja han estat expandits:



- Test de satisfactibilitat: Un predicat IDB és satisfactible respecte a una consulta P si li pot generar almenys una *EDB-label* consistent en la primera fase del mètode. En l'exemple, tant $Q(x,y)$ com $Works_for(x,y)$ són satisfactibles.

Noteu que amb

$$C2'' : Works_for(x,y) \leftarrow Boss(x,z) \wedge \neg Staff(z) \wedge Works_for(z,y)$$

una *EDB-label* hagués estat $L2'' = [Works_for(x,y), \emptyset]$, també consistent.

- Test de P-assolibilitat: Un àtom és assolible respecte a una consulta P si es pot unificar almenys amb un dels nodes del *query-tree* generat a partir de Q . $Staff(a)$, $\neg Staff(b)$, $Works_for(x,c)$ són exemples d'àtoms assolibles respecte a P . En canvi, $\neg Works_for(z,y)$ no és P -assolible.

3.4.4 Discussió del mètode

El principal problema que li veiem a [LMSS93] no és tant del mètode com de l'enfocament que en fan els autors. El fet de no considerar restriccions d'integritat creiem que li treu bona part de la "gràcia", quan ja molts sistemes comercials (relacionals) incorporen mecanismes per a llur tractament.

Tanmateix, voldríem remarcar una de les aportacions dels autors, que no és altra que el concepte de P-assolibilitat (*query-reachability*), que cap autor ni anterior ni posterior ha tingut en consideració. Potser la P-assolibilitat més que un problema de validació (eficàcia) planteja una tema d'optimització (eficiència): De què ens serveix mantenir informació (fets) sobre determinats predicats si després potser no s'utilitzarà mai en cap aplicació?

3.5 El mètode de comprovació de la redundància relativa de restriccions d'integritat de Gupta et al. [GSUW94]

3.5.1 Plantejament

La intenció dels autors és proposar un seguit d'estratègies que permetin fer els tests de consistència d'una BDD el més eficients possibles, tot minimitzant els accessos a les dades emmagatzemades, especialment si aquestes són remotes (BDD distribuïdes). Del que es tracta és explotar el coneixement apriorístic sobre les restriccions per tal d'assegurar llur satisfacció sense haver de fer-ne una avaluació completa. Per tant, els autors proposen un seguit de tests que intenten comprovar la satisfacció de les restriccions amb un mínim d'informació. Si el test té èxit, les restriccions es satisfan, si no, s'aplica un altre test que utilitzi més informació.

El test que s'ha d'efectuar abans que tots utilitza "només" la informació de l'esquema: donat un conjunt de restriccions d'integritat \mathbf{IC} , es tracta de veure quines són les restriccions que es satisfan sempre que es satisfacin totes les altres i que, per tant, no caldrà comprovar cada cop que s'actualitzi la BDD. D'aquesta manera, els autors identifiquen la propietat de subsumpció d'una restricció per altres restriccions: Sigui I una restricció d'integritat i $\mathbf{C} = \{I_1, \dots, I_n\}$ un conjunt de restriccions d'integritat, direm que \mathbf{C} subsumeix I , o que I es subsumida per \mathbf{C} , si sempre que es viola I hi ha almenys una I_i que també es viola.

La subsumpció de restriccions pot ser una altra manera d'entendre la redundància relativa d'una restricció d'integritat, però enfocant-ho des del punt de vista de la violació de la restricció: una restricció d'integritat $I_j \in \mathbf{IC}$ és relativament redundant si $\mathbf{IC} - \{I_j\}$ subsumeix I_j .

El mètode que els autors proposen per la comprovació de la subsumpció de restriccions és expressar aquesta propietat en termes d'un problema de conteniment de consultes, *query containment*, i resoldre'l amb les tècniques existents.

Siguin \mathbf{P}_1 i \mathbf{P}_2 dos consultes, conjunts finit de clàusules, amb un mateix predicat de consulta Q i un estat arbitrari de la BDD \mathbf{D} . Direm que \mathbf{P}_1 està contingut a \mathbf{P}_2 , $\mathbf{P}_1 \subseteq^c \mathbf{P}_2$, si totes les respostes a \mathbf{P}_1 contra \mathbf{D} respecte Q ho són també a \mathbf{P}_2 : $\bigcup_i \{Q\sigma_i \mid Q\sigma_i \in \mathbf{P}_1(\mathbf{D})\} \subseteq \bigcup_k \{Q\sigma_k \mid Q\sigma_k \in \mathbf{P}_2(\mathbf{D})\}$, on $Q\sigma_s$ és una instanciació de Q i $\mathbf{P}_r(\mathbf{D})$ és (el *fixpoint* mínim de) l'avaluació (*bottom-up*) de \mathbf{P}_r sobre \mathbf{D} .

3.5.2 Delimitació de l'àmbit d'aplicació del mètode

Les limitacions sobre el tipus de clàusules que conformen tant **IC** com **IDB** depenen de l'estat de l'art en *query containment*, ja que aquest és encara un camp de recerca obert.

Els resultats coneguts fins ara són:

- El conteniment de dos programes datalog sense negació i que no siguin recursius tots dos alhora és un problema decidible
- Altrament, és indecidible
- Existeix una propietat suficient però no necessària del *query containment* que és l'*uniform containment* que és decidible per a programes datalog recursius.
- Hi ha algorismes no complets per comprovar l'*uniform containment* en programes datalog amb negació estratificada.

3.5.3 Funcionament del mètode

Per expressar la propietat de subsumpció de restriccions en termes d'un problema de *query containment* necessitem definir o transformar les restriccions d'integritat en forma de clàusules normals definint un mateix predicat 0-ari que no pertany a **IDB**, tal com es feia a [MC93]:

$$\text{Inconsistent} \leftarrow L_1 \wedge \dots \wedge L_n$$

on L_i és un literal positiu o negatiu corresponent a un predicat EDB o IDB. D'aquesta manera, *Inconsistent* fa una mica les funcions d'un predicat de consulta.

$C = \{I_1, \dots, I_n\}$ subsumeix $I \Leftrightarrow \{I'\} \cup \text{IDB} \subseteq^c C' \cup \text{IDB}$ respecte *Inconsistent*, on I' i C' són el resultat d'aplicar la transformació abans esmentada sobre I i C , respectivament.

Els autors expliquen com transformar un problema de subsumpció de restriccions d'integritat en un problema de conteniment de consultes, però no expliquen com funciona aquesta tècnica, entre altres coses perquè la implementació i resultats concrets varien segons els tipus de clàusules amb les que es vulgui treballar i les possibles extensions que es vulguin incorporar, com ara els predicats aritmètics de comparació. Per a un major aprofundiment en el tema recomanem la lectura de [ST95]. Tanmateix i en general, podem dir que els mètodes que volen comprovar que $P_1 \subseteq^c P_2$ treballen en dues fases quan no hi ha negació:

1. A partir de P_1 es genera un prototip de BDD D , de tal manera que existeix una instància positiva $Q\sigma_s$ que pertany a $P_1(D)$.
2. Si $Q\sigma_s \in P_2(D)$ llavors $P_1 \subseteq^c P_2$

Exemple 3.5.1:

$$\mathbf{IDB} = \{ \text{Staff_manager}(x) \leftarrow \text{Boss}(y,x) \wedge \text{Staff}(y) \}$$

$$\mathbf{IC} = \{ I_1: \leftarrow \text{Boss}(x,y) \wedge \text{Staff_manager}(x) \\ I_2: \leftarrow \text{Boss}(x,y) \wedge \text{Staff}(x) \}$$

$$\mathbf{IC}' = \{ I'_1: \text{Inconsistent} \leftarrow \text{Boss}(x,y) \wedge \text{Staff_manager}(x) \\ I'_2: \text{Inconsistent} \leftarrow \text{Boss}(x,y) \wedge \text{Staff}(x) \}$$

- $\{I_2\}$ subsumeix I_1 : $\{I'_1\} \cup \mathbf{IDB} \subseteq^c \{I'_2\} \cup \mathbf{IDB}$

1. Generem el prototip $D = \{ \text{Boss}(a,b), \text{Boss}(c,a), \text{Staff}(c) \}$ de tal manera que $(\{I'_1\} \cup \mathbf{IDB})(D) = \{ \text{Boss}(a,b), \text{Boss}(c,a), \text{Staff}(c), \text{Staff_manager}(a), \text{Inconsistent} \}$

2. $\text{Inconsistent} \in (\{I'_2\} \cup \mathbf{IDB})(D) = \{ \text{Boss}(a,b), \text{Boss}(c,a), \text{Staff}(c), \text{Inconsistent} \}$

- $\{I_1\}$ no subsumeix I_2 : $\{I'_2\} \cup \mathbf{IDB} \not\subseteq^c \{I'_1\} \cup \mathbf{IDB}$

3. Generem el prototip $D = \{ \text{Boss}(a,b), \text{Staff}(a) \}$ de tal manera que

$$(\{I'_2\} \cup \mathbf{IDB})(D) = \{ \text{Boss}(a,b), \text{Staff}(a), \text{Inconsistent} \}$$

4. $\text{Inconsistent} \notin (\{I'_1\} \cup \mathbf{IDB})(D) = \{ \text{Boss}(a,b), \text{Staff}(a), \text{Staff_manager}(b) \}$

3.5.4 Discussió del mètode

Tal com argumentàvem en el cas de [MC93], el fet de poder definir restriccions d'integritat amb literals negats augmenta tant la potència expressiva de la BDD com la nostra dificultat en solventar els problemes que plantegen. Per desgràcia, l'estat de l'art del *query containment* és bastant descoratjador pel que fa a l'ús de la negació.

3.6 Validació d'esquemes de Bases de Dades Deductives en Decker et al. [DTU96]

3.6.1 Plantejament

Partint de la premissa que la validació d'esquemes de bases de dades no és un problema que sigui fàcilment formalitzable i menys encara automatitzable, els autors proposen un seguit de "tasques" que poden servir com a eines de suport als dissenyadors. Aquestes tasques són la comprovació de que l'esquema satisfà algunes de les propietats que ja hem vist fins ara com, per exemple, la satisfactibilitat o la factibilitat d'un predicat. La motivació principal dels autors és proposar un únic mètode per tractar totes aquestes tasques diferents. Aquest mètode no és altre que l'actualització de vistes.

El problema de l'actualització de vistes consisteix en que donada una petició d'actualització de la BDD, que tant pot incloure insercions com esborrats de àtoms EDB i/o IDB, es tracta de trobar una "traducció" en termes de insercions i/o esborrats només de fets EDB, de tal manera que la BDD actualitzada amb aquesta traducció satisfaci la petició inicial.

El que proposen els autors és definir declarativament cada propietat a validar en termes d'un predicat IDB 0-ari exclusiu P . Partint de l'estat inicial de la BDD $\mathbf{D}_0 = (\mathbf{IDB}, \mathbf{IC}, \emptyset)$, es fa la petició d'inserir P . Llavors el mètode de actualització de vistes intentarà trobar una traducció \mathbf{U} que contingui insercions i/o esborrats de fets EDB. Si la troba, voldrà dir que existeix un estat, $\mathbf{U}(\mathbf{D}_0)$, on P és cert i, per tant, on la propietat es satisfà. Per assegurar que a més $\mathbf{U}(\mathbf{D}_0)$ és un estat vàlid de la BDD, en el cos de la clàusula que defineix P s'haurà d'incloure també la condició de satisfactibilitat.

Les propietats de validació que els autors aborden i llur formulació són les que hem pres com a base d'aquest treball (apartat 2.3).

3.6.2 Delimitació de l'àmbit d'aplicació del mètode

Tenim un esquema $S = (IDB, IC)$ on

- **IDB** és un conjunt finit de clàusules normals permissibles amb negació estratificada
- **IC** és un conjunt de restriccions d'integritat representades en forma de denegació.

El mètode concret d'actualització de vistes que proposen els autors, [TO95], és consistent i complet en aquestes condicions, encara que en presència de clàusules recursives pot tenir una terminació no finita. A això cal afegir el fet que, en general, la satisfactibilitat i les altres propietats són problemes semidecidibles.

3.6.3 Funcionament del mètode

Els autors no ho fan i tampoc és el propòsit d'aquest treball explicar com funciona un o més mètodes d'actualització de vistes. De fet els autors afirmen que llur plantejament es prou general com per aplicar qualsevol mètode d'actualitzacions de vistes, sempre i quan es compleixin dues condicions essencials:

1. La classe de BDD suportada pel mètode ha de permetre definir les propietats de validació de forma declarativa.
2. En cas d'haver-hi un o més traduccions que satisfacin un determinada petició d'actualització, el mètode en trobarà almenys una.

Abans de definir de manera declarativa les propietats de validació mitjançant un predicat exclusiu, cal fer una sèrie de transformacions en l'esquema $S = (IDB, IC)$:

$$\begin{aligned} IDB' = & IDB \cup \bigcup_i \{ Inconsistent_i \leftarrow L_{i1} \wedge \dots \wedge L_{in} \mid \leftarrow L_{i1} \wedge \dots \wedge L_{in} \in IC \} \\ & \cup \bigcup_i \{ Inconsistent \leftarrow Inconsistent_i \} \\ & \cup \{ Satisfied \leftarrow \neg Inconsistent \} \end{aligned}$$

on *Satisfied*, *Inconsistent_i* i *Inconsistent* són predicats que no pertanyen a IDB.

Un esquema S satisfarà una propietat de validació representada mitjançant una regla deductiva C que defineix un predicat 0-ary P , ssi la petició d'inserció de P , denotada com $\leftarrow Insert(P)$, sobre un estat inicial "buit" $D^x_0 = (IDB' \cup \{C\}, \emptyset, \emptyset)$ és satisfactible, és a dir, existeix un traducció U de $\leftarrow Insert(P)$ aplicant actualització de vistes tal que $U(D^x_0)$ és un estat vàlid de S :

1. S és satisfactible $\Leftrightarrow \leftarrow Insert(Satisfied)$ és satisfactible en $D^1_0 = (IDB', \emptyset, \emptyset)$

2. Un predicat Q és factible en $S \Leftrightarrow \leftarrow Insert(Lively_Q)$ és satisfactible en

$$D^2_0 = (IDB' \cup \{Lively_Q \leftarrow Q(x_1, \dots, x_n) \wedge Satisfied\}, \emptyset, \emptyset)$$

3. Un fórmula F expressada com una conjunció de literals és assolible en S

$\Leftrightarrow \leftarrow Insert(Reachable_F)$ és satisfactible en

$$D^3_0 = (IDB' \cup \{Reachable_F \leftarrow F \wedge Satisfied\}, \emptyset, \emptyset)$$

Les propietats de validació que s'han de satisfer en cada estat de S s'han de comprovar indirectament per refutació:

4. Una restricció d'integritat $I_i = \leftarrow L_{i1} \wedge \dots \wedge L_{in} \in IC$ és absolutament redundant

$\Leftrightarrow \leftarrow Insert(Inconsistent)$ NO és satisfactible en

$$D^4_0 = (IDB \cup \{Inconsistent \leftarrow L_{i1} \wedge \dots \wedge L_{in}\}, \emptyset, \emptyset)$$

5. Una restricció d'integritat $I_i = \leftarrow L_{i1} \wedge \dots \wedge L_{in} \in IC$ és relativament redundant

$\Leftrightarrow \leftarrow Insert(Not_redundant)$ NO és satisfactible en $D^5_0 = (IDB^5, \emptyset, \emptyset)$ on

$$IDB^5 = (IDB' - \{Inconsistent \leftarrow Inconsistent_i\})$$

$$\cup \{Not_redundant \leftarrow Inconsistent_i \wedge Satisfied\}$$

Exemple 3.6.1:

IDB = { *Sibling*(*x,y*) ← *Parent*(*z,x*) ∧ *Parent*(*z,y*)
Brother(*x,y*) ← *Sibling*(*x,y*) ∧ *Male*(*x*)
Sister(*x,y*) ← *Sibling*(*x,y*) ∧ *Female*(*x*) }

IC = { ← *Brother*(*x,y*) ∧ *Sister*(*x,z*)
← *Sibling*(*x,x*) }

Primer de tot cal construir **IDB'**:

IDB' = { *Sibling*(*x,y*) ← *Parent*(*z,x*) ∧ *Parent*(*z,y*)
Brother(*x,y*) ← *Sibling*(*x,y*) ∧ *Male*(*x*)
Sister(*x,y*) ← *Sibling*(*x,y*) ∧ *Female*(*x*) }
*Inconsistent*₁ ← *Brother*(*x,y*) ∧ *Sister*(*x,z*)
*Inconsistent*₂ ← *Sibling*(*x,x*) }
Inconsistent ← *Inconsistent*₁
Inconsistent ← *Inconsistent*₂
Satisfied ← ¬*Inconsistent* }

- **S** = (**IDB,IC**) és satisfactible:

Amb ← *Insert*(*Satisfied*) en **D**₀¹ = (**IDB'**,∅,∅) el mètode obté la traducció **U** = { }, ja que ¬*Inconsistent* ja es cert en **D**₀¹

- *Parent* no és factible en **S**:

D₀² = (**IDB'** ∪ { *Lively_Parent* ← *Parent*(*z,x*) ∧ *Satisfied* }, ∅, ∅)

Amb ← *Insert*(*Lively_Parent*) el mètode satisfà primer ← *Insert*(*Parent*(*z,x*)) tot generant la traducció parcial **U****p** = { *Insert*(*Parent*(*a,b*)) }, però al resoldre després ← *Insert*(*Satisfied*) sobre **D**₀² ∪ { *Parent*(*a,b*) }, el mètode fracassa al deduir-se *Inconsistent* de *Sibling*(*b,b*).

Per idèntics motius són també no factibles *Sibling*, *Brother* i *Sister*

- $\leftarrow \text{Brother}(x,y) \wedge \text{Sister}(x,z)$ és relativament redundant:

$$\text{IDB}^5 = (\text{IDB}' - \{\text{Inconsistent} \leftarrow \text{Inconsistent}_1\}) \cup \{\text{Not_redundant} \leftarrow \text{Inconsistent}_1 \wedge \text{Satisfied}\}$$

$\leftarrow \text{Insert}(\text{Not_redundant})$ és clarament insatisfactible ja que, tal com passava en el cas anterior, ni *Brother* ni *Sister* són factibles en $\mathbf{S}^5 = (\text{IDB}^5, \emptyset)$.

3.6.4 Discussió del mètode

Podem dir que [DTU95] realitza dues grans aportacions a la validació de BDD. Per una banda identifica, amplia i formalitza de manera no ambigua tot un ventall de propietats sobre validació que d'una manera o altra havien estat abordades per autors anteriors. I de l'altra, proposen un mètode únic per tractar-les de manera uniforme mitjançant les tècniques l'actualització de vistes. El mètode resta obert en el sentit de que qualsevol nova propietat de validació expressable declarativament segons el formalisme proposat pels autors pot beneficiar-se d'aquest enfocament.

Amb tot, no hem de veure l'actualització de vistes com a una panacea. Uns dels problemes principals continua essent el de la introducció de noves constants quan es necessita generar nous fets per reparar, per exemple, alguna restricció d'integritat violada. El mètode és complet sota la hipòtesi de dominis de variables finits. A la pràctica això s'implementa o bé demanant-li a l'usuari directament que introdueixi ell mateix la constant, nova o no, que vulgui o bé assignar valors per defecte, cosa que implicà tenir emmagatzemat en algun lloc tots els valors dels diferents dominis. La primera solució és més justificable en un context de funcionament "normal" de la BDD: un usuari llença una petició d'actualització contra la BDD, però el SGBD pot necessitar informació addicional que demana a l'usuari. En canvi, en un context de validació estariem obligant a l'usuari que anés inventant "sobre la marxa" dades fictícies. La segona opció obliga a prendre la decisió de quin és el tamany mínim per cada domini a partir del qual poden considerar els tests de validació prou realístics.

Actualment s'està treballant en perfeccionar els mètode d'actualitzacions de vistes [TO95] per tal d'incorporar-hi algunes tècniques de la programació lògica amb restriccions (*Constraint Programming*), cosa que sens dubte tindrà un impacte positiu en tota aquesta problemàtica de introducció de constants, ja que precisament una de les "gràcies" de la programació lògica amb restriccions es postergar el màxim possible la instanciació dels literals tot generant respostes "qualificades" on no és diu quin són els valors concrets de les variables però sí els rangs on poden ocórrer.

4. Conclusions

En tots els articles que hem vist, hom planteja la validació de l'esquema com la comprovació de que aquest satisfà un seguit de propietats "desitjables". La satisfactibilitat de l'esquema és la propietat "primària", la més important ja que no té sentit plantejar-se les altres sense haver-la resolt abans. La factibilitat fa un pas més enllà comprovant que els predicats de l'esquema són també "satisfactibles", és a dir, poden tenir fets instanciats en algun moment de la vida de la base de dades. La assolibilitat és una propietat més "fina", en el sentit que permet al dissenyador/validador fer simulacions sobre possibles situacions reals, determinar si certes condicions excepcionals han estat previstes. Finalment, la detecció de redundàncies la podem entendre com una activitat complementària de refinament de l'esquema.

Els mètodes que s'han utilitzat per comprovar aquestes propietats han adaptat tècniques de raonament automatitzat utilitzades en altres camps: demostració de teoremes, processament de consultes, conteniment de consultes, actualització de vistes. Malauradament, hem vist que, pel cas general, les propietats de validació són semidecidibles. Alguns autors han preferit restringir l'àmbit d'aplicació per obtenir resultats decidibles, però en detriment de la potència i expressivitat de la base de dades deductiva. Els primers mètodes estaven més a prop de la programació lògica que de les bases de dades, considerant la satisfactibilitat des del punt de vista de la teoria de models i utilitzant bàsicament demostradors de teoremes. Creiem que [DTU96] ha encetat una nova etapa pel que fa als mètodes de validació, compendiant totes les propietats que s'havien tractat fins aleshores, definint-les en termes d'estats consistents de la base de dades i proposant un mètode d'èxit provat en el camp de l'actualització de vistes.

Agraïments

Voldria donar les gràcies a Fèlix Saltor, Ernest Teniente i Toni Urpí per llurs comentaris i suggeriments.

Referències:

- [ABC82] W.R. Adrion, M.A. Branstad, J.C. Cherniavsky: "Validation, Verification and Testing of Computer Software", *ACM Computing Surveys*, Vol. 14, No. 2, pp. 159-192, 1982.
- [BDM88] F. Bry, H. Decker, R. Manthey: "A uniform approach to constraint satisfaction and constraint satisfiability in deductive databases", In J.W. Schmidt, S. Ceri, M. Missikoff (Eds.): *Advances in Database Technology -EDBT'88. Proceedings of the International Conference on Extending Database Technology*, pp. 488-505. Lecture Notes in Computer Science, Vol. 303, Springer, 1988.
- [BM86] F. Bry, R. Manthey: "Checking consistency of database constraints: a logical basis", In W.W. Chu, G. Gardarin, S. Ohsuga, Y. Kambayashi (Eds.): *Proceedings of the 12th International Conference on Very Large Data Bases*, pp. 13-20. Morgan Kaufmann, 1986.
- [CTUF96] D. Costal, E. Teniente, T. Urpí, C. Farré: "Handling Conceptual Model Validation by Planning", In P. Constantopoulos, J. Mylopoulos, Y. Vassiliou (Eds.): *Advances in Information System Engineering, Proceedings of the 8th International Conference, CAiSE'96*, pp. 255-271. Lecture Notes in Computer Science, Vol. 1080, Springer, 1996.
- [Das92] S.K. Das: *Deductive Databases and Logic Programming*, Addison-Wesley, 1992
- [DTU96] H. Decker, E. Teniente, T. Urpí: "How to tackle schema validation by view updating", In P. Apers, M. Bouzeghoub, G. Gardarin (Eds.): *Advances in Database Technology - EDBT'96, Proceedings of the 5th International Conference on Extending Database Technology*, pp. 535-549. Lecture Notes in Computer Science, Vol. 1057, Springer, 1996.
- [FW95] R. Feenstra, R. Wieringa: "Validating Database Constraints and Updates Using Automated Reasoning Techniques", In B. Thalheim (Ed.): *Proceedings of the Workshop on Semantics in Databases*, pp. 24-32. Fakultät für Mathematik, Naturwissenschaften und Informatik, Technische Universität Cottbus, 1995. Technical report.
- [GSUW94] A. Gupta, Y. Sagiv, J.D. Ullman, J. Widom: "Constraint Checking with Partial Information", *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 45-55. ACM Press, 1994.

- [Kun84] C.H. Kung: *A Temporal Framework for Information Systems Specifications and Verification*, PhD Thesis, University of Trondheim, Norway, 1984.
- [Kun85] C.H. Kung: "A Tableaux Approach for Consistency Checking", In A. Sernadas, J. Bubenko, A. Olivé (Eds.): *Information Systems: Theoretical and Formal Aspects*, pp. 191-207, Elsevier Science Publishers, North-Holland, 1985.
- [LMN+93] P.C Lockemann, G. Moerkotte, A. Neufeld, K. Radermacher, N.Runge: "Database design with user-definable modelling concepts", *Data & Knowledge Engineering 10*, pp. 229-257, North-Holland, 1993.
- [LMSS93] A. Levy, I.S. Mumick, Y. Sagiv, O. Shmueli: "Equivalence, query-reachability and satisfiability in Datalog extensions", *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 109-122. ACM Press, 1993.
- [Llo87] J.W. Lloyd: *Foundations of Logic Programming*, Springer, 1987
- [MC93] F. Marqués, J.C. Casamayor: "Consistency Verification of Deductive Database Schemes", In A. Olivé (Ed.): *Proceedings of the 4th International Workshop on the Deductive Approach to Information Systems and Databases*, pp. 287-307. Report de recerca LSI/93-25-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1993.
- [ST95] M. Staudt, K.v. Thadden: "Subsumption Checking in Knowledge Bases", Technical Report, RWTH Aachen num. 95-11, 1995.
- [TAL92] B. Theodoulidis, P. Alexakis, P. Loucopoulos: "Verification and Validation of Temporal Business Rules", In A. Olivé (Ed.): *Proceedings of the 3rd International Workshop on the Deductive Approach to Information Systems and Databases*, pp. 1-15. Report de recerca, LSI/92/19, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1992.
- [TO95] E. Teniente, A. Olivé: "Updating Knowledge Bases while Maintaining their Consistency", *The VLDB Journal*, Vol. 4, No. 2, pp. 193-241, 1995.

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports – 1996

- LSI-96-1-R “(Pure) Logic out of Probability”, Ton Sales.
- LSI-96-2-R “Automatic Generation of Multiresolution Boundary Representations”, C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé.
- LSI-96-3-R “A Frame-Dependent Oracle for Linear Hierarchical Radiosity: A Step towards Frame-to-Frame Coherent Radiosity”, Ignacio Martin, Dani Tost, and Xavier Pueyo.
- LSI-96-4-R “Skip-Trees, an Alternative Data Structure to Skip-Lists in a Concurrent Approach”, Xavier Messeguer.
- LSI-96-5-R “Change of Belief in SKL Model Frames (Automatization Based on Analytic Tableaux)”, Matías Alvarado and Gustavo Núñez.
- LSI-96-6-R “Compressibility of Infinite Binary Sequences”, José L. Balcázar, Ricard Gavaldà, and Montserrat Hermo.
- LSI-96-7-R “A Proposal for Word Sense Disambiguation using Conceptual Distance”, Eneko Agirre and German Rigau.
- LSI-96-8-R “Word Sense Disambiguation Using Conceptual Density”, Eneko Agirre and German Rigau.
- LSI-96-9-R “Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees”, Lluís Màrquez and Horacio Rodríguez.
- LSI-96-10-R “POS Tagging Using Relaxation Labelling”, Lluís Padró.
- LSI-96-11-R “Hybrid Techniques for Training HMM Part-of-Speech Taggers”, Ted Briscoe, Greg Grefenstette, Lluís Padró, and Iskander Serail.
- LSI-96-12-R “Using Bidirectional Chart Parsing for Corpus Analysis”, A. Ageno and H. Rodríguez.
- LSI-96-13-R “Limited Logical Belief Analysis”, Antonio Moreno.
- LSI-96-14-R “Logic as General Rationality: A Survey”, Ton Sales.
- LSI-96-15-R “A Syntactic Characterization of Bounded-Rank Decision Trees in Terms of Decision Lists”, Nicola Galesi.
- LSI-96-16-R “Algebraic Transformation of Unary Partial Algebras I: Double-Pushout Approach”, P. Burmeister, F. Rosselló, J. Torreus, and G. Valiente.

- LSI-96-17-R “Rewriting in Categories of Spans”, Miquel Monserrat, Francesc Rosselló, Joan Torrens, and Gabriel Valiente.
- LSI-96-18-R “On the Depth of Randomly Generated Circuits”, Tatsuie Tsukiji and Fatos Xhafa.
- LSI-96-19-R “Learning Causal Networks from Data”, Ramon Sangüesa i Solé.
- LSI-96-20-R “Boundary Generation from Voxel-based Volume Representations”, R. Joan-Arinyo and J. Solé.
- LSI-96-21-R “Exact Learning of Subclasses of CDNF Formulas with Membership Queries”, Carlos Domingo.
- LSI-96-22-R “Modeling the Thermal Behavior of Biosphere 2 in a Non-Controlled Environment Using Bond Graphs”, Angela Nebot, François E. Cellier, and Francisco Mugica.
- LSI-96-23-R “Obtaining Synchronization-Free Code with Maximum Parallelism”, Ricard Gavaldà, Eduard Ayguadé, and Jordi Torres.
- LSI-96-24-R “Memoisation of Categorical Proof Nets: Parallelism in Categorical Processing”, Glyn Morrill.
- LSI-96-25-R “Decision Trees Have Approximate Fingerprints”, Víctor Lavín and Vijay Raghavan.
- LSI-96-26-R “Visible Semantics: An Algebraic Semantics for Automatic Verification of Algorithms”, Vicent-Ramon Palasí Lallana.
- LSI-96-27-R “Massively Parallel and Distributed Dictionaries on AVL and Brother Trees”, Joaquim Gabarró and Xavier Messeguer.
- LSI-96-28-R “A Maple package for semidefinite programming”, Fatos Xhafa and Gonzalo Navarro.
- LSI-96-29-R “Bounding the expected length of longest common subsequences and forests”, Ricardo A. Baeza-Yates, Ricard Gavaldà, and Gonzalo Navarro.
- LSI-96-30-R “Parallel Computation: Models and Complexity Issues”, Raymond Greenlaw and H. James Hoover.
- LSI-96-31-R “ParaDict, a Data Parallel Library for Dictionaries (Extended Abstract)”, Joaquim Gabarró and Jordi Petit i Silvestre.
- LSI-96-32-R “Neural Networks as Pattern Recognition Systems”, Lourdes Calderón.
- LSI-96-33-R “Semàntica externa: una variant interessant de la semàntica de comportament” (written in Catalan), Vicent-Ramon Palasí Lallana.
- LSI-96-34-R “Automatic verification of programs: algorithm ALICE”, V.R. Palasí Lallana.
- LSI-96-35-R “Multiresolution Approximation of Polyhedral Solids”. D. Ayala, P. Brunet, R. Joan-Arinyo, I. Navazo.
- LSI-96-36-R “Algebraic Transformation of Unary Partial Algebras II: Single-Pushout Approach”, P. Burmeister, M. Monserrat, F. Rosselló, and G. Valiente.

- LSI-96-37-R "Probabilistic Conditional Independence: A Similarity-Based Measure and its Application to Causal Network Learning", Ramon Sangüesa Solé, Joan Cabós Fabregat, and Ulises Cortés García.
- LSI-96-38-R "Analysing the Process of Enforcing Integrity Constraints", Enric Mayol and Ernest Teniente.
- LSI-96-39-R "Reducció de l'equivalència inicial visible a teoremes inductius" (written in Catalan), Vicent-Ramon Palasí Lallana.
- LSI-96-40-R "A Compendium of Problems Complete for Symmetric Logarithmic Space", Carme Àlvarez and Raymond Greenlaw.
- LSI-96-41-R "Semàntica algebraica del llenguatge AI: l'algorisme α " (written in Catalan), V.R. Palasí Lallana.
- LSI-96-42-R "Partial Occam's Razor and its Applications", Carlos Domingo, Tatsuie Tsujiki, and Osamu Watanabe.
- LSI-96-43-R "Transparent Distributed Problem Resolution in the MAKILA Multi-Agent System", Karmelo Urzelai.
- LSI-96-44-R "The Intensional Events Method for Consistent View Updating". Dolors Costal, Ernest Teniente, and Toni Urpí.
- LSI-96-45-R "Extending Eiffel as a Full Life-cycle Language", Alonso J. Peralta and Joan Serras.
- LSI-96-46-R "Analysis of Methods for Generating Octree Models of Objects from Their Silhouettes", Marta Franquesa and Pere Brunet.
- LSI-96-47-R "Learning nearly monotone k -term DNF". Jorge Castro, David Guijarro, and Víctor Lavín.
- LSI-96-48-R "Learning Monotone Term Decision Lists". David Guijarro, Víctor Lavín, and Vijay Raghavan.
- LSI-96-49-R "Coding Complexity: The Computational Complexity of Succinct Descriptions", José L. Balcázar, Ricard Gavaldà, and Osamu Watanabe.
- LSI-96-50-R "Algorithms for Learning Finite Automata from Queries: A Unified View", José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe.
- LSI-96-51-R "Mètodes de validació d'esquemes de bases de dades deductives". Carles Farré.

Hardcopies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
`secrelsi@lsi.upc.es`

See also the Department WWW pages, <http://www-lsi.upc.es/www/>