Comprovació de Restriccions d'Integritat i Modificació de Vistes en Bases de Dades Deductives

Ernest Teniente

Report LSI–91–3

ABSTRACT

This work describes the main concepts of deductive databases and studies some of the problems (integrity constraint satisfaction and view updating) that they have. Before studying some solutions proposed to solve these problems, the concepts of internal event and internal events rules are defined.

Two possible solutions to each problem are analyzed, for integrity checking the Consistency Method and the Internal Events Method, and for view updating Decker's Method and Internal Events Method.

ABSTRACT

Aquest treball descriu els conceptes bàsics de les bases de dades deductives i estudia algun dels problemes que aquestes tenen (comprovació de les restriccions d'integritat i modificació de vistes). Abans d'estudiar algunes solucions proposades, es defineixen els conceptes d'esdeveniment intern i de regla d'esdeveniments interns.

S'analitzen dues solucions proposades per a cada problema, en comprovació de les restriccions d'integritat el Consistency Method i el Mètode dels Esdeveniments Interns, i per a modificació de vistes: el mètode proposat per Hendrik Decker i el Mètode dels Esdeveniments interns.

INDEX

1. INTRODUCCIO

Les bases de dades deductives generalitzen les relacionals incloent no només fets i restriccions d'integritat, sinó també regles deductives. Mitjançant aquestes regles es poden derivar nous fets a partir dels fets emmagatzemats explícitament.

En aquest treball es dóna una idea general del concepte de base de dades deductiva i s'estudien dos dels problemes principals que aquestes plantegen, concretament el de la comprovació de les restriccions d'integritat i el de modificació de vistes, així com alguns mètodes que proposen solucions als problemes esmentats. Un dels problemes importants no tractat en aquest treball és el del processament de consultes sobre una base de dades deductiva, una referència general es pot trobar a [5] on s'esmenten els enfocaments bottom-up i top-down.

Les restriccions d'integritat són condicions que la base de dades ha de satisfer en qualsevol instant de temps. Així doncs, els mètodes que tracten aquest problema proporcionen procediments que comproven si realment es compleixen o no aquestes condicions.

El problema de la modificació de vistes en bases de dades deductives és similar al problema del mateix nom en bases de dades relacionals. El seu objectiu és proporcionar procediments efectius per traduir les modificacions de vistes en modificacions correctes sobre la base de dades.

Aquest treball està organitzat de la següent manera. A la propera secció es defineixen els conceptes bàsics de bases de dades deductives [3,4,5,6,9]. La secció 3, defineix els conceptes d'esdeveniment intern, i de regles de transició i regles d'esdeveniments interns [7,8], aquests conceptes són claus per a dos dels mètodes estudiats posteriorment. La secció 4 tracta el problema de comprovació de restriccions d'integritat, explicant-se dos mètodes: el Consistency Method [1, 10], i el mètode dels esdeveniments interns [8]. Finalment, a la secció 5 es parla del problema de modificació de vistes i concretament es desenvolupen el mètode d'en Decker [2] i el mètode dels esdeveniments interns [11].

2. CONCEPTES BÀSICS

Les bases de dades deductives són resultat de l'aplicació de la lògica al camp de les bases de dades. La lògica té l'avantatge de poder ser utilitzada com a sistema d'inferència a més de com a llenguatge de representació.

2.1 Consideracions prèvies

Abans de considerar la formalització de les bases de dades deductives, cal esmentar algunes hipòtesis que governen l'avaluació de consultes (i restriccions d'integritat) a les bases de dades:

1- Hipòtesi del món tancat: també anomenada convenció per informació negativa, assumeix que els fets que no es coneixen com a certs són falsos (p. ex. $\neg R(e_1,\ldots, e_n)$ se suposa que és cert ssi el tuple $<e_1,\ldots, e_n>$ no pertany a la relació R).

2- Hipòtesi del nom únic: els individus que tenen noms diferents són diferents.

3- Hipòtesi de tancament del domini: no existeixen més individus que els de la base de dades.

Aquestes hipòtesis expressen una certa representació implícita dels fets negatius i precisen l'univers de referència al qual es refereixen les consultes.

Una base de dades vista des de la lògica es pot considerar des de dos punts de vista diferents, ja sigui com una *interpretació* (d'una teoria de primer ordre) o bé com una *teoria* (de primer ordre). Quan es considera des del punt de vista de les interpretacions, les consultes (i les restriccions d'integritat) són fórmules que han de ser avaluades en la interpretació usant la definició semàntica de certesa. Des del punt de vista de la teoria, considerem les consultes i les restriccions com teoremes que han de ser provats. Els punts de vista de la interpretació i de la teoria formalitzen els conceptes de base de dades convencional i deductiva respectivament.

Reiter [9] i Kowalski [6] han investigat també aquests enfocs. Reiter els anomena *model-theoretic view* i *proof-theoretic view* respectivament, mentre que Kowalski els anomena *relational structure view* i *logic database view.*

Tant Kowalski com Reiter han demostrat que, encara que les bases de dades convencionals es consideren generalment des del model-theoretic view, es poden considerar també des del proof-theoretic view i per tant ser considerades com un cas particular de les bases de dades deductives.

2.2 Bases de dades deductives

Una primera classificació de les bases de dades [4] distingeix entre les bases de dades deductives definides i les indefinides. Les bases de dades definides estan formades per clàusules definides del tipus $A \leftarrow B_1 \wedge \ldots \wedge B_n$, on tant A com els B_i són àtoms. En canvi, les bases de dades indefinides admeten també sentències del tipus $A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$, per tant, tenen tot el poder expressiu de la lògica de primer ordre. Aquestes bases de dades tracten amb informació incompleta, ja que si es compleixen totes les condicions $B_1, \ldots, B_n$ sabem que algun dels àtoms $A_1, \ldots, A_m$ és cert, però no sabem quants ni quins d'aquests A_i són certs.

En aquest treball ens centrarem en bases de dades deductives definides augmentades amb negació, ja que són les considerades pels mètodes estudiats. El fet de no tractar les bases de dades indefinides fa que no puguem tractar amb informació incompleta. Malgrat tot, les clàusules definides tenen l'avantatge de que proporcionen un gran poder expressiu amb relativa simplicitat i que els sistemes que les usen poden ser implementats més eficientment.

Una base de dades deductiva definida augmentada amb negació és un conjunt finit de regles deductives, que són fórmules tancades del tipus:

$$A \leftarrow L_1 \wedge \ldots \wedge L_n, \quad \text{amb } n \geq 0$$

On A és un àtom, els L_i són literals (poden ser tant àtoms com àtoms negats), i se suposa que totes les variable estan quantificades universalment davant la fórmula en que ocorren. A rep el nom de *conclusió* (o *cap*) de la regla i els L_i de *condicions* (o *cos*).

En el cas n=0, la regla deductiva rep el nom de *fet*, i s'acostuma a representar $A(x_1, \ldots, x_n)$.

Un cop vista la definició de regla deductiva, podem refinar una mica més el concepte de base de dades deductiva. Direm doncs que una base de dades deductiva consisteix en tres conjunts finits de clàusules: un conjunt F de fets, un conjunt R de regles deductives i un conjunt I de restriccions d'integritat. Una base de dades relacional és una base de dades deductiva sense regles deductives. El conjunt de fets rep el nom de base de dades *extensional*, mentre que el conjunt de regles deductives s'anomena base de dades *intensional*.

Suposarem que els predicats de la base de dades poden ser només bàsics o derivats. Un predicat *bàsic* apareix només a la base de dades extensional i (eventualment) en el cos de les regles deductives. Un predicat *derivat* apareix només a la base de dades intensional. Qualsevol base de dades es pot definir d'aquesta manera.

Normalment es requereix que la base de dades abans i després de les modificacions sigui *allowed* (que equival al concepte de range-restricted en bases de dades deductives definides), o sigui, que cada variable que ocorre en una regla deductiva tingui una ocorrència en una condició positiva de la regla. Això assegura que totes les condicions negatives siguin plenament instanciades abans de que siguin avaluades per la regla de "negació com a fracàs finit".

2.3 Restriccions d'integritat

Les restriccions d'integritat, de la mateixa manera que les regles deductives, corresponen al coneixement general del món modelat per la base de dades, i més concretament es corresponen amb condicions que ha de satisfer la base de dades en qualsevol instant de temps. Si una modificació de la base de dades viola alguna d'aquestes restriccions haurà de ser refusada.

Alguns exemples de restriccions d'integritat són:

EXEMPLE 1

Una restricció bastant usual pot expressar una dependència funcional. Per exemple, "un empleat pot tenir com a màxim un sou", això es pot expressar:

$$\forall x \forall y \forall z \, [y=z \leftarrow \text{Empleat}(x) \wedge \text{Sou}(x,y) \wedge \text{Sou}(x,z)]$$

EXEMPLE 2

"Cap estudiant pot tenir alhora una beca de la CIRIT i una beca ERASMUS"

$$\forall x [\leftarrow \text{Estudiant}(x) \wedge \text{té}(x, \text{beca\_ERASMUS}) \wedge \text{té}(x, \text{beca\_CIRIT})]$$

Ambdós mètodes de comprovació de restriccions d'integritat estudiats en aquest article tracten amb restriccions que tenen forma de *denegació*. Una denegació funciona com una sentència objectiu de condicions que no s'han de produir a la base de dades. Es representen:

$$\leftarrow L_1 \wedge \ldots \wedge L_n, \quad \text{amb } n \geq 1$$

On els L_i són literals i totes les variables se suposa que estan universalment quantificades davant la restricció en la que ocorren. Com s'esmenta a [8 i 10] les restriccions d'integritat de tipus més general es poden transformar en denegacions. La restricció de l'EXEMPLE2 anterior està en forma de denegació.

Les denegacions han de ser també *allowed*, o sigui, qualsevol variable que ocorre en una condició negativa de la regla ha de tenir una ocurrència en una condició positiva de la restricció.

Les restriccions explicades fins ara reben el nom de restriccions *estàtiques*, ja que han de ser satisfetes en qualsevol estat de la base de dades. Les restriccions que inclouen dos o més estats de la base de dades, s'anomenen restriccions d'integritat de *transició* (o dinàmiques).

2.4 Resolució SLDNF

El concepte de procediment de resolució SLDNF és molt important en l'àmbit de les bases de dades deductives, per aquest motiu creiem necessari dedicar un subapartat d'aquest treball a tractar d'especificar-lo formalment. Ens basarem en la descripció donada per en Kowalski al seu article [10]

El mètode de resolució es pot considerar com un intent de mecanitzar i fer sistemàtica la deducció natural. L'objectiu d'aquest mètode consisteix en fer demostracions per reducció a l'absurd (d'on es dedueix que és un mètode refutatiu) usant una única regla: la de resolució. Esquemàticament:

$$A_1, \ldots, A_n \vdash B \quad \Leftrightarrow \quad A_1, \ldots, A_n, \neg B \vdash []$$

Una *regla de càlcul* és una regla que selecciona un literal d'una denegació. Una regla de càlcul és *segura* si i només si no selecciona condicions negatives que no són de la base (p. ex. que no contenen variables). Entenem per "de la base" la traducció del terme anglès "ground".

Sigui S un conjunt de regles deductives, G una denegació, i R una regla de càlcul segura. Una *derivació-SLDNF* de $S \cup \{G\}$, via R és una seqüència (que pot ser infinita), $G_0, G_1, G_2, \ldots$ tal que l'objectiu inicial $G_0 = G$, i per tota i, $i \geq 0$, G_{i+1} s'obté de G_i per una de les regles següents:

1. Sigui G_i "$\leftarrow L_1 \wedge \ldots \wedge L_n$", i suposem que R selecciona una condició positiva de G_i. Aleshores G_{i+1} és el resolvent en L_k de G_i i alguna clàusula d'entrada en S.

   Quan parlem de "resolvent" ens estem referint a la generalització de la noció estàndard de resolvent: Sigui C una regla deductiva

   $$A \leftarrow L'_1 \wedge \ldots \wedge L'_m$$

   del conjunt d'entrada S tal que A i L_k unifiquen amb l'unificador més general (mgu) θ.

   Aleshores per "resolvent" de G_i i C en L_k ens referim a la fórmula:

   $$\leftarrow (L_1 \wedge \ldots \wedge L_{k-1} \wedge L_{k+1} \wedge \ldots \wedge L_n \wedge L'_1 \wedge \ldots \wedge L'_m)\theta.$$

2. Sigui G_i "$\leftarrow L_1 \wedge \ldots \wedge L_n$", i suposem que R selecciona de G_i un literal L_k, que és un àtom negat "NOT A". Aleshores G_{i+1} és G_i amb L_k eliminat si l'objectiu "$\leftarrow$NOT A" té èxit. "$\leftarrow$NOT A" *té èxit* si l'objectiu "$\leftarrow$ A" fracassa finitament, p.ex. si per alguna regla de càlcul segura R' l'espai de cerca SLDNF per $S \cup \{A\}$ fracassa finitament.

Una *refutació SLDNF* de $S \cup \{G\}$ via R és una derivació SLDNF de $S \cup \{G\}$ via R que acaba a la clàusula buida.

Un objectiu G *té èxit* amb un conjunt d'entrada S si i només si és una refutació SLDNF de $S \cup \{G\}$. Si un objectiu "$\leftarrow$ W" té èxit amb el conjunt d'entrada S, aleshores "$\exists x_1, \ldots, \exists x_n$ W" és una conseqüència lògica de Comp(S), on $x_1, \ldots, x_n$ són totes les variables lliures que ocorren a W.

Una *derivació SLDNF fracassada finitament* és una derivació SLDNF $G_0, G_1, \ldots, G_n$, $n \geq 0$, tal que el literal seleccionat de G_n és una condició negativa "NOT A" i "$\leftarrow$ A" té èxit, o bé el literal seleccionat és una condició positiva i G_n no té cap resolvent en aquest literal amb cap de les regles deductives del conjunt d'entrada.

Un *espai de cerca SLDNF* per $S \cup \{G_0\}$ via R és el conjunt de totes les derivacions SLDNF per $S \cup \{G_0\}$ via R tal que qualsevol derivació finita en el conjunt és una refutació o bé una derivació fracassada finitament.

S'ha comprovat que la resolució SLDNF és sòlida per qualsevol tipus de regles deductives i objectius i completa per certs casos restringits.

2.5 Condicions relatives a la completesa de la resolució SLDNF

A l'apartat anterior hem esmentat que la resolució SLDNF és completa només en casos determinats. Sembla per tant interessant veure les propietats sintàctiques que han de tenir les bases de dades deductives per tal que la resolució SLDNF sigui completa. Seguirem la terminologia i definicions donades a [3].

Moltes de les propietats que donarem estan basades en la manera en que les fórmules depenen unes de les altres. Les propietats es defineixen en termes del *graf de dependències*. Els nodes del graf de dependències són els fets i regles de la base de dades (D), i per cada parell de nodes F, F', hi ha una fletxa de F' a F si hi ha un àtom A en el cos de F tal que els predicats en A i en la conclusió de F' són els mateixos predicats bàsics. La fletxa es marcarà positiva (resp. negativa) si A és positiu (resp. negatiu) en F.

Si F i F' són dos nodes del graf de dependències, direm que:

a) F *depèn en* F' si hi ha un camí de F' a F.
b) F *depèn positivament* (*resp. negativament*) en F' si hi ha un camí de F' a F que no conté cap fletxa negativa (resp. com a mínim una fletxa negativa).
c) F *depèn en forma parell* (resp. *en forma senar*) en F' si hi ha un camí de F' a F que conté un nombre parell (resp. senar) de fletxes negatives.
d) F *depèn recursivament en ell mateix* si hi ha un camí de F a F de longitud superior a 0.

Aquestes definicions s'usen per caracteritzar les següents propietats d'una base de dades D.

a) D és *jeràrquica* si cap node en el graf de dependències de D depèn recursivament d'ell mateix.
b) D és *estratificada* si cap node en el graf de dependències de D depèn negativament d'ell mateix.
c) D és *call-consistent* si cap node en el graf de dependències de D depèn en forma senar d'ell mateix.
d) D és *estricta* si no hi ha cap parell F,F' de nodes en el graf de dependències de D tal que F depèn de forma parella i senar en F'.
e) D és *even* si no hi ha cap parell F,F' de nodes en el graf de dependències de D tal que F depèn de forma parella i senar en F' i F' depèn recursivament en ell mateix.

La resolució SLDNF és incompleta en general, però hi ha classes de programes lògics (bases de dades) i objectius pels quals si que és completa. La resolució SLDNF és completa en els següents programes lògics:

- Bases de dades que són jeràrquiques i allowed
- Bases de dades que són allowed, estrictes i estratificades
- Bases de dades que són allowed, estrictes i call-consistent
- Bases de dades que són call-consistent, even i recursivament recobertes (una generalització d'allowed)

3. BASE DE DADES AUGMENTADA

Dos dels mètodes estudiats en aquest article [8,11] augmenten la base de dades D amb els conceptes de regles de transició i de regles d'esdeveniments interns, parlant aleshores de base de dades augmentada, que es denota per A(D). Aquesta consisteix en la base de dades D, les seves regles de transició i les seves regles d'esdeveniments interns.

En aquesta secció es descriuen els conceptes d'esdeveniment intern, regla de transició i regla d'esdeveniments interns. En seccions posteriors ja es parlarà de quin és el paper que juga la base de dades augmentada en aplicar-se els mètodes esmentats als problemes de comprovació de les restriccions d'integritat i de modificació de vistes en bases de dades deductives.

3.1 Esdeveniments interns [7,8]

Degut a les regles de deducció, l'aplicació d'una modificació (p. ex. inserció o esborrat de fets bàsics) a una base de dades deductiva pot provocar que hi hagi altres modificacions induïdes sobre alguns predicats derivats. Els esdeveniments interns permeten explicitar les modificacions induïdes sobre aquests predicats.

O sigui, suposem que tenim un fet derivat P(K), on K és un vector de constants, que és cert en una base de dades D, i s'aplica una modificació qualsevol U sobre aquesta base de dades. Diem que U indueix una transició entre D i la base de dades modificada D'. Si el fet P'(K) (es denota per P'(K) el mateix predicat P, avaluat en la nova base de dades modificada) no és cert en D', voldrà dir que s'ha produït un esdeveniment intern d'esborrat, i es denotarà per $\delta P(K)$.

De la mateixa manera, si el fet P(K) no era cert a la base de dades original, i després d'aplicar la modificació sí que és cert a la nova base de dades, voldrà dir que s'ha produït un esdeveniment intern d'inserció i l'escriurem $\iota P(K)$.

Si P és un predicat bàsic, aleshores els fets ιP i δP representen, respectivament, insercions i esborrats de fets bàsics.

Lògicament, un esdeveniment intern d'inserció requereix que el fet no existeix abans de la modificació, mentre que l'esdeveniment intern d'esborrat requereix que prèviament a la modificació el fet sigui cert.

$$\forall x\ (\iota P(x) \rightarrow \neg P(x))$$
$$\forall x\ (\delta P(x) \rightarrow P(x))$$

Per motius d'uniformitat, s'associa a cada restricció d'integritat un predicat d'inconsistència Icn, amb o sense termes, i d'aquesta manera té la mateixa forma que les regles deductives. Els anomenarem *regles d'integritat.* Per exemple, si tenim una restricció d'integritat en forma de denegació com ara $\leftarrow$ P(K), la reescriurem Ic1 $\leftarrow$ P(K), i així té la mateixa forma que una regla deductiva.Si P és un predicat d'inconsistència, aleshores els fets ιP que ocorren durant la transició es correspondran a violacions de les seves restriccions d'integritat.

3.2 Regles de transició

Una regla de transició defineix el predicat P' (predicat P en el futur estat de la base de dades) en termes dels predicats de l'estat actual i dels esdeveniments interns que poden ocórrer a la transició. Les regles de transició només es defineixen pels predicats derivats.

A partir d'aquestes regles de transició es dedueixen les regles d'esdeveniments interns. A l'article [8] s'explica el procediment que cal seguir per obtenir les regles de transició.

3.3 Regles d'esdeveniments interns

Cal distingir entre les regles d'esdeveniments interns d'*inserció* i les d'*esborrat* . Les regles d'esdeveniments interns d'inserció permeten deduir quins fets ιP (o insercions induïdes) es produeixen en una transició. Altrament, les regles d'esdeveniments interns d'esborrat permeten deduir quins fets δP (o esborrats induïts) es produeixen en una transició.

Com que les regles de transició només es defineixen pels predicats derivats i les regles d'esdeveniments interns (tan d'inserció com d'esborrat) es dedueixen a partir de les regles de transició, només hi haurà regles d'esdeveniments interns pels predicats derivats.

El procediment que se segueix per deduir aquestes regles a partir de les regles de transició així com algunes condicions de simplificació es poden trobar a [8].

4. COMPROVACIO DE LES RESTRICCIONS D'INTEGRITAT

En aquesta secció anem a estudiar dos mètodes que tracten de resoldre el problema de determinar si una base de dades deductiva satisfà les seves restriccions d'integritat, i més concretament, si una base de dades deductiva que satisfà les seves restriccions d'integritat abans d'una transacció, les continua satisfent a posteriori.

Ambdós mètodes suposen que la base de dades satisfà les restriccions d'integritat abans de la transacció, d'aquesta manera eviten comprovació redundant d'instàncies de restriccions d'integritat que no es veuen afectades per la transacció.

4.1 The Consistency Method [10]

Per tal d'explotar la hipòtesi que la base de dades satisfà les restriccions d'integritat abans de la transacció, el Consistency Method raona cap endavant a partir de les modificacions.

En el seu article, Sadri i Kowalski proposen una extensió de la resolució SLDNF per a fer la comprovació de les restriccions d'integritat en bases de dades deductives. Necessiten extendre la resolució SLDNF per tal de poder:

- usar com a clàusula inicial qualsevol regla deductiva arbitrària, denegació o fet negat.
- incorporar regles d'inferència per raonar sobre els esborrats implícits resultants d'altres insercions o esborrats.
- permetre un pas de resolució extès per raonar cap endavant a partir dels fets negats.

Usen clàusules corresponents a les modificacions com a clàusules inicials per a l'espai de cerca. Això els permet aprofitar la hipòtesi de que la base de dades satisfà les restriccions d'integritat abans de la transacció, i per tant, qualsevol violació de les restriccions d'integritat ha d'incloure com a mínim una de les modificacions de la transacció.

Veiem quina és la clàusula inicial escollida en funció del tipus de modificació:

Tipus de modificació	Clàusula inicial
Inserció d'una regla deductiva R	Regla deductiva R
Esborrat d'un fet A	NOT A
Insercions de restricció d'integritat IC	Restricció d'integritat IC

Com que els fets són casos particulars de regles deductives la inserció d'un fet F es tractarà com el primer cas, o sigui la clàusula inicial que s'escollirà serà F.

En el cas que la modificació sigui un esborrat d'una regla deductiva (que no sigui un fet), caldria determinar quines instàncies de la conclusió d'aquesta regla s'esborren com a resultat de la supressió i seleccionar com a clàusules inicials la negació d'aquestes instàncies esborrades.

Si es vol esborrar un predicat derivat (també anomenats vistes), caldrà modificar (inserir o esborrar) alguns dels fets bàsics de la base de dades. Malgrat tot, diferents conjunts de clàusules poden aconseguir el mateix efecte, i caldria triar quina és la modificació a aplicar. Aquest és concretament el problema de modificació de vistes que es tractarà a la propera secció, i per tant no tractarem ara la seva vessant de comprovació de restriccions d'integritat.

L'esborrat d'una restricció d'integritat no pot causar una inconsistència i, per tant, no hi ha cap necessitat per comprovar la integritat en una modificació que esborra una restricció.

En el Consistency Method el conjunt d'entrada sobre el que s'aplicarà el procediment de prova (com ja hem esmentat abans diferent de la resolució SLDNF) està composat per les restriccions d'integritat i la base de dades modificada.

El Consistency Method consisteix en aplicar el seu procediment de prova, començant per la clàusula inicial que es correspon en funció de la modificació que es fa. Si s'arriba a la clàusula buida, voldrà dir que es viola alguna restricció d'integritat, i per tant caldrà refusar o variar la modificació. Al contrari, si l'espai de cerca fracassa finitament vol dir que no es produeix cap violació de les restriccions d'integritat i per tant es pot aplicar la modificació desitjada.

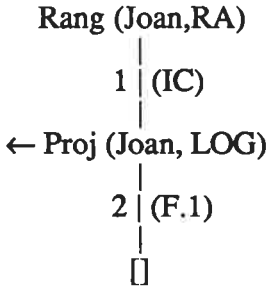
Exemple1: Suposem que tenim la següent base de dades, restriccions i transacció:

F.1 Proj (Joan, LOG)
F.2 Proj (Maria, LOG)
IC $\leftarrow$ Rang (x, RA) $\wedge$ Proj (x, LOG)

La restricció imposa que cap dels RA (assistents en recerca) pot treballar al projecte LOG.

T: Afegir Rang (Joan, RA).

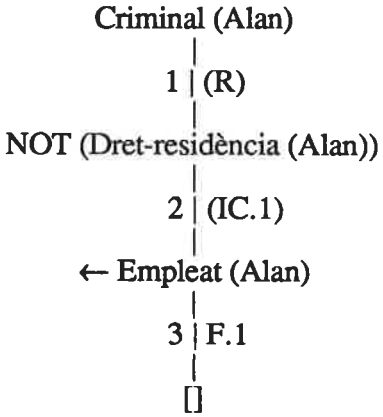
Com es pot deduir directament, com que en Joan treballa al projecte LOG, la inserció d'un fet que ens diu que el rang d'en Joan és RA farà que es violi la restricció d'integritat. Anem a veure com ho fa el Consistency Method per detectar que s'ha produït aquesta violació.



Exemple2: Suposem que tenim la següent base de dades, restriccions i transacció:

F.1 Empleat (Alan)
F.2 Estranger (Alan)
R.1 Dret-residència (x) ← Estranger (x) ∧ NOT (Criminal (x))
IC.1 ← Empleat (x) ∧ NOT (Dret-residència (x))
Tx: Afegir Criminal (Alan)

El següent espai de cerca il.lustra la violació de IC.1 a la base de dades modificada



El primer pas d'aquesta refutació usa una regla d'inferència (R) que ens permet raonar de la següent manera:

(R) **ja que** a la base de dades modificada
Criminal (Alan) és cert i tenim que
Dret-residència (x) ← Estranger (x) ∧ NOT (Criminal (x))
i no hi ha cap altra manera de provar Dret-residència (Alan) i
Dret-residència (Alan) es podia provar a la base de dades
aleshores Dret-residència és esborrat
Així, NOT (Dret-residència (Alan)) és provable a la base de dades modificada

El segon pas és un pas de resolució extès entre les clàusules:

NOT (Dret-residència (Alan)) i
← Empleat (x) ∧ NOT (Dret-residència (x))

en els literals subratllats. El resolvent és ← Empleat (Alan).

El tercer és un pas de resolució SLDNF estàndard.

Transaccions amb modificacions múltiples: En general, quan una transacció consisteix en diverses modificacions, cada modificació és una clàusula inicial candidata.

Si una modificació porta a una refutació, aleshores la transacció múltiple viola alguna restricció d'integritat. Si totes les modificacions porten a espais de cerca que fracassen finitament, aleshores la transacció satisfà les restriccions.

4.2 Mètode dels esdeveniments interns

El mètode explicat a [8] utilitza directament les regles de transició i les d'esdeveniments interns per verificar si una transacció produeix inconsistències o no.

Com ja hem esmentat a l'apartat 3.1, per motius d'uniformitat s'associa a cada restricció d'integritat un predicat d'inconsistència Icn, amb o sense termes, i d'aquesta manera té la mateixa forma que les regles deductives. Els anomenarem *regles d'integritat*. Per exemple, si tenim una restricció d'integritat en forma de denegació com ara ← P(K), la reescriurem Ic1 ← P(K), i així té la mateixa forma que una regla deductiva.

El mètode d'esdeveniments interns per comprovació de restriccions d'integritat es basa completament en l'ús de la resolució SLDNF estàndard. Per cada restricció d'integritat Icj, s'agafaran com a clàusules inicials totes les regles d'esdeveniments interns d'inserció {← ιIcj (x)} associades. Una transacció T violarà la restricció d'integritat si la clàusula inicial té èxit amb el conjunt d'entrada A(D) ∪ T, i per tant hauria de ser refusada o modificada. Altrament, T pot ser acceptada i la base de dades modificada.

El mètode dels esdeveniments interns permet considerar els tipus de modificació següents:

- Inserció o esborrat de fets bàsics
- Inserció o esborrat de fets bàsics qualificats
- Inserció o esborrat de regles deductives
- Inserció o esborrat de restriccions d'integritat
- Transacció amb modificacions múltiples

Si la transacció T consisteix d'una inserció o un esborrat d'un fet bàsic, es denotarà per $T=\{\iota Q(K)\}$ o $T=\{\delta Q(K)\}$, respectivament, essent Q un predicat bàsic i K un vector de constants. Diem que $\iota Q(K)$ o $\delta Q(K)$ són els esdeveniments interns produïts per la modificació.

Inserció o esborrat de regles deductives:

Si la modificació és una inserció d'una regla deductiva del tipus:

$$P(x) \leftarrow L_1 \wedge \ldots \wedge L_n$$

degut a la inserció d'aquesta regla deductiva, la base de dades pot contenir nous fets P implícits que violin alguna de les restriccions d'integritat. El mètode dels esdeveniments interns pot detectar també aquestes violacions.

L'única cosa que cal fer és transformar el predicat derivat P(x) per obtenir-ne les seves noves regles d'esdeveniments interns d'inserció. Un cop les tenim només cal aplicar el mètode de la manera ja explicada. Si totes les restriccions d'integritat se satisfan, s'accepta la nova regla, es modifica la base de dades i es modifiquen les regles de transició i les d'esdeveniments interns.

Si la modificació esborra una regla deductiva existent, se n'obtenen les noves regles d'esdeveniments interns d'esborrat i es procedeix com abans.

Transaccions amb modificacions múltiples

El que cal fer és determinar els esdeveniments interns produïts per cada modificació i analitzar com sempre la satisfacció de les restriccions.

Exemple: Aquest exemple és el mateix que l'exemple2 que hem fet anteriorment quan explicàvem el Consistency Method, això ens permetrà comparar els dos mètodes més fàcilment. Suposem que tenim la següent base de dades, restriccions i transacció:

F.1 Empleat (Alan)

F.2 Estranger (Alan)

R.1 Dret-residència (x) $\leftarrow$ Estranger (x) $\wedge$ NOT (Criminal (x))

IC.1 Ic1 (x) $\leftarrow$ Empleat (x) $\wedge$ NOT (Dret-residència (x))

La restricció d'integritat ens diu que qualsevol empleat ha de tenir dret de residència.

Tx: Afegir Criminal (Alan)

Segons el mètode dels esdeveniments interns, la primera cosa que cal fer és obtenir les regles de transició i d'esdeveniments interns dels predicats derivats i de les restriccions d'integritat. Hem abreviat Dret-residència per Dr, Estranger per Est i Criminal per Cr.

Regles de transició:

T.1 $Dr'_{1,1}(x) \leftarrow Est(x) \wedge \neg \delta Est(x) \wedge \neg Cr(x) \wedge \neg \iota Cr(x)$
T.2 $Dr'_{1,2}(x) \leftarrow Est(x) \wedge \neg \delta Est(x) \wedge \delta Cr(x)$
T.3 $Dr'_{1,3}(x) \leftarrow \iota Est(x) \wedge \neg Cr(x) \wedge \neg \iota Cr(x)$
T.4 $Dr'_{1,4}(x) \leftarrow \iota Est(x) \wedge \delta Cr(x)$

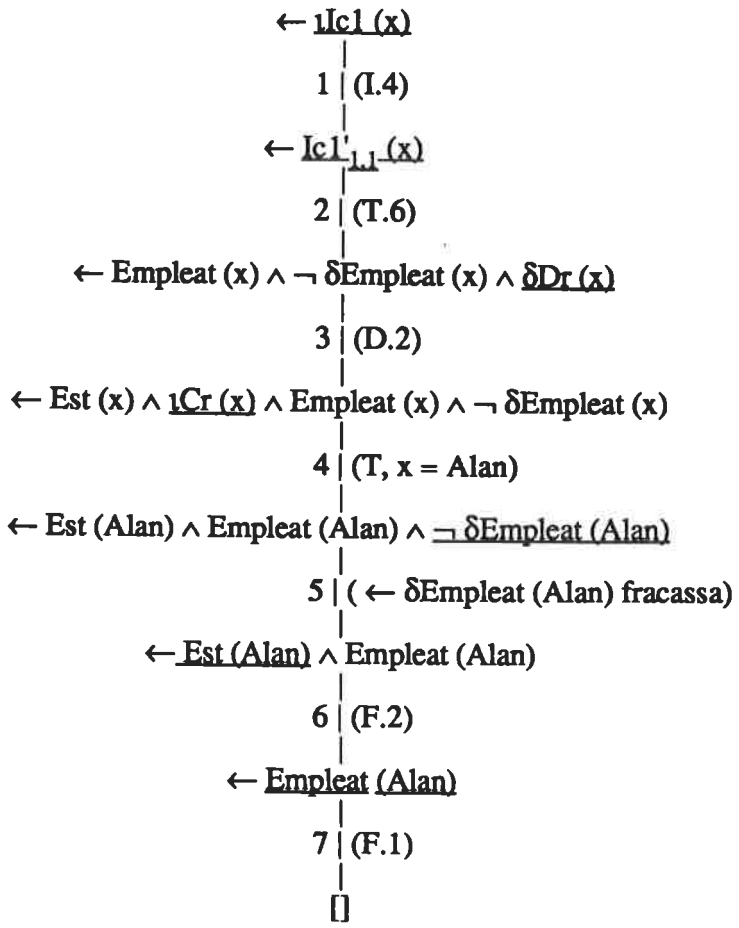
T.5 $Ic1'_{1,1}(x) \leftarrow Empleat(x) \wedge \neg \delta Empleat(x) \wedge \neg Dr(x) \wedge \neg \iota Dr(x)$
T.6 $Ic1'_{1,2}(x) \leftarrow Empleat(x) \wedge \neg \delta Empleat(x) \wedge \delta Dr(x)$
T.7 $Ic1'_{1,3}(x) \leftarrow \iota Empleat(x) \wedge \neg Dr(x) \wedge \neg \iota Dr(x)$
T.8 $Ic1'_{1,4}(x) \leftarrow \iota Empleat(x) \wedge \delta Dr(x)$

Regles d'esdeveniments interns

I1..3 $\iota Dr(x) \leftarrow Dr'_{1,j}(x) \quad j = 2...4$
I4..6 $\iota Ic1(x) \leftarrow Ic1'_{1,j}(x) \quad j = 2...4$

D.1 $\delta Dr(x) \leftarrow \neg Cr(x) \wedge \delta Est(x)$
D.2 $\delta Dr(x) \leftarrow Est(x) \wedge \iota Cr(x)$

La següent refutació mostra que la modificació T = { ιCr (Alan) } viola Ic1.



4.3 Comparació entre ambdós mètodes

Una primera diferència poc important rau en l'objectiu seleccionat com a clàusula inicial, així mentre en el Consistency Method la clàusula inicial seleccionada depèn de la modificació que s'aplica, en el mètode dels esdeveniments interns sempre se seleccionen els objectius del tipus {←ιIcj (x) }.

Una altra diferència es troba en el fet que mentre que per una transacció (ja sigui simple o múltiple) el mètode dels esdeveniments interns té tants espais de cerca com restriccions d'integritat, el Consistency Method té un espai de cerca per cada modificació dins la transacció. Això fa que l'espai de cerca en el mètode dels esdeveniments interns sigui més llarg per a transaccions simples. Malgrat això, a l'article [8] s'argumenta que si es coneixen a priori els tipus de transacció, es pot fer algun treball preparatori en temps de compilació, la qual cosa pot restringir l'espai de cerca.

Potser la diferència més important és que mentre el Consistency Method requereix una regla d'inferència (R) per raonar sobre els esborrats implícits, el mètode dels esdeveniments interns aconsegueix les mateixes capacitats de raonament gràcies a les regles d'esdeveniments interns, però mantenint-se en un marc SLDNF. Degut a aquesta regla d'inferència, la implementació del Consistency Method en Prolog requereix un meta-intèrpret i per tant és menys eficient.

Un altre avantatge important del mètode dels esdeveniments interns és que pot tractar amb les restriccions d'integritat de transició (són aquelles que inclouen dos o més estats de la base de dades), la qual cosa extén els tipus de restriccions que pot tenir una base de dades deductiva.

El cost principal del mètode dels esdeveniments interns és l'espai necessari per emmagatzemar les regles de transició i les d'esdeveniments interns. Aquest cost serà més important com més regles deductives i restriccions d'integritat tingui la base de dades. El guany principal és el temps estalviat en fer la comprovació de les restriccions d'integritat.

5. MODIFICACIó DE VISTES

La solució a aquest problema rau en proporcionar mètodes efectius per traduir les modificacions de vistes en modificacions correctes sobre la base de dades dels predicats bàsics que les defineixen. Aquest no és un problema exclusiu de les bases de dades deductives, sinó que també el trobem a les relacionals.

S'han proposat dos enfocaments a l'hora de buscar solucions al problema. El primer suggereix tractar les vistes com a *tipus abstractes de dades* on la definició de la vista inclou totes les modificacions de vista possibles i la seva traducció. El segon enfocament consisteix en definir un procediment de traducció general (un *traductor*). Les entrades al traductor són la definició de vista, la modificació de la vista i la base de dades actual, mentre que la sortida és la modificació de la base de dades que tradueix la modificació de vista. Ambdós mètodes estudiats en aquesta secció segueixen l'enfocament del traductor.

5.1 Drawing updates from derivations [2]

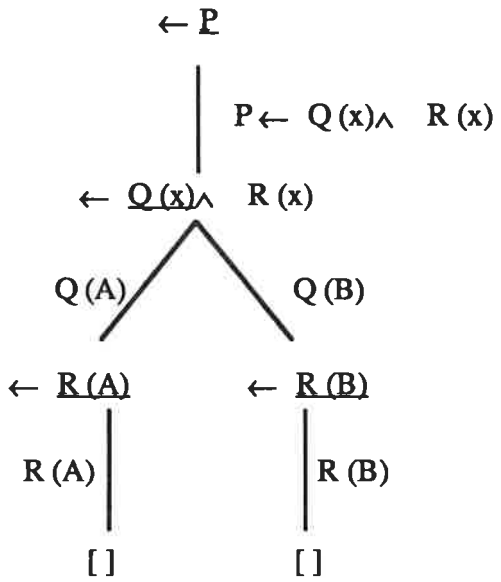
El mètode d'en Decker considera processos diferents de traducció en el cas de que la modificació sigui una inserció o bé un esborrat. Anem a estudiar com seran ambdós traductors en funció de la petició de modificació que es faci.

Modificacions en peticions d'esborrat:

Per una base de dades D i un predicat derivat P, una modificació per la petició d'esborrat P s'obté fent que cada branca que assoleix l'èxit en un arbre SLDNF de D ∪ {← P} fracassi. Això es pot aconseguir esborrant, per cada derivació que té èxit, una clàusula d'entrada en aquella derivació o bé fent una petició d'inserció d'un àtom d'un literal negatiu usat com a clàusula d'entrada. Així, aquests àtoms es tracten com peticions d'esborrat subsidiàries.

Exemple: Suposem que tenim la següent base de dades i fem una petició d'esborrat de P

Q (A).
Q (B).
R (A).
R (B).
R (C).
P ← Q (x)∧ R (x)



Qualsevol modificació possible d'esborrat de P s'obté fent que les branques que tenen èxit en la derivació SLDNF fracassin. Lògicament, pot existir més d'una solució en funció del nombre de combinacions que es puguin formar.

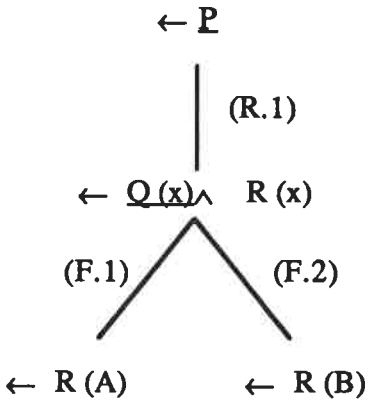
En el cas concret de l'exemple, els quatre conjunts de modificacions de vista que tradueixen la petició d'esborrat de P són: {esborrar Q (A), esborrar Q (B)}, {esborrar R (A), esborrar R (B)}, {esborrar Q (A), esborrar R (B)} i {esborrar Q (B), esborrar R (A)}.

Modificacions en peticions d'inserció:

Per una base de dades D i un predicat derivat P, una modificació per una petició d'inserció de P s'obté fent que alguna derivació fracassada de D ∪ {← P} tingui èxit. Això es pot aconseguir inserint una instància de cada literal positiu en algun objectiu G de la derivació, i fent una petició d'esborrat de l'àtom de cada literal negatiu instanciat. Així, els àtoms de literals negatius instanciats en G es tractaran com a peticions d'esborrat subsidiàries.

Exemple: Suposem que tenim la següent base de dades i fem una petició d'inserció de P.

F.1 Q (A).
F.2 Q (B).
F.3 R (C).
R.1 P ← Q (x) ∧ R (x)



En aquest cas, ens trobem dues branques que fracassen a la derivació, n'hi haurà prou amb fer que una d'elles tingui èxit per inserir el predicat P. Com que els objectius terminals de les derivacions de l'exemple ja estan instanciats, una modificació de vista possible serà {inserir R (A)} i una altra {inserir R (B)}.

Com es pot comprovar, una altra modificació de vista possible seria {inserir Q (C)}, però aquesta no es pot estirar de la derivació anterior, en aquest sentit direm que per peticions d'inserció la resolució SLDNF no és completa, i això obligarà a definir el concepte d'arbres de modificació de vistes.

Arbres de modificació de vistes:

Decker demostra que els arbres SLDNF són complets per les peticions d'esborrat, però que generalment no ho són per les peticions d'inserció, això l'obliga a definir el concepte d'*arbre de modificació de vistes* (view update tree, en anglès).

Els *arbres de modificació de vistes* tenen com a clàusula inicial una petició d'inserció. Per cada objectiu no bàsic G en un arbre de modificació de vistes, els objectius successors de G s'obtenen seleccionant un literal de G d'una manera *justa* i *segura* i resolent-lo amb cada clàusula d'entrada candidata. Per cada objectiu bàsic G en un arbre de modificació de vistes, els objectius successors de G s'obtenen resolent cada literal no instanciat en G amb cada clàusula d'entrada candidata.

Un objectiu G en una derivació és *bàsic* si:

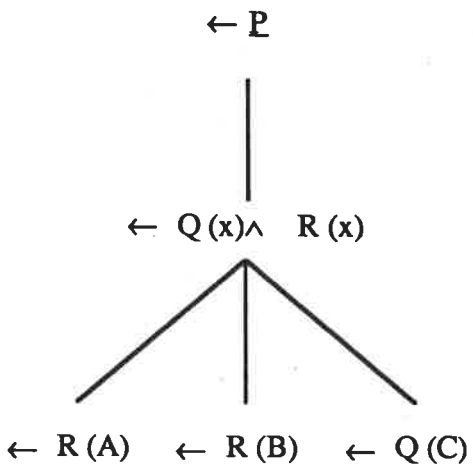
- el predicat de cada literal positiu en G és bàsic, i
- cada literal instanciat en G fracassa immediatament, i
- cada literal negatiu està instanciat.

A l'article [2] es poden trobar les definicions exactes dels conceptes de selecció de literals d'una manera justa i segura.

Exemple: anem a veure l'arbre de modificació de vistes corresponent a l'exemple que haviem fet en l'apartat de modificacions en peticions d'inserció.

F.1 Q (A).
F.2 Q (B).
F.3 R (C).

R.1 P ← Q (x)∧ R (x)



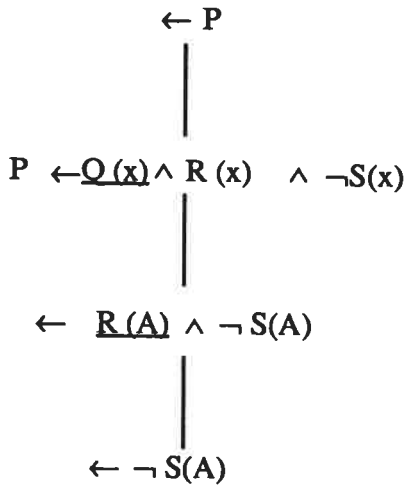
Com es pot comprovar, en aquest cas de l'arbre de modificació de vistes obtingut es poden estirar les tres modificacions possibles {inserir R(A)}, {inserir R (B)} i {inserir Q (C)}.

Comentaris sobre el mètode

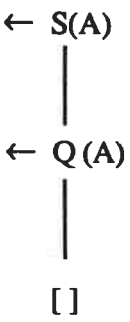
Un primer problema que trobem en el mètode d'en Decker és la invalidació de modificacions per culpa de la negació o de les restriccions d'integritat.

Exemple: Donada la següent base de dades, fem una petició d'inserció de P.

Q (A). R (A). R (B).
$P \leftarrow Q(x) \wedge R(x) \wedge \neg S(x)$
$S(A) \leftarrow Q(A)$



Arbre subsidiari:



La derivació ens mostra que P fracassa perquè S(A) té èxit, i S(A) té èxit perquè Q(A) també té èxit. En canvi, l'esborrat de Q(A) no satisfà la petició d'inserció de P. Aquest fet és el que Decker anomena invalidació de modificacions degut a la negació.

D'aquí es pot comprovar que el que aquest mètode permet estirar de les derivacions en presència de negació o de restriccions d'integritat són *possibles* modificacions. Una modificació possible serà *vàlida* si i només si satisfà la petició de modificació i les restriccions d'integritat.

Un altre problema del mètode, és que parla d'un concepte relatiu de completesa, relatiu en el sentit que poden existir modificacions de vista que no són obtingudes pel mètode. Ens sembla un problema important, ja que fa renunciar a l'objectiu desitjat d'obtenir totes les modificacions possibles. Exemples concrets es poden trobar a [2].

5.2 Mètode dels esdeveniments interns [11]

Aquest mètode per a la modificació de vistes en bases de dades deductives augmenta la base de dades amb les regles de transició i d'esdeveniments interns. Això li permet usar la resolució SLDNF estàndard per obtenir les traduccions de les modificacions de vistes.

El mètode dels esdeveniments interns està fortament relacionat amb el mètode d'en Decker [2] i més concretament amb el procediment emprat per estirar les modificacions a partir de derivacions que fracassen arrelades en una petició d'inserció. La idea és assolir un objectiu bàsic G en una derivació SLDNF que fracassa, i fer-li tenir èxit suposant que alguns esdeveniments interns en G succeeixen a la transacció. Aquests esdeveniments interns es corresponen a la modificació que cal aplicar a la base de dades.

El concepte d'objectiu bàsic no és idèntic al que dóna en Decker. Segons el mètode dels esdeveniments interns, un objectiu G en una derivació és *bàsic* si:

- El predicat de cada literal positiu en G és bàsic, i
- El predicat de cada literal negatiu en G és un esdeveniment intern o un predicat de transició, i
- Cada literal negatiu està instanciat

Un cop s'obté un objectiu bàsic G en l'arbre SLDNF de $A(D) \cup \{\leftarrow u\}$, on u és la modificació de vista, el conjunt d'insercions i esborrats de fets bàsics que cal aplicar a la base de dades, denotat per T(u), es pot obtenir directament a partir d'instanciacions apropiades dels literals que es corresponen als esdeveniments interns positius. Els literals de l'objectiu bàsic G que són esdeveniments interns bàsics negatius representen fets que no poden estar inclosos a la traducció T(u), mentre que els literals que es corresponen a esdeveniments interns derivats o predicats de transició requereixen un arbre subsidiari, del que es pot deduir també un subconjunt de T(u).

Exemple: Suposem que la base de dades conté:

F.1 Q(A)
F.2 R(A)
F.3 R(B)
DR.1 $P \leftarrow Q(x) \wedge R(x) \wedge \neg S(x)$
DR.2 $S(A) \leftarrow Q(A)$

Algunes de les regles de transició i d'esdeveniments interns corresponents a P i S són:

T.1 $P'_{1,2} \leftarrow Q(x) \wedge \neg \delta Q(x) \wedge R(x) \wedge \neg \delta R(x) \wedge \delta S(x)$
T.2 $P'_{1,3} \leftarrow \iota Q(x) \wedge R(x) \wedge \neg \delta R(x) \wedge \neg S(x) \wedge \neg \iota S(x)$
I.1...7 $\iota P \leftarrow P'_{1,j} \wedge \neg P \quad j = 2 \ldots 8$

T.3 $S'_{1,2}(A) \leftarrow \iota Q(A)$
I.8 $\iota S(A) \leftarrow S'_{1,2}(A)$

D.1 $\delta S(A) \leftarrow \delta Q(A)$

Sigui la modificació de vista la inserció de P. La següent derivació de l'arbre SLDNF per $A(D) \cup \{\leftarrow \iota p\}$ assoleix l'objectiu bàsic $\leftarrow \iota Q(B) \wedge \neg \delta R(B) \wedge \neg \iota S(B)$ i, aleshores $T(\iota P) = \{\iota Q(B)\}$. Notar que $T(\iota P)$ ha de ser tal que no inclogui $\delta R(B)$ ni $\iota S(B)$.

$\leftarrow \iota P$

1 | (I.2)

$\leftarrow P'_{1,3} \wedge \neg P$

2 | (← P fracassa)

$\leftarrow P'_{1,3}$

3 | (T.2)

$\leftarrow \iota Q(x) \wedge R(x) \wedge \neg \delta R(x) \wedge \neg S(x) \wedge \neg \iota S(x)$

4 | (F.3, x=B)

$\leftarrow \iota Q(B) \wedge \neg \delta R(B) \wedge \neg S(B) \wedge \neg \iota S(B)$

5 | (← S(B) fracassa)

$\leftarrow \iota Q(B) \wedge \neg \delta R(B) \wedge \neg \iota S(B)$

5.3 Comparació entre ambdós mètodes

La distinció que fa el mètode dels esdeveniments interns entre la base de dades i les regles de transició i d'esdeveniments interns, li permet usar la resolució SLDNF i evitar alguna de les limitacions del mètode d'en Decker. Recordar que aquest necessitava definir el concepte d'arbre de modificació de vistes en peticions d'inserció per tal de ser capaç d'estirar un conjunt de modificacions possibles.

El mètode d'en Decker pot trobar traduccions que són posteriorment invalidades per negació. Per resoldre això, ha d'aplicar les traduccions (p.ex. modificar la base de dades) i aleshores validar-les executant la petició a la base de dades modificada. Això no succeeix en el mètode dels esdeveniments interns, ja que directament a partir d'un objectiu bàsic es pot comprovar si la solució obtinguda és invalidada per negació o no.

Finalment, en alguns casos concrets, el mètode de Decker ha de ser iterat un cert nombre de vegades per obtenir una solució. El mètode dels esdeveniments interns tampoc es troba amb aquests inconvenients.

El principal cost implicat pel mètode dels esdeveniments interns és l'espai requerit per emmagatzemar les regles de transició i les d'esdeveniments interns. Aquest cost serà més important com més regles deductives i restriccions d'integritat contingui una base de dades. El guany més important és el poder addicional que s'obté.

BIBLIOGRAFIA

1. Costal, D. "Restriccions d'integritat en bases de dades deductives", Report LSI-88-27

2. Decker, H. "Drawing updates from derivations", IR-KB-65, ECRC, març 1989.

3. Decker, H.; Cavedon, L. "Generalizing allowedness while retaining completeness of SLDNF-resolution". ECRC IR-KB-52, febrer 1990.

4. Gallaire, H.; Minker, J.; Nicolas, J. "Logic and databases: A deductive approach". Computing Surveys, Vol. 16, Nº 2, juny 1984.

5. Gardarin,G.; Valduriez,P. "Deductive databases". En *Relational Databases and Knowledge Bases*. Capítol 10. Addison-Wesley, 1989.

6. Kowalski, R. "Logic as a database language". En *Proceedings of the 3rd National Conference on Databases (BNCOD3)* ,BCS Workshop series, Leeds, july 1984.

7. Olivé, A. "On the design and implementation of information systems from deductive conceptual models". En *Proceedings of the 15th. VLDB*, Amsterdam, 1989, pp. 3-11.

8. Olivé, A. "The internal events method for integrity checking in deductive databases", 1989. Per aparèixer.

9. Reiter, R. "Towards a logical reconstruction of relational database theory". En *On Conceptual modelling,* M.Brodie, J.Mylopoulos i J.W.Schmidt, Eds. Springer - Verlag, 1984.

10. Sadri, F.; Kowalski, R. "A theorem-proving approach to database integrity". In Minker,J. (Ed.) *Foundations of deductive databases and logic programming*, Morgan Kaufmann Pub., 1988, pp. 313-362.

11. Teniente, E.; Olivé, A. "The internal events method for view updating in deductive databases", 1990. Per aparèixer.