Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Semantica well-founded e vincoli di integrità

P. Asirelli, G. Cacciatore, P. Inverardi, D. Musa

Nota Interna B4-15
Giugno 1992

Semantica well-founded e vincoli di integrità*

P. Asirelli[β] , G. Cacciatore[™] , P. Inverardi[β], D. Musa[™]

1 Introduzione

E' ormai comunemente accettato che la logica, in quanto linguaggio uniforme per esprimere dati, programmi, queries e vincoli di integrità costituisce uno strumento potente e flessibile per disegnare basi di dati. In questo contesto il problema dell'integrità può essere ricondotto a quello di provare proprietà di teorie logiche. D'altra parte, la logica è anche un valido strumento di programmazione e quindi i vincoli di integrità possono essere visti come proprietà di programmi logici.

In questo lavoro presentiamo un metodo per il controllo dei vincoli di integrità in basi di dati deduttive (DDB). Tra le proposte di maggior interesse per il controllo della consistenza della base di dati rispetto ai suoi vincoli di integrità vi sono quelle di Lloyd-Topor e di Sadri-Kowalski. Queste possono essere viste come gli assi portanti di tutto l'insieme di metodi successivamente elaborati, che ne costituiscono fondamentalmente delle estensioni ottimizzate.
Nella progettazione del metodo che proponiamo, abbiamo cercato di superare le limitazioni che si possono riscontrare nelle ipotesi dei metodi suddetti, in particolare quella di dover considerare solo basi di dati stratificate. Infatti, l'obiettivo essenziale che ci siamo proposti nell'intraprendere questo lavoro è stato quello di non imporre alcun tipo di restrizione sintattica sulla natura delle basi di dati considerate. Per il raggiungimento di tale scopo ci siamo allontanati dalle semantiche "precedentemente consolidate" e abbiamo spostato la nostra attenzione sulla semantica well-founded e sulla semantica dei modelli stabili, che sembrano cogliere in modo appropriato il significato intuitivo associato ad un database generale. Queste semantiche affrontano il problema della negazione in maniera classica distaccandosi dalla visione di negazione intesa come fallimento per superarne i limiti.

Dopo aver esaminato dettagliatamente gli elementi più interessanti emergenti dalle semantiche sopra menzionate, abbiamo deciso di utilizzare la semantica well-founded come contesto per l'analisi dei vincoli di integrità. Infatti, sebbene la semantica dei modelli stabili sembri essere più espressiva nel caratterizzare l'insieme delle conseguenze logiche di un database generale, esistono diverse ragioni che testimoniano a favore della nostra scelta:

- Un database ammette sempre un unico modello well-founded, mentre può avere diversi modelli stabili la cui intersezione non è detto sia un modello, [SZ90].
- Il modello well-founded è ottenibile come minimo punto fisso di un operatore monotono definito costruttivamente; questa computazione richiede un tempo polinomiale rispetto alla dimensione dell'universo di Herbrand, se questo é finito, [VG89]. Al contrario, il problema di determinare un particolare modello stabile é NP-completo.

Diamo ora una succinta descrizione di come verrà articolato il lavoro.

Nella sezione successiva viene data tutta una serie di definizioni indispensabili per lo sviluppo del nostro metodo. La sezione 3 è dedicata alla descrizione di come ricostruire il modello well-founded di un database modificato in seguito all'applicazione di una transazione. Inoltre, in tale sezione viene messo in evidenza come la complessita della computazione del nuovo modello sia in

---

* Work partially supported by Progetto Finalizzato Sistemi Informatici e Calcolo Parallel

[β] .E.I.-C.N.R. Via S. Maria, 46-I56126 Pisa- tel.: +39-50-593 400-593477-593486;
fax: +39-50-554342; email:asirelli/inverard@iei.pi.cnr.it

[™] Dip. Informatica, Università di Pisa.

stretta correlazione con la transazione applicata. Nella sezione 4 viene descritto il metodo da noi proposto per il controllo della consistenza di un database rispetto ai suoi vincoli d'integrità. Infine, nella sezione 5 evidenziamo come il nostro metodo non abbia bisogno dell'intero modello well-founded per poter essere applicato.

2 Database Deduttivi, Vincoli d'Integrità e Semantica Well-Founded

I vantaggi della logica in ambito database sono ormai universalmente riconosciuti. La logica infatti fornisce un ambiente espressivo per la modellazione dei dati; consente l'eliminazione di un linguaggio di interrogazione separato e mette a disposizione la maggior parte dei fondamenti richiesti dai sistemi database. In ultimo, la logica incoraggia una netta separazione tra concetti dichiarativi e procedurali. Per esempio, possiamo distinguere il concetto dichiarativo di risposta corretta dal processo di valutazione usato per calcolare tale risposta. Ciò contrasta con l'approccio standard ai database relazionali, in cui il concetto dichiarativo é comunemente ignorato, oppure é identificato con l'implementazione. L'esistenza di una definizione dichiarativa fornisce un importante riferimento su cui misurare la validità dell'implementazione.

Riportiamo di seguito le definizioni riguardanti i DDB che possono essere ritrovate in [Apt88], [Prz90], [SZ90] e [SZ91].

Definizione 1: Una *clausola* (o *regola*) *database* é una formula del primo ordine del tipo A¨W, dove A é un atomo e W é una congiunzione di letterali $L_1,...,L_n$, con n≥0. A é la *testa* della clausola e W é il *corpo*. Tutte le variabili di A e tutte le variabili libere di W sono da considerarsi come quantificate universalmente davanti alla clausola. I letterali in W possono essere positivi o negativi; se W non contiene letterali negativi la clausola é detta *definita*, altrimenti é detta *generale*. ®

Definizione 2: Un *database* é un insieme finito di clausole database. Se questo consiste solo di clausole definite é detto *definito*, altrimenti é detto *generale*. ®

Definizione 3: Un *vincolo di integrità* é una formula chiusa del primo ordine. ®

Informalmente parlando, i vincoli di integrità sono condizioni che un database deve soddisfare, indipendentemente dal suo evolvere nel tempo e compiono un controllo semantico sulle relazioni del database stesso.

I vincoli di integrità possono essere usati in due modi:

a) Assumendo che il database sia inizialmente corretto, per verificare che gli aggiornamenti che operano su di esso producano una situazione finale ancora corretta.
b) Per ripristinare la consistenza del database violata in seguito ad un aggiornamento.

Nel nostro lavoro i vincoli vengono utilizzati secondo la modalità espressa nel punto a), che consente di conservare la correttezza del database, accettando soltanto quegli aggiornamenti il cui effetto soddisfa tutti i vincoli, e rigettando gli altri.

Per poter esporre in modo preciso la nozione di soddisfazione di un vincolo, introduciamo ora alcune definizioni basilari.

Sia L un linguaggio del primo ordine. Indichiamo con U e con H rispettivamente l'universo di Herbrand e la base di Herbrand relativi ad L.

Se X è un insieme di letterali ground; denotiamo con ÿX l'insieme {ÿA | AŒX} e con X^+ (risp. X^-) l'insieme di tutti i letterali positivi (risp. negativi) in X.

Definizione 4: Sia D un database generale in L. Un sottoinsieme I di H » ÿH é un'*interpretazione parziale* per D se I é consistente, cioè se I^+« ÿI^-= Δ. Inoltre, se I^+» ÿI^-= H, l'interpretazione I é detta *totale*. ®

Intuitivamente, un'interpretazione parziale I può contenere informazione incompleta: i predicati che non occorrono in I possono essere pensati come dei "fatti indefiniti".

Nel seguito parleremo genericamente di interpretazione per riferire sia un'interpretazione totale che una parziale e distingueremo tra queste ultime solo dove sarà necessario.

Definizione 5: Sia I un'interpretazione per D e r una regola in ground(D), insieme di tutte le istanze di tutte le regole in D. Se indichiamo con H(r) e B(r) rispettivamente la testa e il corpo di r, diremo che r é (a) *applicabile rispetto a I* se B(r)ÕI, (b) *applicata rispetto a I* se é applicabile e H(r)ŒI, (c) *bloccata rispetto a I* se esiste un letterale A in B(r) tale che ÿAŒI, e (d) *indefinita rispetto a I* se non é né applicabile né bloccata. ®

Definizione 6: Un'interpretazione parziale M per D é un *modello parziale* per D se per ogni ÿA in M^-, ogni regola in ground(D) con testa A è bloccata. Se M^+» ÿM^-= H allora M è *totale*. ®

La definizione di modello parziale appena data garantisce che, assumendo un fatto falso, questo non può essere successivamente contraddetto modificando il valore di verità dei fatti indefiniti. Questo fa sì che un modello parziale possa essere esteso in modo da ottenere un modello totale.

Definizione 7: Sia I un'interpretazione parziale per D e X un sottoinsieme non vuoto di H. Diciamo che X é un *insieme infondato* rispetto a I se per ogni A in X, ogni regola r con testa A in ground(D) soddisfa una delle seguenti condizioni:

i) r bloccata rispetto a I;

ii) B(r)«XπΔ. ®

La nozione di insieme infondato può essere vista come una regola di inferenza che consente di derivare conclusioni negative nella semantica well-founded. Infatti le regole che soddisfano la condizione i) non possono essere utilizzate per nuove derivazioni in quanto alcune delle loro ipotesi sono false mentre quelle che soddisfano la condizione ii), la condizione di "infondatezza", richiedono che qualche atomo in X sia vero per poter derivare qualcosa nello stesso insieme. In altre parole non esiste nessun atomo in X che possa essere derivato *per primo*. Conseguentemente, se scegliamo di inferire come falsi alcuni o tutti gli atomi in X, non potremo in seguito derivarne la verità.

Come conseguenza immediata della definizione precedente si ha che l'unione di un numero arbitrario di insiemi infondati é un insieme infondato. Ciò conduce in modo naturale alla seguente:

Definizione 8: Sia D un database e I una interpretazione parziale per D. Denotiamo con $U_D(I)$ *il più grande insieme infondato rispetto a I*, che é dato dall'unione di tutti gli insiemi infondati rispetto a I. ®

Definizione 9: Sia D un database e I una interpretazione parziale per D. Le trasformazioni T_D, U_D e W_D sono definite come segue:

- pŒT_D(I) se e solo se esiste rŒground(D) tale che H(r)=p e B(r)ÕI.
- U_D(I) é il più grande insieme infondato di D rispetto a I, come nella definizione 8.
- $W_D(I)=T_D(I)$ » ÿU_D(I). ®

Per come sono state definite, le trasformazioni T_D, U_D e W_D sono monotone nel lattice completo $< 2^{H»ÿH}, Õ >$.

Definizione 10: *Il modello well-founded* di D è definito come il minimo punto fisso di W_D e viene indicato con WF(D). ®

La definizione appena data non è però costruttiva in quanto non lo è la definizione di insieme infondato. Una definizione costruttiva di modello well-founded è stata presentata dal Van Gelder in [VG89] ed è la seguente.

Definizione 11: Sia D un database e $Γ^-$ un insieme di letterali negativi ground. Indichiamo con D' come il nuovo database ottenuto aggiungendo a ground(D) gli atomi in $Γ^-$ e di conseguenza interpretiamo ÿp(a)ŒΓ come un fatto per ÿp. Le funzioni di trasformazioneS_D, S_D^- e A_D sono definite come segue:

- $S_D(Γ^-)=T_{D'}^{•}(Δ)-Γ^-$;
- $S_D^-(Γ^-)= ÿ(H-S_D(Γ^-))$;
- $A_D(Γ^-)=S_D^-(S_D^-(Γ^-))$.

Se A^- é il minimo punto fisso di A_D allora il modello well-founded di D è $A^-»S_D(A^-)$. ®

La funzione di trasformazione A_D é monotona e quindi ha perfettamente senso parlare di minimo punto fisso. L'ordinale chiusura può essere transfinito quando l'universo di Herbrand é infinito. Naturalmente, se l'universo di Herbrand é finito, si può dimostrare che il minimo punto fisso di A_D é calcolabile in **tempo polinomiale** rispetto alla dimensione dell'universo di Herbrand, [VG89].

Introduciamo ora la seguente definizione di funzione di valutazione, che è una opportuna modifica della definizione data dal Przymusinski in [Prz90]

Definizione 12: Sia I un'interpretazione in L. La *funzione di valutazione* corrispondente a I é una funzione $val_I : C Æ V$ dall'insieme C di tutte le formule chiuse in L all'insieme V={0, 1/2, 1}, definita ricorsivamente come segue:

- Se L é un letterale, allora
  - $val_I(L)=0$ se ÿLŒI;
  - $val_I(L)=1/2$ se abs(L)ŒH-(I^+ » ÿI^-);
  - $val_I(L)=1$ se LŒI.
- Se S é una formula chiusa, allora
  - $val_I(ÿS)=1- val_I(S)$.
- Se S e V sono formule chiuse, allora
  - $val_I(S Ÿ V)=min(val_I(S), val_I(V))$;
  - $val_I(S / V)=max(val_I(S), val_I(V))$;
  - $val_I(V ¨ S)=val_I(V / ÿS)$;
  - $val_I(V ´ S)=val_I((V ¨ S)Ÿ(S ¨ V))$.
- Se S(x) è una formula con una variabile libera x, allora
  - $val_I("x S(x))=min\{val_I(S(A)) : AŒU\}$;
  - $val_I(\$x S(x))=max\{val_I(S(A)) : AŒU\}$;

dove il max (risp. il min) di un insieme vuoto é definito come 0 (risp. 1). ®

Siamo alfine in grado di esporre in modo preciso la nozione di soddisfazione di un vincolo.

Definizione 13: Un database D soddisfa il suo insieme IC di vincoli di integrità se e solo se ogni formula W in IC è conseguenza logica di WF(D), cioè $val_{WF(D)}(W)=1$. ®

Da un punto di vista teorico, é desiderabile che un database soddisfi i suoi vincoli di integrità in ogni momento. Comunque, da un punto di vista pratico, esistono serie difficoltà nel trovare metodi efficienti per controllare i vincoli di integrità dopo ogni aggiornamento. Tale problema é in particolar modo complesso per database deduttivi, dato che l'aggiunta di un singolo fatto può avere un impatto sostanziale sulle conseguenze logiche del database a causa della presenza delle regole. Nonostante tali difficoltà, é possibile ridurre la mole di computazione da effettuare, traendo vantaggio dall'informazione che il database soddisfaceva i suoi vincoli di integrità prima dell'aggiornamento. Notiamo che questa é l'ipotesi di fondo che sta alla base sia del teorema di semplificazione di Lloyd, [LST87], sia della procedura SLDNF estesa di Kowalski, [SaKo87].

3 Metodo per Ricomputare il Modello Well-Founded dopo una Transazione

In questo sezione affronteremo il problema di come ricostruire il modello well-founded di un database modificato dall'applicazione di una transazione. Metteremo in evidenza come la complessità della computazione del nuovo modello sia strettamente correlata alla particolare transazione, introducendo a tal fine il concetto di "transazione conservativa".

Precisiamo ora in dettaglio il contesto al quale faremo riferimento:

- Si assume che l'universo di Herbrand del database considerato sia finito, sarà cioè proibita la presenza di costruttori all'interno delle regole. Questa ipotesi non appare comunque particolarmente restrittiva in considerazione del fatto che stiamo per l'appunto lavorando nel contesto dei database.
- Supponiamo inoltre che il linguaggio sottostante al database considerato rimanga fissato indipendentemente dalle modifiche cui va incontro il database stesso. Dunque, ad esempio, l'inserzione di una nuova regola non può portare all'introduzione di nuove costanti nel linguaggio. Notiamo come questa sia anche l'ipotesi utilizzata dal Lloyd nel suo metodo, sebbene le motivazioni alla base del suo uso siano diverse.
- Infine tratteremo solo database ground: per un database non ground D possiamo sempre pensare di considerare la sua versione istanziata, cioè ground(D).

Da un punto di vista metodologico, in quella che é stata la nostra fase di progettazione, abbiamo ritenuto opportuno ricercare dapprima la soluzione al problema in esame nel caso in cui il database fosse modificato in seguito ad una "transazione elementare", una transazione cioè costituita da una singola cancellazione o da una singola inserzione di una regola ground. Sulla base dei risultati ottenuti, siamo poi passati a considerare "transazioni generali", formate da un numero qualsiasi di cancellazioni e/o inserzioni di regole ground.

3.1 Computazione del Modello Well-Founded in seguito ad una Transazione Elementare

Sia D un database e D' il database ottenuto da D mediante l'applicazione di una transazione elementare **t**.

Supponiamo di avere a disposizione il modello well-founded di D, WF(D): vediamo come determinare il modello well-founded del programma aggiornato. Il nostro intento é quello di ottimizzare il calcolo di WF(D'), avvalendoci in un qualche modo del vecchio modello. A tal fine

determiniamo, dopo ogni aggiornamento del programma, l'insieme dei letterali contenuti nel vecchio modello, le derivazioni dei quali non sono sicuramente "intaccate" dall'aggiornamento stesso. Tale insieme, per costruzione, è contenuto nel nuovo modello well-founded e quindi può essere utilizzato per il calcolo di quest'ultimo. La porzione di WF(D) contenuta in WF(D') dipende dal tipo di transazione applicata al programma; come vedremo, per particolari transazioni, vale che WF(D) é interamente contenuto in WF(D').

Definizione 14: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione **t**. Diciamo che **t** é *conservativa* se WF(D)ÕWF(D'). ®

Esaminiamo ora in dettaglio come calcolare WF(D') in seguito ad un aggiornamento di D. La computazione si avvarrà delle definizioni delle trasformazioni S_D e A_D.

L'analisi procederà distinguendo vari casi in base alla forma della transazione: quest'ultima può consistere nell'inserzione o nella rimozione di una regola, la cui testa può essere affermata, negata o "indefinita" rispetto al modello. Più precisamente, se D é un database, i tipi di transazione esaminati sono i seguenti:

$\mathbf{t}_1$: inserzione in D di una regola r con H(r)ŒWF(D);

$\mathbf{t}_2$: rimozione da D di una regola r con ÿH(r)ŒWF(D);

$\mathbf{t}_3$: inserzione in D di una regola r con H(r) "indefinita" rispetto a WF(D), cioè H(r)Œ(H–(WF(D)$^+$»ÿWF(D)$^-$);

$\mathbf{t}_4$: rimozione da D di una regola r con H(r) indefinita rispetto a WF(D);

$\mathbf{t}_5$: inserzione in D di una regola r con ÿH(r)ŒWF(D);

$\mathbf{t}_6$: rimozione da D di una regola r con H(r)ŒWF(D).

E' facile vedere che questi sono tutti e soli i possibili casi che possono presentarsi qualora si consideri una transazione elementare.

Lemma 1: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione elementare **t** di tipo $\mathbf{t}_1$ o $\mathbf{t}_2$. Il modello well-founded di D' coincide con il modello well-founded di D, cioè WF(D')=WF(D). ®

Corollario 1: Le transazioni elementari di tipo $\mathbf{t}_1$ e $\mathbf{t}_2$ sono conservative. ®

Definizione 15: Sia D un database, I una interpretazione per D e D' il database ottenuto da D in seguito all'applicazione di una transazione elementare **t** di tipo $\mathbf{t}_3$ o $\mathbf{t}_4$. La trasformazione T_{elem1} é definita come segue:

$T_{elem1}(\mathbf{t}, D, I)=A^-$» A^+ dove $A^-= A_{D'}{}^{\bullet}(I^-)$ e $A^+= S_{D'}(A^-)$. ®

Il significato intuitivo di T_{elem1} é legato al fatto che, qualora il database D venga modificato tramite una transazione elementare **t** di tipo $\mathbf{t}_3$ o $\mathbf{t}_4$, sembra ragionevole pensare che **t** influisca solo sulla parte indefinita di WF(D). In tal caso sarebbe possibile calcolare WF(D') applicando l'operatore del Van Gelder, cioè A_D, a partire però da WF(D)$^-$ e non da Δ, risparmiando in questo modo gran parte del lavoro. Quanto appena detto viene confermato dal seguente lemma.

Lemma 2: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione elementare **t** di tipo $\mathbf{t}_3$ o $\mathbf{t}_4$. WF(D') é ottenuto da WF(D) applicando la trasformazione T_{elem1} , cioè WF(D')=T_{elem1}(**t**, D, WF(D)). ®

Corollario 2: Le transazioni elementari di tipo $\mathbf{t}_3$ e $\mathbf{t}_4$ sono conservative. ®

Analizziamo ora delle particolari transazioni elementari di tipo $\mathbf{t}_3$ e $\mathbf{t}_4$, per le quali vale che WF(P')=WF(P).

Lemma 3: Sia D un database e D' il database ottenuto inserendo in D una regola r tale che H(r) è indefinita rispetto a WF(D). Se r è bloccata o indefinita rispetto a WF(D) allora WF(D')=WF(D).®

Lemma 4: Sia D un database e D' il database ottenuto rimuovendo da D una regola r tale che H(r) è indefinita rispetto a WF(D). Se r verifica una delle seguenti condizioni:

(i) r bloccata rispetto a WF(D);

(ii) r è indefinita rispetto a WF(D) ed esiste r'ŒD' con H(r')=H(r) e r' indefinita rispetto a WF(D);

allora si ha che WF(D')=WF(D). ®

Per analizzare gli ultimi due casi, individuiamo la porzione di WF(D) sicuramente contenuta in WF(D') e utilizziamo tale insieme come input per l'operatore A_D nel calcolo di WF(D'). Per far ciò sono necessarie alcune definizioni preliminari.

Definizione 16: Dato un database D e due atomi A e B, diciamo che A *riferisce* B in D se e solo se esiste una regola r in D tale che H(r)=A e abs(L)=B per un qualche LŒB(r). La relazione *dipende da* é la chiusura riflessiva e transitiva della relazione riferisce. ®

Definizione 17 Sia D un database ed A un atomo. Indichiamo con dep(A, D) l'insieme degli atomi che dipendono da A in D, cioè *dep*(A, D)={C | CŒH e C dipende da A in D}. ®

Definizione 18: Sia D un database, I una interpretazione per D e D' il database ottenuto da D in seguito all'applicazione di una transazione elementare **t** di tipo $\mathbf{t}_5$ e $\mathbf{t}_6$. Se indichiamo con X l'insieme {ÿA | ÿAŒI⁻ e AœdepH(r_t), D) dove r_t é la regola che compone **t**.}, la trasformazione T_{elem2} é definita come segue: T_{elem2}(**t**, D, I)=A^-» A^+ dove A^-= $A_{D'}$•(X) e A^+= $S_{D'}(A^-)$. ®

Il significato intuitivo di T_{elem2} é legato al fatto che, qualora il database D venga modificato in seguito all'applicazione di una transazione elementare **t** di tipo $\mathbf{t}_5$ e $\mathbf{t}_6$, é possibile calcolare WF(D') applicando l'operatore del Van Gelder a partire da X definito come sopra.

Lemma 5: Sia D un database e D' il database ottenuto da D in seguito all'applicazione a D di una transazione elementare **t** di tipo $\mathbf{t}_5$ e $\mathbf{t}_6$. WF(D') é ottenuto da WF(D) applicando la trasformazione T_{elem2} , cioè WF(D')=T_{elem2}(**t**, D, WF(D)). ®

Analizziamo ora delle particolari transazioni elementari di tipo $\mathbf{t}_5$ e $\mathbf{t}_6$, per le quali vale che WF(D')=WF(D) e che quindi sono conservative.

Lemma 6: Sia D un database e D' il database ottenuto inserendo in D una regola r tale che ÿH(r)ŒWF(D). Se r é bloccata rispetto a WF(D) ed esiste LŒB(r) tale che ÿLŒWF(D) e abs(L)œdep(H(r), D), allora WF(D')=WF(D). ®

Lemma 7: Sia D un database e D' il database ottenuto rimuovendo da D una regola r con H(r)ŒWF(D). Nei seguenti casi:

(i) r bloccata o indefinita rispetto a WF(D)

(ii) r applicabile rispetto a WF(D) ed esiste una regola r'ŒD' con H(r')=H(r_t), r' applicabile rispetto a WF(D) e tale che per ogni LŒB(r') abs(L)œdep(H(r_t), D)

si ha che WF(D')=WF(D). ®

Esaminati i casi basilari ai quali faremo riferimento, risulta subito chiaro come possa essere operata la fase di ricalcolo del modello nella situazione in cui siano considerate transazioni generali.

3.2 Computazione del Modello Well-Founded in seguito ad una Transazione Generale

Come premesso, all'interno di questo paragrafo forniremo una estensione della metodologia da noi proposta per la computazione del modello well-founded, nel caso in cui venga applicata al database una transazione generale.

Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t**. Esaminiamo in dettaglio come calcolare il modello well-founded del database aggiornato. Distinguiamo tre tipi di transazione generale:

$\mathbf{G_1}$: **t** é composta esclusivamente da transazioni elementari di tipo $\mathbf{t_1}$ e/o $\mathbf{t_2}$;

$\mathbf{G_2}$: **t** contiene solo transazioni elementari di tipo $\mathbf{t_1}$, $\mathbf{t_2}$, $\mathbf{t_3}$ e/o $\mathbf{t_4}$, tra le quali almeno una di tipo $\mathbf{t_3}$ o $\mathbf{t_4}$;

$\mathbf{G_3}$: **t** contiene almeno una transazione elementare di tipo $\mathbf{t_5}$ o $\mathbf{t_6}$.

Lemma 8: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t** di tipo $\mathbf{G_1}$. Il modello well-founded di D' coincide con il modello well-founded di D, cioè WF(D')=WF(D). ®

Corollario 5: Le transazioni generali di tipo $\mathbf{G_1}$ sono conservative. ®

Le definizioni che seguono generalizzano opportunamente quelle date nel paragrafo precedente.

Definizione 18: Sia D un database, I una interpretazione per D e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t**. La trasformazione T_{gen1} é definita come segue: $T_{gen1}(\mathbf{t}, D, I)=A^- \text{»} A^+$ dove $A^-= A_{D'}^{\bullet}(I^-)$ e $A^+= S_{D'}(A^-)$.®

Lemma 9: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t** di tipo $\mathbf{G_2}$. WF(D') é ottenuto da WF(D) applicando la trasformazione T_{gen1}, cioè $WF(D')=T_{gen1}(\mathbf{t}, D, WF(D))$. Inoltre, se tutte le transazioni elementari di tipo $\mathbf{t_3}$ in **t** soddisfano le ipotesi del lemma 3 e tutte le transazioni elemantari di tipo $\mathbf{t_4}$ in **t** soddisfano le ipotesi del lemma 4, allora WF(D')=WF(D). ®

Corollario 6: Le transazioni generali di tipo $\mathbf{G_2}$ sono conservative. ®

Definizione 19: Sia D un database e **t** una transazione generale di tipo $\mathbf{G_3}$. Definiamo $DEP(\mathbf{t}, D) = \bigcup_{\mathbf{t'}} dep(H(r_{\mathbf{t'}}), D)$ con **t'** transazione elementare di tipo $\mathbf{t_5}$ o $\mathbf{t_6}$ contenuta in **t**. ®

Definizione 20: Sia D un database, I una interpretazione per D e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t**. Se indichiamo con X l'insieme $\{\forall A \mid \forall A \in I^- \text{ e } A \in DEP(\mathbf{t}, D)\}$, la trasformazione T_{gen2} é definita come segue: $T_{gen2}(\mathbf{t}, D, I) = A^- \cup A^+$ dove $A^- = A_{D'}^{\bullet}(X)$ e $A^+ = S_{D'}(A^-)$. ®

Lemma 10: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t** di tipo $\mathbf{G_3}$. WF(D') é ottenuto da WF(D) applicando la trasformazione T_{gen2}, cioè $WF(D') = T_{gen2}(\mathbf{t}, D, WF(D))$. ®

Lemma 11: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t** di tipo $\mathbf{G_3}$. Se valgono le seguenti condizioni:

(a) tutte le transazioni elementari di tipo $\mathbf{t_3}$ in **t** soddisfano le ipotesi del lemma 3;
(b) tutte le transazioni elementari di tipo $\mathbf{t_4}$ in **t** soddisfano le ipotesi del lemma 4;
(c) tutte le transazioni elementari di tipo $\mathbf{t_5}$ in **t** soddisfano le ipotesi del lemma 6;
(d) tutte le transazioni elementari di tipo $\mathbf{t_6}$ in **t** soddisfano le ipotesi del lemma 7;

allora WF(D')=WF(D). Se valgono solo le condizioni (c) e (d) allora $WF(D') = T_{gen1}(\mathbf{t}, D, WF(D))$. In entrambi i casi la transazione è conservativa. ®

4 Esposizione del Metodo

Descriviamo ora la soluzione al problema del controllo della consistenza di un database deduttivo D rispetto ai suoi vincoli di integrità in seguito ad una transazione generale **t**.

L'ipotesi basilare che si utilizza é, come avviene per i metodi del Lloyd e del Kowalski, quella secondo cui il database soddisfa i suoi vincoli di integrità prima della transazione e dunque, che ogni violazione dei vincoli nel database modificato deve coinvolgere almeno una delle transazioni elementari che compongono **t**.

Comunque, per poter esporre l'algoritmo da noi proposto, abbiamo bisogno di alcuni risultati preliminari.

Lemma 12: Siano I e I' due interpretazioni in L tali che $I \tilde{O} I'$. Se W é una formula chiusa in L allora si ha che:

a) Se $val_I(W)=1$ allora $val_{I'}(W)=1$.
b) Se $val_I(W)=0$ allora $val_{I'}(W)=0$. ®

Teorema 1: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t**. Supponiamo che D soddisfi il suo insieme di vincoli di integrità IC. Se **t** é conservativa allora D' soddisfa l'insieme di vincoli IC. ®

Teorema 2: Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione generale **t** che contiene almeno una transazione elementare di tipo 5) o 6). Supponiamo che D soddisfi IC. Indichiamo con X l'insieme $\{\forall A \mid \forall A \in WF(D)^- \text{ e } A \in DEP(\mathbf{t}, D)\}$ e con Y l'insieme $X \cup S_{D'}(X)$. Se $val_Y(W)=1$ per ogni vincolo $W \in IC$, allora anche D' soddisferà tale insieme di vincoli. ®

ALGORITMO

Passo 1. Se, in base ai risultati ottenuti nel capitolo precedente, possiamo stabilire che **t** é conservativa, allora per il teorema 1 anche D' soddisfa l'insieme IC. STOP. Altrimenti vai al passo 2.

Passo 2. Sia X l'insieme {ÿA | ÿAŒWF(D)¯ e AœDEP(**t**, D)} e Y l'insieme X»$S_{D'}$(X). Se $val_Y(W)=1$ per ogni vincolo WŒIC, allora per il teorema 2 anche D' soddisfa tale insieme di vincoli. STOP. Altrimenti, se esiste WŒIC tale che $val_Y(W)=0$, allora D' viola W e quindi non soddisfa IC. STOP. Altrimenti, esiste cioè WŒIC tale che $val_Y(W)=1/2$, vai al passo 3.

Passo 3. Sia X' l'insieme $A_{D'}(X)$. Se X'=X allora D' non soddisfa IC. STOP. Altrimenti si assegna ad X l'insieme X' e ad Y l'insieme X » $S_{D'}(X)$. Se $val_Y(W)=1$ per ogni vincolo WŒIC allora anche D' soddisfa tale insieme di vincoli. STOP. Altrimenti se esiste WŒIC tale che $val_Y(W)=0$, allora D' viola W e quindi non soddisfa IC. STOP. Altrimenti riesegui il passo 3. ®

Notiamo come nel passo 3 venga eseguito un passo della computazione di WF(D'). La condizione X'=X implica che X' è un punto fisso per $A_{D'}$ e dunque che X'=WF(D')¯. Da quanto detto segue che D' non soddisfa i vincoli in quanto $\min\{val_{WF(D')}(W) \mid W\text{Œ}IC\}=1/2$.

5 Considerazioni Generali sull'Algoritmo e Conclusioni

Vediamo ora di esporre chiaramente l'idea che é alla base del nostro algoritmo.

Sia D un database e D' il database ottenuto da D in seguito all'applicazione di una transazione **t**. Supponiamo che D soddisfi un insieme IC di vincoli di integrità. Il nostro intento é stato quello di individuare una classe, la più ampia possibile, di transazioni per la quale si può affermare che D' soddisfa l'insieme di vincoli in IC, senza dover computare WF(D') o parte di esso. A questo proposito é stato introdotto il concetto di transazione conservativa. Chiaramente, se la transazione é conservativa, D' soddisfa ancora l'insieme IC, in quanto WF(D) é interamente contenuto in WF(D'). Altrimenti, nel caso in cui non é possibile decidere se t é conservativa, l'algoritmo individua una porzione di WF(D) che é sicuramente contenuta in WF(D'). Se questa porzione non é sufficiente per affermare che l'insieme di vincoli in IC é soddisfatto da D' oppure che esiste un vincolo W violato da D', l'algoritmo computa la minima parte di WF(D') grazie alla quale é possibile decidere la soddisfacibilità dei vincoli.

Resta da affrontare il problema di come stabilire se una transazione é conservativa.

In precedenza abbiamo fornito dei criteri che risultano essere sufficienti per una tale verifica. Questi criteri si basano nel controllare se le teste delle regole che compongono la transazione appartengono o meno a WF(D). Sembrerebbe quindi che l'algoritmo necessiti dell'intero modello well-founded di D. In realtà, per poter applicare l'algoritmo é sufficiente avere a disposizione un sottoinsieme di WF(D) che permetta di affermare che D soddisfa l'insieme di vincoli in IC. Tale sottoinsieme è proprio l'insieme Y calcolato a regime dall'algoritmo. Quindi per controllare se la transazione t applicata a D é conservativa, si utilizza per l'appunto Y.

Lo scopo di questo articolo è quello di presentare un nuovo metodo per il controllo della consistenza del database rispetto ai suoi vincoli di integrità, superando la restrizione, comune a tutti gli altri metodi presenti in letteratura, di dover considerare solo database stratificate.

Gli sviluppi futuri potrebbero essere i seguenti:

- ridurre ulteriormente l'insieme Y;
- considerare come semantica del database il modello massimale deterministico, [SZ90].

6 Bibliografia

[Apt88] K.R.Apt, *Introduction to Logic Programming*, Revised and extended version, 1988.

[CGT91] S.Ceri, G.Gottlob, L.Tanca, *Logic Programming and Databases*, New York: Springer-Verlag, 1991.

[GL88] M.Gelfond, V.Lifschitz, *The Stable Model Semantics for Logic Programming*, in: R.Kowalski, K.Bowen, editors, Proceedings of the Fifth Logic Programming Symposium, pp. 1070-1080, Cambridge, Mass., 1988. Association for Logic Programming, MIT Press.

[LST87] J.W.Lloyd, E.A.Sonenberg, R.W.Topor, *Integrity Constraint Checking in Stratified Databases*, in: Journal of Logic Programming 4: pp.331-343, 1987.

[Prz90a] T.C.Przymusinski, *Extended Stable Semantics for Normal and Disjunctive Logic Programs*, in: Proceedings of the Seventh International Logic Programming Conference, Jerusalem, Israel, pp. 459-477, Cambridge, Mass., 1990. Association for Logic Programming, MIT Press.

[SZ90] D.Saccà, C.Zaniolo, *Stable Models in Logic Programs with Negation and Non-Determinism*, in: Proc. ACM PODS Symp., 1990.

[SZ91] D.Saccà, C.Zaniolo, *Partial Models and Three-Valued Models in Logic Programs with Negation*, in: Proceedings First Internatinal Workshop on Logic Programming and Non-Monotonic Reasoning, (Nerode, Marek e Subrahnia-man eds.), MIT Press, 1991, pp. 87-102.

[SaKo88] F.Sadri, R.Kowalski, *A Theorem-Proving Approach to Database Integrity*, in: Foundations of Deductive Databases and Logic Programming, (Minker J., ed.), Morgan Kaufmann, Los Altos, 1988.

[VG89] A.Van Gelder, *The Alternating Fixpoint of Logic Programs with Negation*, in: Proceedings of the Eighth Symposium on Principles of Database Systems, pp. 1-10. ACM SIGATC-SIGMOD, March 1989.

[VGRS88] A.Van Gelder, K.A.Ross, J.S.Schlipf, *Unfounded Sets and Well-Founded Semantics for General Logic Programs*, in: ACM SIGMOD-SIGACT Symp. on Principles of Database Systems, March. 1988, pp. 221-230.

[VGRS91] A.Van Gelder, K.A.Ross, J.S.Schlipf, *The Well-Founded Semantics for General Logic Programs*, in: Journal of the ACM, July 1991, pp. 620-650.

[YoYu91] J.H.You, L.Y.Yuan, *Extended Well-Founded Model Semantics for General Logic Programs*, in: Proceedings of the Eighth International Conference, edited by Koichi Furukawa, 1991, pp. 412-425.