ΕΠΑΓΩΓΙΚΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Γ. ΣΤΑΥΡΑΚΑΣ, Π. ΒΑΣΙΛΕΙΑΔΗΣ

9.1 ΕΙΣΑΓΩΓΗ

Μια επαγωγική βάση δεδομένων είναι ένα σύστημα βάσεων δεδομένων που έχει τη δυνατότητα να ορίζει κανόνες (rules)[1], οι οποίοι μπορούν να συνάγουν επιπλέον πληροφορία από τα δεδομένα που είναι αποθηκευμένα στη βάση δεδομένων [EN94].

Σε ένα σύστημα επαγωγικών βάσεων δεδομένων, συνήθως χρησιμοποιείται μια δηλωτική (declarative) γλώσσα για τον καθορισμό των κανόνων. Με τον όρο "δηλωτική" χαρακτηρίζονται όλες εκείνες οι γλώσσες στις οποίες ο χρήστης προσδιορίζει *τι* ακριβώς είναι το ζητούμενο του προγράμματος (και όχι τον τρόπο με τον οποίο το ζητούμενο υπολογίζεται). Μια *μηχανή εξαγωγής συμπερασμάτων (inference engine),* η οποία αποτελεί κομμάτι της αρχιτεκτονικής του συστήματος, χρησιμοποιείται για την αξιοποίηση των κανόνων αυτών και τη συναγωγή νέων δεδομένων. Το μοντέλο που χρησιμοποιείται στις επαγωγικές βάσεις δεδομένων μοιάζει με το σχεσιακό (και κυρίως με το σχεσιακό λογισμό) από τη μια και με τη Prolog και το λογικό προγραμματισμό από την άλλη. Η γλώσσα που χρησιμοποιείται είναι ένα υποσύνολο της Prolog και ονομάζεται *Datalog*. Η Datalog μπορεί να μοιάζει με την Prolog, όμως η *εκτελεστική σημασιολογία* της (operational semantics) -ο τρόπος, δηλαδή, με τον οποίο εκτελούνται τα προγράμματά της- είναι ανοιχτός.

Μια επαγωγική βάση δεδομένων χρησιμοποιεί κυρίως δύο τύπους για τον ορισμό γνώσης: *δεδομένα (facts)* και *κανόνες (rules)*. Τα γεγονότα ορίζονται με τρόπο παρόμοιο με τους σχεσιακούς πίνακες (με τη μόνη διαφορά ότι δε χρειάζεται να ορίσει κανείς ονόματα πεδίων, καθώς η θέση μιας τιμής μέσα σε μια πλειάδα καθορίζει μονοσήμαντα σε ποια πραγματική πληροφορία αντιστοιχεί). Οι κανόνες μοιάζουν με σχεσιακές όψεις (views), καθώς ορίζουν σχέσεις που δεν είναι αποθηκευμένες, αλλά μπορούν να εξαχθούν από τα αποθηκευμένα

[1] Προσοχή: η έννοια των κανόνων εδώ είναι διαφορετική από την αντίστοιχη έννοια στις ενεργές βάσεις δεδομένων.

γεγονότα. Η βασική διαφορά κανόνων και σχεσιακών όψεων είναι ότι οι κανόνες μπορεί να περιλαμβάνουν αναδρομή, ενώ οι όψεις όχι.

Η αποτίμηση των προγραμμάτων σε Prolog έχει κατασκευαστεί με τέτοιο τρόπο, ώστε να επεξεργάζεται δεδομένα στην κύρια μνήμη του υπολογιστή. Στην περίπτωση, όμως, μιας βάσης δεδομένων, έχουμε να κάνουμε με μεγάλους όγκους δεδομένων, αποθηκευμένων στο δίσκο. Ως εκ τούτου, είναι απαραίτητη η χρησιμοποίηση εναλλακτικών τεχνικών για την αποδοτική εκτέλεση των προγραμμάτων. Στη συνέχεια του κεφαλαίου αυτού θα εξετάσουμε αναλυτικότερα αυτές τις τεχνικές.

Το κεφάλαιο αυτό βασίζεται κυρίως στο [EN94].

9.2 ΣΥΜΒΟΛΙΣΜΟΣ ΣΤΗΝ PROLOG/DATALOG

Στην ενότητα αυτή θα εισάγουμε το βασικό συμβολισμό για κανόνες εκπεφρασμένους σε Prolog/Datalog. Ο βασικός μηχανισμός είναι η παράθεση *κατηγορημάτων (predicates)* με μοναδικό όνομα. Κάθε κατηγόρημα έχει ένα όνομα (που έμμεσα οφείλει να προδίδει το νόημα του κατηγορήματος) και μια σειρά από *ορίσματα (arguments)*. Στην περίπτωση που όλα τα ορίσματα είναι σταθερές, το κατηγόρημα αναπαριστά ένα γεγονός. Αν, όμως το κατηγόρημα έχει μεταβλητές σαν ορίσματα, μπορεί να θεωρηθεί ερώτηση (query), μέρος ενός κανόνα ή περιορισμός. Θεωρούμε τις εξής συμβάσεις:

- οι σταθερές τιμές είναι αριθμητικές ή αλφαριθμητικές και συμβολίζονται με *μικρά* γράμματα,
- οι μεταβλητές συμβολίζονται με *κεφαλαία* γράμματα.

9.2.1 Παράδειγμα

Έστω το παρακάτω παράδειγμα από το [EN94]. Έχουμε τρία κατηγορήματα: *επιβλέπει, ανώτερος, κατώτερος*. Το κατηγόρημα *επιβλέπει* ορίζεται από ένα σύνολο γεγονότων, κάθε ένα από τα οποία έχει δύο ορίσματα: το όνομα ενός επιβλέποντος, και το όνομα ενός άμεσα επιβλεπόμενου από τον επιβλέποντα αυτόν. Τα γεγονότα αυτά αντιστοιχούν στα δεδομένα που αποθηκεύονται στη βάση δεδομένων (με την ίδια συλλογιστική που θα είχαμε ένα πίνακα `SUPERVISE` με δύο πεδία: `SUPERVISE(SUREVISOR, SUPERVISEE)`). Ως εκ τούτου, η δήλωση `επιβλέπει(X, Y)` περιγράφει το γεγονός ότι ο X επιβλέπει άμεσα τον Y.

Τα υπόλοιπα γεγονότα θα τα εκφράσουμε με κανόνες. Η μεγάλη ισχύς των επαγωγικών βάσεων δεδομένων βρίσκεται στη δυνατότητα ορισμού αναδρομικών κανόνων και εξαγωγής πληροφορίας μέσω αυτών. Ένας κανόνας είναι της μορφής:

```
head :- body
```

Συνήθως, αριστερά του συμβόλου `:-` βρίσκεται ακριβώς ένα κατηγόρημα, το οποίο αποκαλείται *κεφαλή (head)*, ή *αριστερό μέρος (left hand side -LHS)*, ή *συμπέρασμα (conclusion)* του κανόνα. Στην έκφραση δεξιά του συμβόλου `:-` βρίσκονται ένα ή περισσότερα κατηγορήματα. Η εν λόγω έκφραση αποκαλείται το *σώμα (body)*, ή *δεξί μέρος (right hand side -RHS)*, ή *προϋπόθεση (premise)* του κανόνα. Η παράθεση δύο κατηγορημάτων του σώματος με κόμμα (",") ανάμεσά τους, σημαίνει *λογική σύζευξη (AND)*. Ένα κατηγόρημα με σταθερές για ορίσματα ονομάζεται *βασικό (ground)*.

Ένας κανόνας ορίζει ότι, αν μια συγκεκριμένη ανάθεση (binding) σταθερών τιμών στις μεταβλητές του σώματος κάνει όλα τα κατηγορήματα του σώματος να παίρνουν αληθή τιμή, τότε και η κεφαλή γίνεται αληθής για την ίδια ανάθεση τιμών.

Έτσι, με βάση γεγονότα που ήδη γνωρίζουμε, μπορούμε να χτίζουμε λογικές συνεπαγωγές με βάση τις αναθέσεις τιμών, καθώς *κάθε ανάθεση τιμών που κάνει την κεφαλή αληθή προσδιορίζει ένα νέο γεγονός*.

Στο σχήμα 9.1 παρατηρούμε τον ορισμό του κατηγορήματος ανώτερος, του οποίου το πρώτο όρισμα είναι το όνομα ενός υπαλλήλου και το δεύτερο όρισμα είναι ένας κατώτερος (έμμεσα ή άμεσα) του εν λόγω υπαλλήλου. Ως εκ τούτου ανώτερος (X, Y) σημαίνει ότι ο X επιβλέπει άμεσα ή έμμεσα τον Y. Παρατηρεί κανείς ότι ο ορισμός του κατηγορήματος αποτελείται από δύο δηλώσεις κανόνων. Ο πρώτος κανόνας λέει ότι για κάθε τιμή του X και του Y, αν το επιβλέπει(X, Y) είναι αληθές, τότε και το ανώτερος(X, Y) είναι αληθές. Ο δεύτερος αναδρομικός κανόνας λέει ότι αν επιβλέπει(X, Z) και ανώτερος(Z, Y) είναι ταυτόχρονα αληθή για μια ανάθεση τιμών, τότε και το ανώτερος(X, Y) είναι αληθές. Εδώ έχουμε ένα παράδειγμα αναδρομικού κανόνα, όπου το δεξί μέρος του κανόνα περιέχει ένα κατηγόρημα που βρίσκεται και το αριστερό μέρος του κανόνα. Επίσης, πρέπει να επισημάνουμε, ότι *στην περίπτωση που έχουμε παραπάνω από έναν κανόνες με το ίδιο αριστερό μέρος (ήτοι, ορίζουμε ένα κατηγόρημα με παραπάνω από ένα κανόνες), το κατηγόρημα γίνεται αληθές αν οποιοσδήποτε από τους κανόνες γίνει αληθής: ως εκ τούτου έχουμε τη λογική διάζευξη (OR) των δεξιών μελών*.

Ένα σύστημα Prolog περιέχει ένα σύνολο από *ενσωματωμένα (built-in)* κατηγορήματα. Αυτά, εν γένει, περιλαμβάνουν το κατηγόρημα ισότητας =(X, Y) και το οποίο μπορεί και να εκφραστεί ως X = Y, καθώς και τα κατηγορήματα <, >, <=, >=. Ιδιαίτερη προσοχή θέλει το ζήτημα των αριθμητικών συναρτήσεων: ενώ στην Prolog μπορούμε να έχουμε αριθμητικές συναρτήσεις σαν ορίσματα στα κατηγορήματα (+, −, κλπ.), στην κλασσική Datalog δεν επιτρέπεται η χρήση συναρτήσεων. Αυτό αποτελεί και τη βασική διαφορά μεταξύ Prolog και Datalog.

Μια ερώτηση (query) περιλαμβάνει το σύμβολο ενός κατηγορήματος, με κάποιες μεταβλητές σαν παραμέτρους και σκοπός της είναι να συνάγει τους διαφορετικούς συνδυασμούς σταθερών που όταν ανατεθούν στις μεταβλητές, κάνουν το κατηγόρημα αληθές. Για παράδειγμα, η πρώτη ερώτηση του σχήματος 9.1 ζητά τα ονόματα όλων των κατώτερων του "λούκυ" (άμεσα και έμμεσα). Η δεύτερη ερώτηση, όπου όλα τα ορίσματα έχουν ανατεθεί σε σταθερές, απλώς ψάχνει για μια boolean απάντηση.

Κανόνες

```
ανώτερος(X, Y) :- επιβλέπει (X, Y).
ανώτερος(X, Y) :- επιβλέπει (X, Z), ανώτερος (Z, Y).
κατώτερος(X, Y) :- ανώτερος (Y, X).
```

Ερωτήσεις

```
ανώτερος(λούκυ, Y)?
ανώτερος(λούκυ, μαμά_ντάλτον)?
```

Σχήμα 9.1 Κανόνες και ερωτήσεις σε Prolog

9.2.2 Συμβολισμός σε Datalog

Στην Datalog, ένα πρόγραμμα γράφεται από τμήματα που καλούνται *ατομικές φόρμουλες (atomic formulas)*. Στην Datalog οι ατομικές φόρμουλες είναι *λέξεις (literals)* της μορφής $p(a_1, a_2, \ldots, a_n)$, όπου p είναι ένα κατηγόρημα και n είναι ο αριθμός των ορισμάτων του κατηγορήματος p (αποκαλούμενος και *arity* του p). Τα ορίσματα μπορεί να είναι είτε σταθερές, είτε μεταβλητές. Τα ενσωματωμένα κατηγορήματα της Datalog είναι τα <(less),

`>(greater)`, `<=(less equal)`, `>=(greater equal)` για πεδία ορισμού με σχέση διάταξης και τα `=(equal)`, `/= (not equal)` για πάσης φύσεως πεδία. Ο σύνταξη μπορεί να είναι είτε `=(X, Y)` είτε `X = Y`. Απαιτεί ιδιαίτερη προσοχή το γεγονός ότι ακριβώς επειδή τα πεδία ορισμού αυτών των κατηγορημάτων είναι πιθανώς άπειρα, μπορεί να δημιουργηθεί πρόβλημα στον ορισμό των κανόνων. Για παράδειγμα, το κατηγόρημα `greater(X, 3)` αν χρησιμοποιηθεί μόνο του, παράγει ένα άπειρο σύνολο από τιμές για το X που ικανοποιεί το κατηγόρημα (όλοι οι ακέραιοι που είναι μεγαλύτεροι από το 3).

Μια λέξη είναι είτε μια ατομική φόρμουλα (οπότε ονομάζεται *θετική λέξη - positive literal*), είτε μια ατομική φόρμουλα της οποίας προηγείται το `not` (οπότε ονομάζεται *αρνητική λέξη - negative literal*). Τα προγράμματα Datalog μπορούν να θεωρηθούν σαν υποσύνολο του κατηγορικού λογισμού που μοιάζει με τον σχεσιακό λογισμό. Στην Datalog, όμως, οι φόρμουλες κατηγορικού λογισμού μετασχηματίζονται σε *προτασιακό φορμαλισμό (clausal form)* πριν εκφραστούν σε ένα υποσύνολο του προτασιακού φορμαλισμού, τις *προτάσεις Horn*.

9.2.3 Προτασιακός φορμαλισμός και προτάσεις Horn

Στον προτασιακό φορμαλισμό, μια φόρμουλα πρέπει να έχει τα παρακάτω χαρακτηριστικά:

- Όλες οι μεταβλητές της φόρμουλας είναι *καθολικά ποσοτικοποιημένες (universally quantified)*. Ως εκ τούτου, πουθενά δεν εμφανίζεται ο καθολικός ποσοδείκτης (∀), καθώς υπονοείται.
- Στον προτασιακό φορμαλισμό μια φόρμουλα αποτελείται από ένα σύνολο *προτάσεων (clauses)*. Κάθε πρόταση αποτελείται από ένα σύνολο λέξεων που συνδέονται με *λογικές διαζεύξεις (OR)*.
- Οι προτάσεις συνδέονται με *λογικές συζεύξεις (AND)* για την κατασκευή μιας φόρμουλας.

Μπορεί να αποδειχθεί ότι οποιαδήποτε φόρμουλα μπορεί να αναχθεί σε μια φόρμουλα προτασιακού φορμαλισμού. Ας υποθέσουμε μια πρόταση της μορφής:

$$\text{not}(P_1)\ \text{OR}\ \text{not}(P_2)\ \text{OR}\ \ldots\ \text{OR}\ \text{not}(P_n)\ \text{OR}\ Q_1\ \text{OR}\ Q_2\ \text{OR}\ \ldots\ \text{OR}\ Q_m \qquad (1)$$

Η πρόταση αυτή έχει n θετικές λέξεις και m αρνητικές. Μπορούμε να μετασχηματίσουμε την εν λόγω πρόταση στην παρακάτω ισοδύναμη λογική φόρμουλα:

$$P_1\ \text{AND}\ P_2\ \text{AND}\ \ldots\ \text{AND}\ P_n\ \text{=>}\ Q_1\ \text{OR}\ Q_2\ \text{OR}\ \ldots\ \text{OR}\ Q_m \qquad (2)$$

Οι φόρμουλες (1) και (2) είναι ισοδύναμες (ήτοι έχουν πάντα τις ίδιες αληθοτιμές). Αν όλα τα P_i είναι αληθή, τότε για να είναι η (2) αληθής πρέπει τουλάχιστο ένα Q_i να είναι αληθές. Αντίστοιχα, αν όλα τα P_i είναι αληθή, όλες οι αρνήσεις τους είναι ψευδείς, οπότε για να είναι η (1) αληθής, πρέπει τουλάχιστο ένα Q_i να είναι αληθές.

Στην Datalog, οι κανόνες εκφράζονται σε ένα υποσύνολο του προτασιακού φορμαλισμού, ονόματι *προτάσεις Horn (Horn clauses)* στο οποίο κάθε πρόταση μπορεί να περιέχει το πολύ μια θετική λέξη. Άρα λοιπόν, μια πρόταση Horn είναι της μορφής:

$$\text{not}(P_1)\ \text{OR}\ \text{not}(P_2)\ \text{OR}\ \ldots\ \text{OR}\ \text{not}(P_n)\ \text{OR}\ Q \qquad (3)$$

ή της μορφής

$$\text{not}(P_1)\ \text{OR}\ \text{not}(P_2)\ \text{OR}\ \ldots\ \text{OR}\ \text{not}(P_n) \qquad (4)$$

Η πρόταση Horn του (3) μπορεί να μετασχηματιστεί στην πρόταση

P_1 AND P_2 AND … AND P_n => Q (5)

που σε Datalog εκφράζεται ως εξής:

Q:- P_1, P_2, …, P_n (6)

Η πρόταση Horn του (4) μπορεί να μετασχηματιστεί στην πρόταση

P_1 AND P_2 AND … AND P_n => (7)

που σε Datalog εκφράζεται ως εξής:

P_1, P_2, …, P_n (8)

Η έκφραση (6) είναι ένας κανόνας σε Datalog. Η έκφραση (8) είναι ένας περιορισμός αξιοπιστίας (όλα τα κατηγορήματα πρέπει να είναι αληθή για την ικανοποίηση της έκφρασης). Γενικά, μια ερώτηση σε Datalog αποτελείται από δύο στοιχεία:

- Ένα πρόγραμμα σε Datalog, το οποίο είναι ένα πεπερασμένο σύνολο από κανόνες
- Μια λέξη $P(X_1, X_2, .., X_n)$ όπου κάθε X_i είναι μια μεταβλητή ή μια σταθερά

9.3 ΕΡΜΗΝΕΙΑ ΤΩΝ ΚΑΝΟΝΩΝ

Υπάρχουν δύο βασικοί εναλλακτικοί τρόποι για την ερμηνεία των κανόνων: ο *βασισμένος στην απόδειξη (proof-theoretic)* και ο *βασισμένος στο μοντέλο (model-theoretic)*. Υπάρχουν πολλές Prolog μηχανές που ακολουθούν μία από τις δύο ερμηνείες, καθώς και άλλες που δεν ακολουθούν καμιά από αυτές (οπότε και λέμε ότι ακολουθούν την *υπολογιστική* ερμηνεία που εξειδικεύεται ανά μηχανή).

Στην *ερμηνεία που βασίζεται στην απόδειξη*, θεωρούμε τα γεγονότα και τους κανόνες ως αληθείς δηλώσεις, ή αλλιώς *αξιώματα.* Τα *βασικά αξιώματα (ground axioms)* (π.χ. τα γεγονότα) δεν περιέχουν μεταβλητές. Οι κανόνες ονομάζονται *επαγωγικά αξιώματα (deductive axioms)* και χρησιμοποιούνται για την εξαγωγή νέων γεγονότων. Στο σχήμα 9.2 παρουσιάζεται ο τρόπος με τον οποίο λειτουργεί η ερμηνεία που βασίζεται στην απόδειξη. Ο τρόπος που ακολουθείται υπολογιστικά προσομοιάζει με τον τρόπο που προγραμματίζουμε σε κάποια γλώσσα προγραμματισμού. Η διαδικασία, δε, για την απόδειξη της αλήθειας (ή όχι) μιας δήλωσης (θεωρήματος) είναι γνωστή και με το όνομα *theorem proving.*

Ο δεύτερος τρόπος ερμηνείας είναι αυτός που βασίζεται στην έννοια του *μοντέλου*. Εδώ, δεδομένου ενός πεπερασμένου (συνήθως) πεδίου σταθερών τιμών, αναθέτουμε σε κάθε κατηγόρημα όλους τους δυνατούς συνδυασμούς τιμών σαν ορίσματα. Στη συνέχεια πρέπει να αποφανθούμε αν το κατηγόρημα είναι αληθές ή ψευδές. Εν γένει, είναι αρκετό να βρούμε τους συνδυασμούς που κάνουν το κατηγόρημα αληθές και να θεωρήσουμε τους υπόλοιπους ως ψευδείς. Αν το κάνουμε αυτό για κάθε μέλος ενός συνόλου κατηγορημάτων, έχουμε μια *ερμηνεία (interpretation)* του εν λόγω συνόλου.

Δεδομένα
Ä1. επιβλέπει(ντόλυ, άβερελ)
Ä2. επιβλέπει(λούκυ, ντόλυ)
Κανόνες
Ê1. ανώτερος(X, Y) :- επιβλέπει (X, Y).
Ê2. ανώτερος(X, Y) :- επιβλέπει (X, Z), ανώτερος (Z, Y).
Απόδειξη του: ανώτερος(λούκυ, άβερελ)?
1. επιβλέπει(ντόλυ, άβερελ) – βασικό αξίωμα
2. επιβλέπει(λούκυ, ντόλυ) – βασικό αξίωμα
3. ανώτερος(ντόλυ, άβερελ) – εφαρμογή του K1 στο 1
4. ανώτερος(λούκυ, άβερελ) :- επιβλέπει(λούκυ, ντόλυ),
   ανώτερος(ντόλυ, άβερελ) – εφαρμογή του K2 στα 2, 3

Σχήμα 9.2 Απόδειξη θεωρήματος

Μια ερμηνεία καλείται *μοντέλο (model)* για ένα συγκεριμένο σύνολο κανόνων αν οι κανόνες αυτοί είναι πάντα αληθείς για τη συγκεκριμένη ερμηνεία. Με άλλα λόγια, έχουμε ένα μοντέλο, αν για κάθε συνδυασμό τιμών που κάνουν το δεξιό μέλος ενός κανόνα αληθές, και το αντίστοιχο αριστερό μέλος γίνεται αληθές. Ένας κανόνας παραβιάζεται αν μια συγκεκριμένη ανάθεση μεταβλητών στο δεξί μέλος δεν αντιστοιχεί στην ίδια αληθοτιμή για το αριστερό μέλος. Για παράδειγμα, αν

```
επιβλέπει(ντόλυ, άβερελ) αληθές
επιβλέπει(λούκυ, ντόλυ) αληθές
επιβλέπει(λούκυ, άβερελ) ψευδές
```

κάτω από μια συγκεκριμένη ερμηνεία, τότε η εν λόγω ερμηνεία δεν μπορεί να αποτελεί μοντέλο για τον κανόνα

```
ανώτερος(X, Y) :- επιβλέπει (X, Z), ανώτερος (Z, Y).
```

Στην ερμηνεία που βασίζεται σε κανόνες, η ερμηνεία των κανόνων ορίζεται με τον καθορισμό ενός μοντέλου για τους κανόνες αυτούς. Μπορούμε να θεωρήσουμε τότε, την κεφαλή ενός κανόνα σαν μια ερώτηση που ορίζεται από το σώμα του κανόνα και έχει ένα σύνολο από τιμές σαν αποτέλεσμα.

Ένα μοντέλο θεωρείται *ελάχιστο (minimal model)* για ένα σύνολο κανόνων αν δεν μπορούμε να αλλάξουμε μια αληθοτιμή και να εξακολουθούμε να έχουμε το ίδιο μοντέλο για τους κανόνες αυτούς. Ας υποθέσουμε ότι στα γεγονότα του Σχήματος 9.2 προσθέτουμε και το

```
ανώτερος(λούκυ, τοτός) (αληθές)
```

Η ερμηνεία είναι ακόμα μοντέλο, αλλά δεν είναι ελάχιστο, καθώς αν αλλάξουμε την αληθοτιμή για τη σχέση του λούκυ με τον τοτό, θα εξακολουθήσουμε να έχουμε το ίδιο μοντέλο.

9.4 ΒΑΣΙΚΟΙ ΜΗΧΑΝΙΣΜΟΙ ΕΞΑΓΩΓΗΣ ΣΥΜΠΕΡΑΣΜΑΤΩΝ ΓΙΑ ΛΟΓΙΚΑ ΠΡΟΓΡΑΜΜΑΤΑ

Υπάρχουν δύο βασικές στρατηγικές για την εξαγωγή συμπερασμάτων: η *bottom-up* και η *top-down*.

9.4.1 Bottom-up εξαγωγή συμπερασμάτων

Στην bottom-up εξαγωγή συμπερασμάτων (η οποία αποκαλείται και *forward chaining*) η μηχανή εξαγωγής συμπερασμάτων αρχίζει από τα βασικά γεγονότα και τους εφαρμόζει κανόνες. Καθώς νέα γεγονότα γεννιώνται, ελέγχονται σε σχέση με την αρχική ερώτηση. Ο όρος forward chaining έχει να κάνει με το ότι η διαδικασία κινείται από τα γεγονότα προς την ερώτηση. Ας υποθέσουμε ότι έχουμε τα δεδομένα και τους κανόνες του σχήματος 9.3 και την ερώτηση `ανώτερος(λούκυ, Υ)?` Δεδομένης της απουσίας δεδομένων για το κατηγόρημα *ανώτερος*, εφαρμόζουμε τον πρώτο κανόνα και το αποτέλεσμα είναι τα δεδομένα του πρώτου κύκλου εξαγωγής συμπερασμάτων. Μόλις παραχθεί το γεγονός `ανώτερος(λούκυ,τζόε)`, έχουμε την πρώτη απάντηση στην ερώτηση. Αντίστοιχα προκύπτει και το γεγονός `ανώτερος(λούκυ,ντόλυ)`. Στο τέλος αυτού του κύκλου, εφαρμόζεται ο επόμενος κανόνας για την εξαγωγή περαιτέρω γεγονότων. Πρέπει πλέον να εξεταστούν ζεύγη γεγονότων και για την ακρίβεια, κάθε γεγονός τύπου *επιβλέπει* με κάθε γεγονός τύπου *ανώτερος*. Αυτό παράγει και τα δεδομένα του δεύτερου κύκλου εξαγωγής συμπερασμάτων. Ο αλγόριθμος, στη συνέχεια, θα προσπαθήσει να εξάγει και περαιτέρω πληροφορία με βάση τα άρτι παραχθέντα γεγονότα, πλην όμως, δε θα προκύψουν νέα αποτελέσματα και η εξαγωγή συμπερασμάτων σταματά.

Είναι σαφές ότι η εξαντλητική αναζήτηση όλων των πιθανών συνδυασμών είναι αργή - γιαυτό και υπάρχουν τεχνικές βελτιστοποίησης στην bottom-up στρατηγική, τις οποίες όμως δε θα καλύψουμε εδώ.

9.4.2 Top-down εξαγωγή συμπερασμάτων

Η top-down εξαγωγή συμπερασμάτων είναι αυτή που χρησιμοποιείται και από τους intepreters της Prolog. Η top-down εξαγωγή συμπερασμάτων ονομάζεται και *backward chaining* καθώς η διαδικασία αποτίμησης μιας ερώτησης ξεκινά από την ερώτηση προς την αναζήτηση των κατάλληλων δεδομένων.

Στην προσέγγιση αυτή, δεν παράγουμε νέα δεδομένα. Ας πάρουμε για παράδειγμα, την ίδια ερώτηση με πριν: `ανώτερος(λούκυ, Υ)?` Το σύστημα ελέγχει κατ' αρχήν αν υπάρχουν δεδομένα που να ικανοποιούν άμεσα την ερώτηση: δεδομένα, δηλαδή, που το πρώτο ορισμά τους να έχει την τιμή "λούκυ". Αφού δεν υπάρχουν τέτοια δεδομένα, *το σύστημα ψάχνει για τον πρώτο κανόνα, του οποίου η κεφαλή έχει το ίδιο κατηγόρημα με την ερώτηση*.

Έτσι, παίρνουμε τον κανόνα `ανώτερος(Χ, Υ) :- επιβλέπει (Χ, Υ)` και αντιστοιχίζουμε τη μεταβλητή Χ στην τιμή "λούκυ", παράγοντας τον κανόνα

```
ανώτερος(λούκυ, Υ) :- επιβλέπει (λούκυ, Υ)
```

Η μεταβλητή Χ λέμε ότι έχει *δεσμευθεί (bound)* στην τιμή "λούκυ". Το σύστημα προχωρά στο να αντικαταστήσει την ερώτηση `ανώτερος(λούκυ, Υ)` με τη ερώτηση `επιβλέπει (λούκυ, Υ)` και ψάχνει να βρει γεγονότα που να ικανοποιούν την εν λόγω ερώτηση. Έτσι, προκύπτουν τα δεδομένα "τζόε" και "ντόλυ" σαν απαντήσεις για τη μεταβλητή Υ. Όταν εξαντληθούν όλα τα δεδομένα, το σύστημα ψάχνει για τον επόμενο κανόνα που έχει στο αριστερό μέρος το κατηγόρημα της ερώτησης.

Λαμβάνοντας τον κανόνα `ανώτερος(X, Y) :- επιβλέπει (X, Z), ανώτερος (Z, Y)` και αντιστοιχίζοντας τη μεταβλητή X στην τιμή "λούκυ", παράγουμε τον κανόνα

```
ανώτερος(λούκυ, Y) :- επιβλέπει (λούκυ, Z), ανώτερος (Z, Y)
```

Στη συνέχεια, ασχολούμαστε ξανά με το δεξί μέλος του κανόνα αυτού και ψάχνουμε για γεγονότα που ικανοποιούν και τα δύο κατηγορήματα του κανόνα. Τα δύο κατηγορήματα ονομάζονται πλέον *υπο-στόχοι (subgoals)* της ερώτησης ενώ ο συνδυασμός τους ονομάζεται *ενιαίος στόχος (compound goal)*. Για την αποτίμηση του ενιαίου στόχου χρησιμοποιούμε μια τακτική depth-first: το σύστημα προσπαθεί πρώτα να δεσμεύσει μια τιμή στη μεταβλητή Z που να κάνει τον πρώτο υπο-στόχο αληθή και στη συνέχεια να βρει γεγονότα των οποίων οι τιμές κάνουν και το δεύτερο υπο-στόχο αληθή (υπενθυμίζουμε ότι ψάχνουμε για τις σωστές τιμές του Y). Αν η τιμή που δεσμεύσαμε στο Z δεν βγάλει αποτελέσματα, ψάχνουμε για την επόμενη πιθανή τιμή.

```
Κανόνες
ανώτερος(X, Y) :- επιβλέπει (X, Y).
ανώτερος(X, Y) :- επιβλέπει (X, Z), ανώτερος (Z, Y).
κατώτερος(X, Y) :- ανώτερος (Y, X).
Δεδομένα
επιβλέπει(τζόε,τζακ).
επιβλέπει(τζόε,ουίλιαμ).
επιβλέπει(τζόε,μαμά_ντάλτον).
επιβλέπει(ντόλυ,ραν-ταν-πλαν).
επιβλέπει(ντόλυ,άβερελ).
επιβλέπει(λούκυ,τζόε).
επιβλέπει(λούκυ,ντόλυ).
Δεδομένα πρώτου κύκλου εξαγωγής συμπερασμάτων
ανώτερος(τζόε,τζακ).
ανώτερος(τζόε,ουίλιαμ).
ανώτερος(τζόε,μαμά_ντάλτον).
ανώτερος(ντόλυ,ραν-ταν-πλαν).
ανώτερος(ντόλυ,άβερελ).
ανώτερος(λούκυ,τζόε).
ανώτερος(λούκυ,ντόλυ).
Δεδομένα πρώτου κύκλου εξαγωγής συμπερασμάτων
ανώτερος(λούκυ,τζακ).
ανώτερος(λούκυ,ουίλιαμ).
ανώτερος(λούκυ,μαμά_ντάλτον).
ανώτερος(λούκυ,ραν-ταν-πλαν).
ανώτερος(λούκυ,άβερελ).
```

Σχήμα 9.3 Bottom-up εξαγωγή συμπερασμάτων

Στο παράδειγμά μας, έστω ότι το σύστημα δεσμεύει τη μεταβλητή Z στην τιμή "τζόε". Έπειτα ψάχνει για τιμές του Y που να κάνουν αληθές το `ανώτερος(τζόε, Y)`. Οι

αντίστοιχες τιμές είναι "ουίλιαμ", "τζακ", "μαμά_ντάλτον". Στη συνέχεια, το σύστημα "επιστρέφει" και δεσμεύει μια άλλη τιμή για το Z (π.χ. "ντόλυ") κ.ο.κ.

Η τεχνική της top-down εξαγωγής συμπερασμάτων έχει και αυτή περιθώρια βελτιστοποίησης (π.χ. χρησιμοποίηση breadth-first τεχνικής). Ένα βασικό μειονέκτημα της top-down τεχνικής είναι ότι εξαρτάται σημαντικά από την σειρά με την οποία θα χρησιμοποιηθούν οι κανόνες. Επιπλέον, και η συγγραφή των κανόνων είναι παράγοντας που επηρεάζει την αποτίμηση των κανόνων (καθώς μπορεί να οδηγήσει ως και σε ατέρμονο κύκλο σε κάποιους αναδρομικούς κανόνες).

9.5 Προγράμματα Datalog και η αποτίμησή τους

Υπάρχουν δύο κύριες μέθοδοι για να οριστούν οι τιμές αληθείας των κατηγορημάτων στα προγράμματα Datalog. Τα *κατηγορήματα-γεγονότα (fact-defined predicates)* και τα *κατηγορήματα-κανόνες (rule-defined predicates)*.

```
υπάλληλος(τζακ).
υπάλληλος(τζόε).
υπάλληλος(ραντανπλαν).
υπάλληλος(ντόλυ).
υπάλληλος(ουίλιαμ).
υπάλληλος(μαμά_ντάλτον).
υπάλληλος(άβερελ).
υπάλληλος(λούκυ).
μισθός(τζακ,30000).
μισθός(τζόε,40000).
μισθός(ραντανπλαν,25000).
μισθός(ντόλυ,43000).
μισθός(ουίλιαμ,38000).
μισθός(μαμά_ντάλτον,25000).
μισθός(άβερελ,25000).
μισθός(λούκυ,55000).
τομέας(τζακ,έρευνα).
τομέας(τζόε,έρευνα).
τομέας(ραντανπλαν,διοίκηση).
τομέας(ντόλυ,διοίκηση).
τομέας(ουίλιαμ,έρευνα).
τομέας(μαμά_ντάλτον,έρευνα).
τομέας(άβερελ,διοίκηση).
τομέας(λούκυ,κεντρικά).
επιβλέπει(τζόε,τζακ).
επιβλέπει(τζόε,ουίλιαμ).
επιβλέπει(τζόε,μαμά_ντάλτον.
επιβλέπει(ντόλυ,ραντανπλαν).
επιβλέπει(ντόλυ,άβερελ).
Επιβλέπει(λούκυ,τζόε).
Επιβλέπει(λούκυ,ντόλυ).
Άνδρας(τζακ).
άνδρας(τζόε).
Άνδρας(ουίλιαμ).
άνδρας(άβερελ).
άνδρας(λούκυ).
άνδρας(ραντανπλαν).
γυναίκα(ντόλυ).
γυναίκα(μαμά_ντάλτον).
έργο(τράπεζα).
έργο(τρένο).
έργο(άμαξα).
δουλεύει_σε(τζακ,τράπεζα,32).
δουλεύει_σε(τζακ,άμαξα,8).
δουλεύει_σε(ουίλιαμ,τρένο,40).
δουλεύει_σε(μαμά_ντάλτον,τράπεζα,
20).
δουλεύει_σε(τζόε,άμαξα,10).
δουλεύει_σε(τζόε,τράπεζα,10).
δουλεύει_σε(τζόε,τρένο,10).
δουλεύει_σε(ραντανπλαν,άμαξα,30).
δουλεύει_σε(ραντανπλαν,τρένο,10).
δουλεύει_σε(άβερελ,τράπεζα,35).
δουλεύει_σε(άβερελ,άμαξα,5).
δουλεύει_σε(ντόλυ,άμαξα,20).
δουλεύει_σε(ντόλυ,τρένο,15).
Δουλεύει_σε(λούκυ,τρένο,10).
```

Σχήμα 9.4 Κατηγορήματα-γεγονότα που περιγράφουν μια βάση δεδομένων

Τα *κατηγορήματα-γεγονότα (fact-defined predicates)* -ή αλλιώς *σχέσεις*- ορίζονται σαν ο κατάλογος όλων των συνδυασμών των τιμών (ή αλλιώς σαν ο κατάλογος όλων των πλειάδων)

που κάνουν το κατηγόρημα αληθές. Τα κατηγορήματα-γεγονότα αντιστοιχούν σε βασικές σχέσεις των οποίων τα περιεχόμενα είναι αποθηκευμένα σε ένα Σύστημα Βάσης Δεδομένων. Το σχήμα 9.4 δείχνει τα κατηγορήματα-γεγονότα *υπάλληλος, άνδρας, γυναίκα, τομέας, επιβλέπει, έργο*, και *δουλεύει_σε*, τα οποία είναι φανερό ότι ορίζουν δομές ανάλογες των σχέσεων στις σχεσιακές βάσεις δεδομένων. Τα *κατηγορήματα-κανόνες (rule-defined predicates)* -ή αλλιώς *όψεις*- ορίζονται σαν η κεφαλή ενός ή περισσοτέρων κανόνων Datalog. Αντιστοιχούν σε *υπερβατικές σχέσεις (virtual relations)* των οποίων τα περιεχόμενα μπορούν να εξαχθούν από την μηχανή εξαγωγής συμπερασμάτων. Το σχήμα 9.5 δείχνει κάποια κατηγορήματα-κανόνες που αναφέρονται στην βάση του σχήματος 9.4.

```
ανώτερος(X,Y) :- επιβλέπει(X,Y).
ανώτερος(X,Y) :- επιβλέπει(X,Z), ανώτερος(Z,Y).
κατώτερος(X,Y) :- ανώτερος(Y,X).
επιβλέπων(X) :- υπάλληλος(X), επιβλέπει(X,Y).
πάνω_40Κ_υπάλληλος(X) :- υπάλληλος(X), μισθός(X,Y), Y >= 40000.
κάτω_40Κ_επιβλέπων(X) :- επιβλέπων(X),
                          not(πάνω_40Κ_υπάλληλος(X)).
κύριο_προιόνχ_υπαλ(X) :- υπάλληλος(X),
                          δουλεύει_σε(X,προιόνχ,Y), Y >= 20.
πρόεδρος(X) :- υπάλληλος(X), not(επιβλέπει(Y,X)).
```

Σχήμα* 9.5 *κατηγορήματα-κανόνες

9.5.1 Ασφάλεια των προγραμμάτων

Λέμε ότι ένα πρόγραμμα ή ένας κανόνας είναι *ασφαλής* (safe) εάν γεννάει ένα *πεπερασμένο* σύνολο γεγονότων. Το γενικό θεωρητικό πρόβλημα του να αποφανθούμε για το εάν ένα σύνολο κανόνων είναι ασφαλές, είναι μη-αποφασιστέο (undecidable). Μπορούμε πάντως να αποφανθούμε για την ασφάλεια μίας *περιορισμένης μορφής* των κανόνων. Για παράδειγμα, οι κανόνες του σχήματος 9.4 είναι ασφαλείς. Μία περίπτωση όπου έχουμε μη-ασφαλείς κανόνες, που μπορούν να γεννήσουν άπειρο αριθμό γεγονότων, συμβαίνει όταν μία από τις μεταβλητές του κανόνα παίρνει τιμές μέσα από ένα άπειρο πεδίο τιμών, και επιπλέον για την μεταβλητή αυτή δεν ισχύει κάποιος άλλος περιορισμός ώστε να αναγκαστεί να κυμανθεί μέσα σε μία περιορισμένη σχέση. Για παράδειγμα, θεωρήστε τον κανόνα:

```
μεγάλος_μισθός(Y) :- Y > 60000
```

Εδώ είναι δυνατόν να πάρουμε άπειρο αριθμό αποτελεσμάτων εάν το Υ κινείται πάνω σε όλους τους δυνατούς ακέραιους. Υποθέστε όμως ότι αλλάζουμε τον κανόνα ως εξής:

```
μεγάλος_μισθός(Y) :- υπάλληλος(X),μισθός(X,Y),Y > 60000
```

Στον δεύτερο κανόνα το αποτέλεσμα δεν είναι άπειρο, γιατί οι τιμές που μπορεί να πάρει το Υ είναι πια περιορισμένες σε αυτές που αντιστοιχούν στον μισθό κάποιου υπαλλήλου στην βάση δεδομένων - και που τελικά αποτελούν ένα πεπερασμένο σύνολο τιμών. Μπορούμε επίσης να ξαναγράψουμε τον κανόνα ως εξής:

```
μεγάλος_μισθός(Y) :- Y > 60000,υπάλληλος(X), μισθός(X,Y)
```

Στην περίπτωση αυτή ο κανόνας εξακολουθεί να είναι ασφαλής, από θεωρητικής πλευράς. Προσέξτε όμως, ότι στην Prolog ή και σε οποιοδήποτε άλλο σύστημα που υιοθετεί τον μηχανισμό επαγωγής από-πάνω-προς-τα-κάτω, με διάσχιση-με-προτεραιότητα-βάθους (top-down, depth-first), ο κανόνας οδηγεί σε άπειρες επαναλήψεις (infinite loop). Αυτό συμβαίνει επειδή πρώτα δίνουμε μια τιμή στο Υ και μετά ελέγχουμε εάν η τιμή αυτή αντιστοιχεί στον μισθό κάποιου υπαλλήλου. Το αποτέλεσμα είναι η γέννηση απείρου αριθμού τιμών για το Υ, ακόμη κι αν αυτές από ένα σημείο και μετά δεν έχουν πιθανότητα να επαληθεύσουν την κεφαλή του κανόνα. Ένας ορισμός της Datalog θεωρεί και τους δύο κανόνες σαν ασφαλείς, επειδή έτσι ο ορισμός γίνεται ανεξάρτητος από την επιλογή της μηχανής επαγωγής. Παρόλα αυτά, είναι καλύτερα οι κανόνες να γράφονται με την ασφαλέστερη δυνατή μορφή τους, δηλαδή με τα κατηγορήματα που περιορίζουν τις πιθανές συνδέσεις (bindings) μεταβλητών να είναι πρώτα. Σαν άλλο ένα παράδειγμα μη-ασφαλούς κανόνα, θεωρήστε τον παρακάτω κανόνα:

```
έχει_κάτι(X,Y) :- υπάλληλος(X)
```

Εδώ, μπορεί και πάλι να γεννηθεί ένας άπειρος αριθμός Υ τιμών, επειδή η μεταβλητή Υ εμφανίζεται μόνο στην κεφαλή του κανόνα και έτσι δεν είναι περιορισμένη μέσα σε ένα πεπερασμένο σύνολο τιμών. Για να καταλήξουμε σε έναν πιο αυστηρό ορισμό για τους ασφαλής κανόνες, χρησιμοποιούμε την έννοια της *περιορισμένης μεταβλητής*.

```
σ_ένα(Α,Β,Γ).
σ_δύο(Δ,Ε,Ζ).
σ_τρία(Η,Θ,Ι,Κ).
επιλογή_ένα_Α_ίσο_γ(X,Y,Z) :- σ_ένα(γ,Y,Z).
επιλογή_ένα_Β_μικρ_5(X,Y,Z) :- σ_ένα(X,Y,Z), Y<5.
επιλογή_ένα_Α_ίσο_γ_και_Β_μικρ_5(X,Y,Z) :- σ_ένα(γ,Y,Z), Y<5.
επιλογή_ένα_Α_ισο_γ_ή_Β_μικρ_5(X,Y,Z) :- σ_ένα(γ,Y,Z).
επιλογή_ένα_Α_ισο_γ_ή_Β_μικρ_5(X,Y,Z) :- σ_ένα(X,Y,Z), Y<5.
προβολή_τρία_σε_Η_Θ(Σ,X) :- σ_τρία(Σ,X,Y,Z).
ένωση_ένα_δύο(X,Y,Z) :- σ_ένα(X,Y,Z).
ένωση_ένα_δύο(X,Y,Z) :- σ_δύο(X,Y,Z).
τομή_ένα_δύο(X,Y,Z) :- σ_ένα(X,Y,Z), σ_δύο(X,Y,Z).
διαφορά_δύο_ένα(X,Y,Z) :- σ_δύο(X,Y,Z), not(σ_ένα(X,Y,Z)).
καρτ_γινόμ_ένα_τρία(Τ,Θ,Ω,Σ,X,Y,Z) :- σ_ένα(Τ,Θ,Ω),
        σ_τρία(Σ,X,Y,Z).
φυσική_ένωση_ένα_τρία_Γ_ίσο_Η(Θ,Ω,Σ,X,Y,Z) :- σ_ένα(Θ,Ω,Σ),
        σ_τρία(Σ,X,Y,Z).
```

Σχήμα* 9.6 *Κατηγορήματα για την έκφραση σχεσιακών πράξεων.

Μια μεταβλητή X σε έναν κανόνα είναι *περιορισμένη* εάν:

(α) εμφανίζεται σε ένα κανονικό (όχι ενσωματωμένο) κατηγόρημα στο σώμα του κανόνα,

(β) εμφανίζεται σε ένα κατηγόρημα της μορφής X=c ή c=X ή (c1<=X και X<=c2) στο σώμα του κανόνα , όπου c, c1 και c2 είναι σταθερές τιμές, ή

(γ) εμφανίζεται σε ένα κατηγόρημα της μορφής X=Y ή Y=X στο σώμα του κανόνα, όπου Y είναι περιορισμένη μεταβλητή.

Ένας κανόνας είναι *ασφαλής* εάν όλες του οι μεταβλητές είναι περιορισμένες.

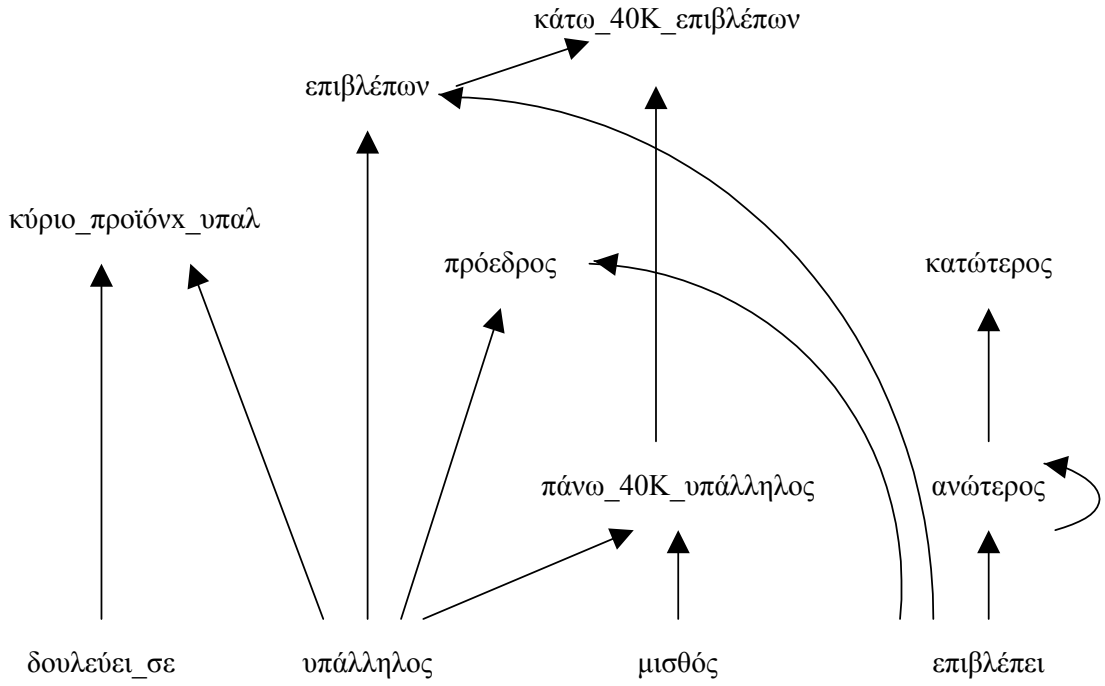
9.5.2 Χρήση των σχεσιακών πράξεων

Οι πράξεις της σχεσιακής άλγεβρας αντιστοιχίζονται εύκολα σε κανόνες Datalog, οι οποίοι ορίζουν το αποτέλεσμα της εφαρμογής των πράξεων αυτών πάνω στις σχέσεις (κατηγορήματα-γεγονότα) της βάσης δεδομένων. Αυτό σημαίνει ότι οι σχεσιακές ερωτήσεις (queries) και οι όψεις που επίσης βασίζονται σε ερωτήσεις μπορούν εύκολα να οριστούν στην Datalog. Η επιπλέον δύναμη που προσφέρει η Datalog βρίσκεται στην προδιαγραφή αναδρομικών (recursive) ερωτήσεων, καθώς και όψεων που βασίζονται σε αναδρομικές ερωτήσεις. Στην ενότητα αυτή, αναφερόμαστε στο πώς μερικές συνηθισμένες σχεσιακές πράξεις μπορούν να εκφραστούν σαν κανόνες Datalog. Ας υποθέσουμε ότι δίνονται οι βασικές σχέσεις (κατηγορήματα-γεγονότα) *σ_ένα, σ_δύο, σ_τρία*, των οποίων το σχήμα αναγράφεται στο σχήμα 9.6, και ότι οι σχεσιακές πράξεις εφαρμόζονται πάνω στις σχέσεις αυτές. Προσέξτε ότι στην Datalog δεν χρειάζεται να καθορίσουμε τα ονόματα των χαρακτηριστικών στο σχήμα 9.6, επειδή αυτά υπονοούνται απο το arity και την διάταξη κάθε κατηγορήματος.Σε ένα πραγματικό σύστημα, ο τύπος δεδομένων κάθε χαρακτηριστικού είναι επίσης σημαντικός για πράξεις όπως η *ένωση (union),* η *τομή (intersection)*, και η *σύνδεση (join),* και έτσι θεωρούμε ότι οι τύποι των χαρακτηριστικών είναι συμβατοί ως προς τις διάφορες πράξεις.

Το σχήμα 9.6 παραθέτει μερικές βασικές σχεσιακές πράξεις. Παρατηρήστε ότι, εάν το μοντέλο Datalog βασιστεί στο σχεσιακό μοντέλο και ως εκ τούτου θεωρήσει ότι τα κατηγορήματα (σχέσεις γεγονότων και αποτελέσματα ερωτήσεων) ορίζουν σύνολα πλειάδων, οι διπλές πλειάδες για το ίδιο κατηγόρημα εξαλείφονται αυτόματα. Στην πραγματικότητα αυτό μπορεί να ισχύει ή μπορεί και όχι, ανάλογα με τον μηχανισμό επαγωγής της Datalog. Σε κάθε περίπτωση, *δεν* ισχύει στην Prolog, κι έτσι καθένας από τους κανόνες του σχήματος 9.6 πού υποκρύπτει εξάλειψη διπλότυπου δεν είναι σωστός στην Prolog. Για παράδειγμα, αν θέλουμε να ορίσουμε κανόνες Prolog για την πράξη της ένωσης με εξάλειψη διπλοτύπων, θα πρέπει να τους ξαναγράψουμε ως εξής:

```
ένωση_ένα_δύο(X,Y,Z) :- σ_ένα(X,Y,Z).
ένωση_ένα_δύο(X,Y,Z) :- σ_δύο(X,Y,Z), όχι(σ_ένα(X,Y,Z)).
```

Πάντως, οι κανόνες του σχήματος 9.5 θα δουλέψουν στην Datalog, αν οι διπλότυπες πλειάδες εξαλείφονται αυτόματα. Ομοίως, οι κανόνες για την πράξη της προβολής (projection) στο σχήμα 9.6 θα δουλέψουν στην Datalog εφόσον ισχύει η ίδια συνθήκη, αλλά δεν θα δουλέψουν στην Prolog, μια και στην περίπτωση αυτή θα εμφανιστούν διπλότυπα.



Σχήμα 9.7 ***Γράφος εξάρτησης κατηγορημάτων για το σχήμα 9.6***

9.5.3 Αποτίμηση μη αναδρομικών ερωτήσεων Datalog

Οι υλοποιήσεις της Prolog έχουν βασιστεί στην προσέγγιση του *backward chaining*, όπου η σειρά διάταξης των κατηγορήματα είναι ουσιώδης. Μια και η Datalog έχει οριστεί σαν ένα υποσύνολο της Prolog, οι επαγωγικοί μηχανισμοί των γλωσσών λογικού προγραμματισμού, όπως το forward chaining ή το backward chaining, μπορούν να χρησιμοποιηθούν και για την Datalog. Παρόλα αυτά, για να μπορεί η Datalog να χρησιμοποιηθεί σε συστήματα επαγωγικής βάσης δεδομένων, είναι σκόπιμο να βασιστεί ο επαγωγικός της μηχανισμός πάνω στις έννοιες των σχεσιακών συστημάτων για την επεξεργασία ερωτήσεων. Στην σχεσιακή επεξεργασία ερωτήσεων, η τακτική συνίσταται σε από κάτω-προς-τα-πάνω αποτίμηση, αρχίζοντας από τις βασικές σχέσεις. Η σειρά των πράξεων διατηρεί μια ευελιξία που της επιτρέπει να γίνεται αντικείμενο βελτιστοποιήσεων (query optimization). Σε αυτή την ενότητα παρουσιάζουμε έναν επαγωγικό μηχανισμό βασισμένο στις σχεσιακές πράξεις που μπορεί να εφαρμοστεί σε *μη-αναδρομικές* ερωτήσεις Datalog. Για τα παραδείγματα χρησιμοποιούμε την βάση γεγονότων και κανόνων των σχημάτων 9.4 και 9.5.

Σε περίπτωση που μια ερώτηση αποτελείται αποκλειστικά από κατηγορήματα-γεγονότα, η επαγωγή εκφυλίζεται σε μια απλή αναζήτηση ανάμεσα στα γεγονότα για το αποτέλεσμα της ερώτησης. Για παράδειγμα, μια ερώτηση όπως η

```
τομέας(X, έρευνα)?
```

είναι μια επιλογή των ονομάτων όλων των υπαλλήλων X που εργάζονται στον τομέα "έρευνα", και μπορεί να απαντηθεί διατρέχοντας τα κατηγορήματα-γεγονότα `τομέας(X,Y)`. Τέτοιες ερωτήσεις είναι παρόμοιες με την σχεσιακή πράξη της επιλογής (select) πάνω σε μια

βασική σχέση, και μπορούν να γίνουν αντικείμενο εφαρμογής τεχνικών βελτιστοποίησης και επεξεργασίας ερωτήσεων.

Όταν μια ερώτηση συμπεριλαμβάνει κατηγορήματα-κανόνες, ο μηχανισμός επαγωγής πρέπει να υπολογίσει το αποτέλεσμα με βάση τους ορισμούς των κανόνων. Αν η ερώτηση είναι μη-αναδρομική και περιέχει κάποιο κατηγόρημα p που εμφανίζεται σαν η κεφαλή του κανόνα p :- p_1, p_2, ..., p_n, η στρατηγική είναι να υπολογιστούν πρώτα οι σχέσεις που αντιστοιχούν στα p_1, p_2, ..., p_n και έπειτα να υπολογιστεί η σχέση που αντιστοιχεί στο p, που μπορεί να χρησιμοποιηθεί για την δημιουργία του αποτελέσματος της ερώτησης. Για να ανιχνευτούν οι σχέσεις ανάμεσα στα κατηγορήματα, αποτυπώνουμε τις αλληλεξαρτήσεις τους σε ένα *γράφο εξαρτήσεων κατηγορημάτων (predicate dependency graph)*. Το σχήμα 9.7 δείχνει τον γράφο εξαρτήσεων για τα γεγονότα και τους κανόνες των σχημάτων 9.5 και 9.6. Ο γράφος εξαρτήσεων περιέχει έναν κόμβο για κάθε κατηγόρημα. Όποτε ένα κατηγόρημα Α υπάρχει στο σώμα ενός κανόνα και η κεφαλή του κανόνα είναι το κατηγόρημα Β, λέμε ότι το Β *εξαρτάται από* το Α, και σχεδιάζουμε μια ακμή με κατεύθυνση από το Α προς το Β. Αυτό υποδεικνύει ότι για να υπολογίσουμε τα γεγονότα για το κατηγόρημα Β (την κεφαλή του κανόνα), πρέπει πρώτα να υπολογίσουμε τα γεγονότα για όλα τα κατηγορήματα Α στο σώμα του κανόνα. Αν ο γράφος εξαρτήσεων δεν παρουσιάζει κύκλους, καλούμε το σύνολο των κανόνων *μη-αναδρομικό* (nonrecursive). Εάν υπάρχει έστω και ένας κύκλος, το σύνολο κανόνων λέγεται *αναδρομικό* (recursive). Στο σχήμα 9.7 υπάρχει ένα αναδρομικά ορισμένο κατηγόρημα - το "ανώτερος" - που έχει μια αναδρομική ακμή που δείχνει στον εαυτό του, μια και ο αντίστοιχος κανόνας είναι ορισμένος αναδρομικά. Επιπλέον, επειδή το κατηγόρημα "κατώτερος" εξαρτάται από το "ανώτερος", απαιτεί κι αυτό αναδρομή για τον υπολογισμό του αποτελέσματός του.

Μια ερώτηση που περιλαμβάνει αποκλειστικά μη-αναδρομικά κατηγορήματα λέγεται *μη-αναδρομική ερώτηση*. Στην ενότητα αυτή παρουσιάζουμε μηχανισμούς επαγωγής *μόνο* για μη-αναδρομικές ερωτήσεις. Στο σχήμα 9.7 κάθε ερώτηση που δεν περιλαμβάνει τα κατηγορήματα *ανώτερος* ή *κατώτερος*, είναι μη-αναδρομική. Στον γράφο εξαρτήσεων, οι κόμβοι που αντιστοιχούν σε κατηγορήματα-γεγονότα δεν έχουν εισερχόμενες ακμές, μια και όλα τα κατηγορήματα-γεγονότα είναι αποθηκευμένα σαν σχέσεις στην βάση δεδομένων. Τα περιεχόμενα ενός κατηγορήματος-γεγονότος μπορούν να υπολογιστούν ανεξάρτητα από άλλα κατηγορήματα απλώς ανακαλώντας τις πλειάδες της αντίστοιχης σχέσης.

Η κύρια λειτουργία ενός μηχανισμού επαγωγής έγκειται στον υπολογισμό των γεγονότων που αντιστοιχούν σε κατηγορήματα ερωτήσεων. Αυτό μπορεί να επιτευχθεί κατασκευάζοντας μια *σχεσιακή έκφραση* αποτελούμενη από σχεσιακούς τελεστές όπως SELECT, PROJECT, JOIN, UNION, και SET DIFFERENCE (με κατάλληλες προφυλάξεις για την αντιμετώπιση ζητημάτων ασφάλειας), η οποία με την εκτέλεσή της δίνει το ζητούμενο αποτέλεσμα για την αρχική ερώτηση. Η ερώτηση μπορεί έτσι να εκτελεσθεί χρησιμοποιώντας τις εσωτερικές λειτουργίες για επεξεργασία ερωτήσεων και βελτιστοποιήσεις που διαθέτει ένα σχεσιακό σύστημα διαχείρισης βάσης δεδομένων. Όποτε ο μηχανισμός αναγωγής θέλει να υπολογίσει το σύνολο γεγονότων που αντιστοιχεί σε κάποιο μη-αναδρομικό κατηγόρημα-κανόνα p, εντοπίζει όλους τους κανόνες που έχουν σαν κεφαλή το p. Η ιδέα είναι να υπολογιστεί το σύνολο γεγονότων για κάθε τέτοιο κανόνα και έπειτα να πάρουμε την *ένωση* των αποτελεσμάτων, μια και η ένωση ισοδυναμεί με την λογική διάζευξη (OR). Ο γράφος εξαρτήσεων δείχνει όλα τα κατηγορήματα q από τα οποία εξαρτάται κάθε p, και αφού υποθέτουμε ότι το κατηγόρημα είναι μη-αναδρομικό, μπορούμε πάντα να βρούμε μια μερική διάταξη για τα κατηγορήματα q. Πριν υπολογίσουμε το σύνολο γεγονότων για το p, υπολογίζουμε τα σύνολα γεγονότων για όλα τα κατηγορήματα q από τα οποία εξαρτάται το p,

με βάση την μερική τους διάταξη. Για παράδειγμα, αν μια ερώτηση περιέχει το κατηγόρημα κάτω_40Κ_επιβλέπων, πρέπει πρώτα να υπολογίσουμε τα επιβλέπων και πάνω_40Κ_υπάλληλος. Επειδή αυτά τα δύο εξαρτώνται μόνο από τα κατηγορήματα-γεγονότα υπάλληλος, μισθός, και επιβλέπει, μπορούν να υπολογισθούν άμεσα από τις αποθηκευμένες σχέσεις της βάσης δεδομένων.

Θα πρέπει επίσης να ασχοληθούμε με τα ενσωματωμένα κατηγορήματα, όπως είναι οι τελεστές σύγκρισης >, <, και =, όταν εμφανίζονται στο σώμα κάποιου κανόνα. Όταν προδιαγράφεται ένας αλγόριθμος επαγωγής που υπολογίζει το σύνολο γεγονότων μιας οποιασδήποτε μη-αναδρομικής ερώτησης, είναι συνηθισμένο να μετατρέπονται οι κανόνες σε μια κανονική μορφή που λέγεται *τροποποιημένοι κανόνες (rectified rules)*. *Ένας κανόνας λέγεται τροποποιημένος εάν όλα τα ορίσματα (arguments) στο κατηγόρημα της κεφαλής του κανόνα είναι διαφορετικές μεταβλητές*. Αν η κεφαλή του κανόνα έχει σταθερές, ή αν μια μεταβλητή επαναλαμβάνεται δυο φορές στην κεφαλή του κανόνα, ο κανόνας μπορεί να τροποποιηθεί εύκολα: μια σταθερά c αντικαθιστάται από μια μεταβλητή X, και ένα κατηγόρημα equals(X,c) προστίθεται στο σώμα του κανόνα. Ομοίως, αν μια μεταβλητή Y εμφανίζεται δυο φορές στην κεφαλή του κανόνα, η μια εμφάνισή της αντικαθίσταται με μια άλλη μεταβλητή Z, και ένα κατηγόρημα equals(Y,Z) προστίθεται στο σώμα του κανόνα.

Προσέξτε ότι στην περίπτωση των μη-αναδρομικών ερωτήσεων η αποτίμηση των ερωτήσεων μπορεί να εκφραστεί σαν δένδρο του οποίου τα φύλλα αντιπροσωπεύουν τις βασικές σχέσεις. Αυτό που χρειάζεται είναι η κατάλληλη εφαρμογή των σχεσιακών τελεστών SELECT, PROJECT, και JOIN, μαζί με τις λειτουργίες συνόλων UNION και SET DIFFERENCE, μέχρι να υπολογιστεί το κατηγόρημα της ερώτησης. Ένα σκιαγράφημα του αλγορίθμου επαγωγής GET_EXPR(Q) που δημιουργεί μια σχεσιακή έκφραση για να υπολογίσει το αποτέλεσμα της ερώτησης DATALOG Q = p(arg1, arg2, ..., argN) δίνεται παρακάτω:

Αλγόριθμος επαγωγής για μη αναδρομικές ερωτήσεις.

1. Εντοπισμός όλων των κανόνων S των οποίων η κεφαλή περιέχει το κατηγόρημα p. Εάν δεν υπάρχουν τέτοιοι κανόνες τότε το p είναι κατηγόρημα-γεγονός και αντιστοιχεί σε κάποια σχέση Rp της βάσης δεδομένων. Στην περίπτωση αυτή, μια από τις παρακάτω εκφράσεις επιστρέφεται και ο αλγόριθμος τερματίζεται (χρησιμοποιούμε το σύμβολο $i για να αναφερθούμε στο όνομα του i-οστού χαρακτηριστικού της σχέσης Rp).

α. Αν όλα τα ορίσματα είναι μοναδικές μεταβλητές, η σχεσιακή έκφραση που επιστρέφεται είναι η Rp.

β. Αν κάποια ορίσματα είναι σταθερές ή αν η ίδια μεταβλητή εμφανίζεται σε περισσότερες από μία θέσεις ορισμάτων, η έκφραση που επιστρέφεται είναι:

SELECT<συνθήκη>(Rp),

όπου η επιλογή <συνθήκη> είναι μια συζευκτική συνθήκη αποτελούμενη από έναν αριθμό απλών συνθηκών που συνδέονται με AND και που κατασκευάζεται ως εξής:

i) Αν μια σταθερά c εμφανίζεται σαν όρισμα i, συμπεριέλαβε μια απλή συνθήκη ($i = c) στην σύζευξη.
ii) Αν η ίδια μεταβλητή εμφανίζεται στις θέσεις ορισμάτων j και k, συμπεριέλαβε μια συνθήκη ($j = $k) στην σύζευξη.

γ. Για ένα όρισμα που δεν υπάρχει σε κανένα κατηγόρημα, κατασκευάζεται μια μοναδιαία (unary) σχέση με τιμές που να ικανοποιούν όλες τις συνθήκες. Από την στιγμή που ο κανόνας θεωρείται ασφαλής, η μοναδιαία σχέση είναι πεπερασμένη.

2. Στην περίπτωση αυτή, υπάρχουν ένας ή περισσότεροι κανόνες Si, i = 1, 2, ..., n, n > 0, με το κατηγόρημα p στην κεφαλή τους. Για κάθε έναν τέτοιο κανόνα Si, κατασκεύασε μια σχεσιακή έκφραση ως εξής:

α. Εφάρμοσε πράξεις επιλογής στα κατηγορήματα του σώματος κάθε τέτοιου κανόνα, όπως αναφέρθηκε στο βήμα 1.

β. Μια φυσική σύνδεση (natural join) κατασκευάζεται μεταξύ των σχέσεων που αντιστοιχούν στα κατηγορήματα του σώματος του κανόνα Si, βασισμένη πάνω στις κοινές μεταβλητές. Για τα ορίσματα που στο βήμα 1(γ) δημιουργούνταν μοναδιαίες σχέσεις, οι αντίστοιχες σχέσεις εισάγονται σαν μέλη στην φυσική σύνδεση. Έστω ότι το αποτέλεσμα αυτής της σύνδεσης είναι η σχέση Rs.

γ. Εάν κάποιο ενσωματωμένο κατηγόρημα X theta Y έχει εφαρμοστεί πάνω στα ορίσματα X και Y, το αποτέλεσμα της σύνδεσης υπόκειται σε μια επιπλέον επιλογή.

SELECT <X theta Y> (Rs),

δ. Επανέλαβε το βήμα 2(β) μέχρι να μην εφαρμόζονται πλέον άλλα ενσωματωμένα κατηγορήματα.

3. Δημιούργησε την ένωση (UNION) των εκφράσεων που κατασκευάστηκαν στο βήμα 2 (αν υπάρχουν περισσότεροι από έναν κανόνες με το κατηγόρημα p στην κεφαλή τους).

9.5.4 Έννοιες για την αποτίμηση αναδρομικών ερωτήσεων στην Datalog

Σε γενικές γραμμές για τις ερωτήσεις σε επαγωγικές βάσεις δεδομένων, το πρόβλημα της βελτιστοποίησης ερωτήσεων μπορεί να χωριστεί σε δύο προβλήματα:

- Το πρόβλημα της αποτίμησης (evaluation) της ερώτησης, που τίθεται σε σχέση με ένα πρόγραμμα το οποίο παράγει μια απάντηση για την ερώτηση.
- Ένα τμήμα στρατηγικής που έρχεται στο προσκήνιο όταν η βέλτιστη εκτέλεση των κανόνων προϋποθέτει την επαναδιατύπωση των κανόνων.

Πολλές προσεγγίσεις έχουν προταθεί και για αναδρομικές και για μη-αναδρομικές ερωτήσεις. Νωρίτερα συζητήσαμε μια προσέγγιση για την αποτίμηση μη-αναδρομικών ερωτήσεων. Εδώ, πρώτα ορίζουμε κάποια ορολογία για τις αναδρομικές ερωτήσεις, και έπειτα

περιγράφουμε σύντομα απλές και ημι-απλές (semi-naive) προσεγγίσεις για την επεξεργασία ερωτήσεων που ανήκουν στην πρώτη κατηγορία των προβλημάτων που προαναφέρθησαν καθώς και την προσέγγιση του μαγικού συνόλου που ανήκει στην δεύτερη κατηγορία.

Έχουμε ήδη δεί παραδείγματα που αφορούν αναδρομικούς κανόνες όπου το ίδιο κατηγόρημα εμφανίζεται στην κεφαλή και στο σώμα του κανόνα. Άλλο ένα παράδειγμα είναι

```
πρόγονος(X,Y) :- πρόγονος(X,Z), γονέας(Z,Y)
```

το οποίο δηλώνει ότι ο Y είναι πρόγονος του X εάν ο Z είναι πρόγονος του X και ο Y είναι γονέας του Z.

Ένας κανόνας λέμε ότι είναι *γραμμικά αναδρομικός* (linearly recursive) αν το αναδρομικό κατηγόρημα (πρόγονος) εμφανίζεται μόνο μια φορά στο σώμα του κανόνα. Για παράδειγμα,

```
ιγ(X,Y) :- γονέας(X,XΓ), γονέας(Y,YΓ), ιγ(XΓ,YΓ)
```

είναι ένας γραμμικός κανόνας στον οποίο το κατηγόρημα *ιγ* (εξαδέλφια ίδιας γενεάς) χρησιμοποιείται μόνο μια φορά στο σώμα. Ο κανόνας δηλώνει ότι οι X και Y είναι εξαδέλφια ίδιας γενεάς εάν οι γονείς τους είναι εξαδέλφια ίδιας γενεάς. Παρατηρήστε ότι ο κανόνας

```
πρόγονος(X,Y) :- πρόγονος(X,Z), πρόγονος(Z,Y)
```

δεν είναι γραμμικά αναδρομικός. Πιστεύεται ότι οι περισσότεροι κανόνες που αντανακλούν την πραγματική ζωή μπορούν να περιγραφούν σαν γραμμικά αναδρομικοί κανόνες. Αλγόριθμοι που εκτελούν γραμμικά σύνολα κανόνων έχουν αναπτυχθεί. Οι προηγούμενοι ορισμοί γίνονται πιο πολύπλοκοι όταν θεωρήσουμε ένα σύνολο κανόνων με κατηγορήματα που παρουσιάζονται και στην κεφαλή και στο σώμα των κανόνων.

Δεδομένου ενός προγράμματος Datalog με σχέσεις που να αντιστοιχούν στα κατηγορήματα, το σύμβολο `:-` μπορεί να αντικατασταθεί από το σύμβολο της ισότητας χωρίς να αλλοιωθεί η σημασία, για να σχηματιστούν *εξισώσεις Datalog* (Datalog equations). *Σταθερό σημείο (fixed point)* ενός συνόλου εξισώσεων ως προς τις σχέσεις R_1, R_2, ..., R_n, ορίζεται σαν μια λύση για τις σχέσεις αυτές που επαληθεύει τις δοσμένες εξισώσεις. Ένα σταθερό σημείο λοιπόν αποτελεί μοντέλο των κανόνων από τους οποίους ορίστηκαν οι εξισώσεις. Είναι δυνατόν για κάποιο σύνολο εξισώσεων να υπάρχουν δύο σύνολα λύσεων, $S_1 <= S_2$, τέτοια ώστε κάθε κατηγόρημα στην λύση S_1 αν είναι υποσύνολο (ή ίσο) του αντίστοιχου κατηγορήματος στην λύση S_2. Μια λύση S_0 καλείται το *ελάχιστο σταθερό σημείο (least fixed point)* εάν $S_0 <= S$ για κάθε λύση S αυτών των εξισώσεων.

Αν συμβολίσουμε έναν κατευθυνόμενο γράφο με το κατηγόρημα `ακμή(X,Y)`, τέτοιο ώστε το `ακμή(X,Y)` να είναι αληθές εάν και μόνο εάν υπάρχει στον γράφο μια ακμή από τον κόμβο X στον κόμβο Y, τα μονοπάτια του γράφου μπορούν να εκφραστούν από τους παρακάτω κανόνες:

```
μονοπάτι(X,Y) :- ακμή(X,Y)
μονοπάτι(X,Y) :- μονοπάτι(X,Z), μονοπάτι(Z,Y)
```

Παρατηρήστε ότι υπάρχουν και άλλοι τρόποι να οριστούν μονοπάτια με αναδρομικό τρόπο. Ας υποθέσουμε ότι οι σχέσεις P και A αντιστοιχούν στα κατηγορήματα μονοπάτι και ακμή των

προηγούμενων κανόνων. *Το μεταβατικό κλείσιμο (transitive closure) της σχέσης P περιέχει όλα τα δυνατά ζευγάρια κόμβων που συνδέονται μέσω κάποιου μονοπατιού, και αντιστοιχεί στην λύση ελάχιστου σταθερού σημείου για τις εξισώσεις που πηγάζουν από τους παραπάνω κανόνες.*

Για την αποτίμηση ενός συνόλου κανόνων (εξισώσεων) Datalog που ίσως περιέχουν αναδρομικούς κανόνες έχουν προταθεί πολλές στρατηγικές, οι λεπτομέρειες των οποίων δεν θα μας απασχολήσουν εδώ. Θα αναφέρουμε τρεις σημαντικές τεχνικές: την *απλή (naive) στρατηγική*, την *ημι-απλή (semi-naive) στρατηγική*, και την χρησιμοποίηση των *μαγικών συνόλων (magic sets)*.

Απλή στρατηγική. Η είσοδος αποτελείται από ένα σύνολο κανόνων Datalog με ένα σύνολο βασικών σχέσεων που αντιστοιχούν στα κατηγορήματα-γεγονότα και άλλο ένα σύνολο σχέσεων που αντιστοιχούν στα κατηγορήματα-κανόνες. Η έξοδος είναι η λύση του ελάχιστου σταθερού σημείου για τις εξισώσεις Datalog που αντιστοιχούν στους κανόνες αυτούς.

Η διαδικασία αρχίζει με την κατασκευή των εξισώσεων για τους κανόνες. Χρησιμοποιούμε σταθερές τιμές για τις σχέσεις-γεγονότα και τις τρέχουσες τιμές για τις σχέσεις-κανόνες, για να υπολογίσουμε νέες τιμές για τις σχέσεις-κανόνες. Η φάση αυτή επαναλαμβάνεται μέχρι κάποιο σημείο όπου δεν παρατηρείται καμιά αλλαγή στις τιμές των κατηγορημάτων-κανόνων. Στην κατάσταση αυτή, υποτίθεται ότι έχουμε φθάσει στην λύση ελάχιστου σταθερού σημείου.

Ημι-απλή στρατηγική. Το κυρίως πρόβλημα με την απλή αποτίμηση είναι ότι σε κάθε επανάληψη της αποτίμησης των σχέσεων-κανόνων, συμβαίνει ένας άχρηστος υπολογισμός των ίδιων πλειάδων. Θα έπρεπε μάλλον να επικεντρωθούμε στην *αυξητική αλλαγή (incremental change)* των κατηγορημάτων-κανόνων σε κάθε γύρο, και να την χρησιμοποιούμε για τον υπολογισμό των νέων πλειάδων στον επόμενο γύρο. Η προσέγγιση της ημι-απλής στρατηγικής είναι αποδοτική επειδή υπολογίζει το "διαφορικό" των πλειάδων για κάθε κατηγόρημα-κανόνα σε κάθε επανάληψη.

Το δεύτερο πρόβλημα έχει να κάνει με την επεξεργασία ερωτήσεων, και αντιμετωπίζει τα παρακάτω αδύνατα σημεία των προηγούμενων προσεγγίσεων:

- Η απλή και η ημι-απλή στρατηγική *δεν* κάνουν καλή χρήση των δεσμεύσεων της ερώτησης (query bindings).
- Οι παραπάνω στρατηγικές εξακολουθούν να κάνουν πολλούς άχρηστους υπολογισμούς.

Πολλές τεχνικές προτάθηκαν, με στόχο να ξεπεραστούν αυτά τα μειονεκτήματα. Περιγράφουμε τώρα μια από αυτές.

Τεχνική μαγικού συνόλου για επανεγγραφή κανόνα. Το πρόβλημα που αντιμετωπίζει η τεχνική μαγικού συνόλου για επανεγγραφή κανόνα είναι ότι συχνά μια ερώτηση δεν ζητάει ολόκληρη την σχέση που αντιστοιχεί στο κατηγόρημα-κανόνα, αλλά μόνο ένα μικρό υποσύνολο αυτής της σχέσης. Θεωρήστε το παρακάτω πρόγραμμα:

```
ιγ(X,Y) :- επίπεδο(X,Y).
ιγ(X,Y) :- πάνω(X,Θ), ιγ(Θ,Ω), κάτω(Ω,Y).
```

Εδώ, *ιγ* είναι ένα κατηγόρημα (εξαδέλφια ίδιας γενεάς), και η κεφαλή καθενός από τους δυο κανόνες είναι το `ιγ(X,Y)`. Τα άλλα κατηγορήματα των κανόνων είναι τα επίπεδο, πάνω, και κάτω. Αυτά πιθανότατα βρίσκονται αποθηκευμένα αναγραφικά (extentionally) σαν γεγονότα, ενώ η σχέση για το *ιγ* είναι *περιγραφική (intentional)* - δηλαδή ορισμένη αποκλειστικά από τους

κανόνες. Για μια ερώτηση σαν `ιγ(τζακ,Ζ)` - δηλαδή "ποιοι είναι ίδιας γενεάς εξάδελφοι του Γιάννη;" - η απάντησή μας στην ερώτηση πρέπει να εξετάσει μόνο το κομμάτι της βάσης δεδομένων που είναι *σχετικό* - δηλαδή το μέρος που αφορά σε άτομα που κάπως συνδέονται με τον Γιάννη.

Μια αναζήτηση από-πάνω-προς-τα-κάτω ή backward chaining θα αρχίσει με την ερώτηση σαν στόχο (goal) και θα χρησιμοποιήσει τους κανόνες από την κεφαλή προς το σώμα για την δημιουργία περισσότερων στόχων. Κανένας από τους στόχους αυτούς δεν θα είναι άσχετος με την ερώτηση, παρά το γεγονός ότι μερικοί ίσως μας οδηγήσουν στην εξερεύνηση μονοπατιών που τυχαίνει να μην οδηγούν πουθενά.

Από την άλλη πλευρά, μια αναζήτηση από-κάτω-προς-τα-πάνω (bottom-up) ή forward chaining που να δουλεύει από τα σώματα των κανόνων προς τις κεφαλές, θα μας έκανε να εξάγουμε *ιγ* κανόνες που ποτέ ούτε καν θα απασχολούσαν την από-πάνω-προς-τα-κάτω αναζήτηση. Ακόμη, η αναζήτηση από-κάτω-προς-τα-πάνω είναι επιθυμητή επειδή αποφεύγει τα προβλήματα των επανειλλημένων υπολογισμών οι οποίοι είναι έμφυτοι στην από-πάνω-προς-τα-κάτω προσέγγιση. Επιπλέον, οι προσεγγίσεις από-κάτω-προς-τα-πάνω μας επιτρέπουν να χρησιμοποιούμε λειτουργίες συνόλων (set-at-a-time operations), όπως είναι οι σχεσιακές συνδέσεις, που είναι πολύ αποδοτικές όταν εφαρμόζονται σε δεδομένα που βρίσκονται στον δίσκο, ενώ από την άλλη οι καθαρές από-πάνω-προς-τα-κάτω μέθοδοι χρησιμοποιούν λειτουργίες πλειάδων (tuple-at-a-time).

Η επανεγγραφή κανόνα με μαγικά σύνολα είναι μια τεχνική που μας επιτρέπει να επαναδιατυπώνουμε τους κανόνες, σαν συνάρτηση αποκλειστικά της μορφής της ερώτησης, με άλλα λόγια, η τεχνική εξετάζει ποια ορίσματα του κατηγορήματος είναι δεσμευμένα σε σταθερές και ποια είναι μεταβαλλόμενα, έτσι ώστε να συνδυαστούν τα πλεονεκτήματα των μεθόδων από-πάνω-προς-τα-κάτω και από-κάτω-προς-τα-πάνω. Η τεχνική εστιάζει στον στόχο που είναι έμφυτος στην από-πάνω-προς-τα-κάτω προσέγγιση, αλλά συνδυάζει επίσης και την ευκολία της επανάληψης, τον έλεγχο τερματισμού και την αποδοτική αποτίμηση της από-κάτω-προς-τα-πάνω μεθόδου. Αντί να αναλύσουμε την μέθοδο, της οποίας είναι γνωστές και χρησιμοποιούνται πολλές παραλλαγές, θα εξηγήσουμε την ιδέα μέσα από ένα παράδειγμα.

Δεδομένων των προηγούμενων κανόνων και της ερώτησης `ιγ(τζακ,Ζ)`, ένας τυπικός μετασχηματισμός των κανόνων μέσω μαγικών συνόλων θα ήταν:

```
ιγ(Χ,Υ) :- μαγικό-ιγ(Χ), επίπεδο(Χ,Υ).
ιγ(Χ,Υ) :- μαγικό-ιγ(Χ), πάνω(Χ,Θ), ιγ(Θ,Ω), κάτω(Ω,Υ).
μαγικό-ιγ(Θ) :- μαγικό-ιγ(Χ), πάνω(Χ,Θ).
μαγικό-ιγ(τζακ).
```

Διαισθητικά μπορούμε να δούμε ότι τα γεγονότα του *μαγικό-ιγ* αντιστοιχούν σε ερωτήσεις ή υποστόχους. Ο ορισμός του κατηγορήματος *μαγικό-ιγ* μιμείται τον τρόπο που γεννούνται οι στόχοι κατά την αποτίμηση από-πάνω-προς-τα-κάτω. Το σύνολο γεγονότων του μαγικό-ιγ χρησιμοποιείται σαν φίλτρο στους κανόνες που ορίζουν το *ιγ*, για την αποφυγή γένεσης γεγονότων που δεν αποτελούν απαντήσεις κάποιου υποστόχου. Έτσι, μια αυστηρά από-κάτω-προς-τα-πάνω με forward chaining αποτίμηση του επαναδιατυπωμένου προγράμματος επιτυγχάνει έναν περιορισμό της αναζήτησης παρόμοιο με αυτόν που κατορθώνει η από-πάνω-προς-τα-κάτω αποτίμηση του αρχικού προγράμματος. Λεπτομέρειες αυτής της τεχνικής βρίσκονται πέρα από τον σκοπό μας.

Ενώ η τεχνική των μαγικών συνόλων αναπτύχθηκε αρχικά για την αντιμετώπιση των αναδρομικών ερωτήσεων, μπορεί να εφαρμοστεί επίσης και στις μη-αναδρομικές ερωτήσεις. Μάλιστα, έχει προσαρμοστεί για να χρησιμοποιείται σε ερωτήσεις SQL (που περιλαμβάνουν χαρακτηριστικά όπως ομαδοποίηση (grouping), πράξεις ομάδων (aggregation), αριθμητικές συνθήκες (arithmetic conditions), και σχέσεις πολυσυνόλων (multiset relations), πράγματα που δεν υπάρχουν σε καθαρές λογικές ερωτήσεις), και έχει βρει χρήση στην αποτίμηση μη-αναδρομικών "φωλιασμένων" (nested) ερωτήσεων SQL.

9.5.5 Stratified negation

Μια γλώσσα ερωτήσεων για επαγωγικές βάσεις δεδομένων μπορεί να επεκταθεί επιτρέποντας την άρνηση στα σώματα των κανόνων. Από την στιγμή πάντως που επιτρέπουμε αρνητικές παραστάσεις στους κανόνες, χάνουμε μια σημαντική ιδιότητα των κανόνων που λέγεται το *ελάχιστο μοντέλο*, για το οποίο μιλήσαμε νωρίτερα. Το μοντέλο υπολογίζεται με απλή ή ημι-απλή αποτίμηση, όπως συζητήθηκε στην προηγούμενη ενότητα. Με την παρουσία όμως αρνητικών παραστάσεων, κάποιο πρόγραμμα ίσως να μην έχει ένα ελάχιστο μοντέλο. Για παράδειγμα, το πρόγραμμα

```
p(a) :- not p(b).
```

(όπου `not` το σύμβολο της άρνησης) έχει δυο ελάχιστα μοντέλα: `{p(a)}` και `{p(b)}`.

Μια λεπτομερής ιστορία των εννοιών της άρνησης βρίσκεται έξω από τους στόχους μας. Για πρακτικούς λόγους πάντως, παρουσιάζουμε στην συνέχεια την σημαντική έννοια της *stratified negation*, που χρησιμοποιείται σε υλοποιήσεις επαγωγικών συστημάτων.

Το νόημα ενός προγράμματος με άρνηση δίνεται συνήθως μέσω κάποιου "προτιθέμενου" μοντέλου (intended model). Η πρόκληση είναι να αναπτυχθούν αλγόριθμοι για την επιλογή ενός προτιθέμενου μοντέλου που:

- Θα έχει νόημα για τον χρήστη των κανόνων.
- Θα μας επιτρέπει να απαντούμε με αποδοτικό τρόπο ερωτήσεις πάνω στο μοντέλο.

Συγκεκριμένα, είναι επιθυμητό το μοντέλο να δουλεύει καλά με τον μετασχηματισμό μαγικών συνόλων, με την έννοια του να μπορούμε να μετατρέψουμε τους κανόνες με κάποια κατάλληλη γενίκευση των μαγικών συνόλων, και οι νέοι κανόνες να επιτρέπουν μόνο στο σχετικό κομμάτι του επιλεγμένου μοντέλου να υπολογίζεται αποδοτικά. Εναλλακτικά μπορούν να αναπτυχθούν και άλλες τεχνικές αποδοτικής αποτίμησης.

Μια σημαντική κατηγορία άρνησης που έχει μελετηθεί σε βάθος είναι η stratified negation. *Ένα πρόγραμμα είναι stratified εάν δεν υπάρχει αναδρομή μέσω της άρνησης*. Τα προγράμματα αυτής της κατηγορίας έχουν πολύ κατανοητή σημασία και μπορούν να αποτιμηθούν αποδοτικά. Το ακόλουθο παράδειγμα περιγράφει ένα stratified πρόγραμμα. Θεωρήστε το ακόλουθο πρόγραμμα P2:

```
κ1: πρόγονος(X,Y) :- γονέας(X,Y).
κ2: πρόγονος(X,Y) :- γονέας(X,Z), πρόγονος(Z,Y).
κ3: όχικυκ(X,Y) :- πρόγονος(X,Y), not πρόγονος(Y,X).
```

Παρατηρείστε ότι στο σώμα του τρίτου κανόνα υπάρχει άρνηση. Αυτό το πρόγραμμα είναι stratified, επειδή ο ορισμός του κατηγορήματος *όχικυκ* εξαρτάται (με αρνητικό τρόπο) από τον

ορισμό του πρόγονος, αλλά ο ορισμός του *πρόγονος* δεν εξαρτάται από τον ορισμό του *όχικυκ*. Δεν είμαστε έτοιμοι να δώσουμε έναν πιο αυστηρό ορισμό αν δεν εισάγουμε κάποιες επιπλέον έννοιες. Μια από-κάτω-προς-τα-πάνω αποτίμηση του P2 θα υπολόγιζε πρώτα ένα σταθερό σημείο των κανόνων κ1 και κ2 (οι οποίοι ορίζουν το πρόγονος). Ο κανόνας κ3 εφαρμόζεται μόνο όταν όλα τα γεγονότα για το πρόγονος έχουν εξαχθεί.

Μια φυσική προέκταση των stratified προγραμμάτων είναι η κατηγορία των τοπικά stratified προγραμμάτων. *Ένα πρόγραμμα P είναι τοπικά stratified για μια συγκεκριμένη βάση δεδομένων εάν, όταν αντικαταστήσουμε τις μεταβλητές με σταθερές τιμές με κάθε δυνατό τρόπο, οι κανόνες που προκύπτουν δεν έχουν αναδρομή μέσω άρνησης.*

9.6 ΑΝΑΦΟΡΕΣ

[EN94] R. Elmasri, S. Navathe. Fundamentals on Database Systems. The Benjamin/Cummings Publishing Company, 1994.