Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Κατασκεύη γεννήτριας μεταφραστών SGML

Αναστάσιος Κεμεντσιετσίδης

Μεταπτυχιακή Εργασία

Ηράκλειο, Ιούνιος 1998

Πανεπιστήμιο Κρήτης
Σχολή Θετικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Κατασκεύη γεννήτριας μεταφραστών SGML

Εργασία που υποβλήθηκε από τον
Αναστάσιος Κεμεντσιετσίδης
ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Αναστάσιος Κεμεντσιετσίδης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εισηγητική Επιτροπή:

Πάνος Κωνσταντόπουλος, Καθηγητής, Επόπτης

Γεώργιος Γεωργακόπουλος, Επίκουρος Καθηγητής, Μέλος

Ευάγγελος Μαρκάτος, Επίκουρος Καθηγητής, Μέλος

Βασίλης Χριστοφίδης, Δόκιμος Ερευνητής Ι.Π. ΙΤΕ, Επιβλέπων, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ιούνιος 1998

... Σε μια αγάπη που χάθηκε.

.

Κατασκεύη γεννήτριας μεταφραστών SGML

Αναστάσιος Κεμεντσιετσίδης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Το έντυπο υλικό αποτελεί αυτή τη στιγμή τον βασικό άξονα ανταλλαγής πληροφοριών. Πλήθος εγγράφων περνάει καθημερινά μέσα απο τα χέρια του κάθε ανθρώπου. Η αποδοτική διαχείρηση όλων αυτών των εγγράφων αποτελεί μία χρονοβόρα και επίπονη διαδικασία. Οι υπολογιστές αποδείχθηκε, απο πολύ νωρίς, οτι αποτελούν το καταλληλότερο εργαλείο, για την διαχείριση και επεξεργασία των εγγράφων. Για το σκοπό αυτό μία σειρά απο πρότυπα δημιουργήθηκαν, μεταξύ των οποίων και η Standard Generalized Markup Language, SGML, η οποία αποτελεί το πρότυπο ISO 8879. Η SGML είναι μία περιγραφική γλώσσα κωδικοποίησης η οποία χρησιμοποιείται κυρίως για την περιγραφή της λογικής δομής των εγγράφων, δομή η οποία ορίζεται μέσω του *Καθορισμού Τύπου Εγγράφου.* Ενα απο τα κύρια χαρακτηριστικά της είναι η ανεξαρτησία αυτής απο αρχιτεκτονική, επιτρέποντας ετσι την ελεύθερη ανταλλαγή και μεταφορά εγγράφων.

Αντικειμενικός σκοπός της παρούσης εργασίας είναι η κατασκευή ενός συντακτικού και λεξικογραφικού μετα-αναλυτή. Ο μετα-αναλυτής δέχεται ως είσοδο ένα καθορισμό τύπου εγγράφου και παράγει ως έξοδο ενα συντακτικό και λεξικογραφικό αναλυτή των περιπτώσεων εγγράφων του δοθέντος τύπου. Η προσέγγιση αυτή υπερέχει έναντι των προηγουμένων λόγω του φορμαλισμού τον οποίο εισάγει στην συντακτική και λεξικογραφική ανάλυση, και της ευελιξίας και ανοικτής αρχιτεκτονικής της. Αυτή τη στιγμή η παρούσα υλοποίηση χρησιμοποιείται στο πρόγραμμα AQUARELLE για την συντακτική και λεξικογραφική ανάλυση εγγράφων SGML καθώς και για την φόρτωση της δομής των εγγράφων στο Σύστημα Σημασιολογικού Ευρετηριασμού.

Επόπτης
Π. Χ. Κωνσταντόπουλος, Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

DTD Specific Parser Generator, DSPG

Anastasios Kementsietsidis

Master of Science Thesis

Computer Science Department
University of Crete

Abstract

In everyday life we come across a number of documents. It is commonly known that it is not an easy task to keep track and manipulate these documents. The use of computers facilitated document manipulation by permitting us to store the documents in an electronic form. Currently there are a number of standards available that deal with the problem of document manipulation, and Standard Generalized Markup Language (SGML), which is ISO standard 8879, is one of them. SGML is a descriptive markup language and it is mainly used for the description of the logical structure of documents. The logical structure of a class of documents is defined by a mechanism called Document Type Definition (DTD). One of the main characteristics of SGML is architecture independence, which permits document interchange.

The objective of this thesis is the construction of an SGML parser generator. This takes as input a DTD and produces as output a parser, that parses the instance documents of the specified DTD. The suggested approach introduces formalism in the syntactical and lexical analysis. It is flexible and it follows an open architecture model. All these features make this approach more attractive than previous ones. The current implementation is used in the AQUARELLE project for parsing SGML documents and for loading their structure into the Semantic Index System.

Supervisor
P. Constantopoulos, Professor
Computer Science Department
University of Crete

Ευχαριστίες

Κατ' αρχήν θα ήθελα να ευχαριστήσω το Θεό.

Ευχαριστώ τους γονείς μου, Γεδεών και Γεωργία Κεμεντσιετσίδη, στους οποίους οφείλω την ύπαρξη μου, και οι οποίοι με ανάθρεψαν με τον καλύτερο δυνατό τρόπο, και μου έδωσαν όλα τα απαραίτητα εφόδια για να αντιμετωπίσω τις δυσκολίες της ζωής.

Αν στους γονείς μου οφείλω την βιολογική μου ζωή στον κ. Π. Χ. Κωνσταντόπουλο οφείλω την ακαδημαϊκή μου ζωή. Είναι αυτός ο οποίος απο πολύ νωρίς με εμπιστεύθηκε και με την ορθή καθοδήγηση του είμαι σε θέση σήμερα να είμαι ο συγγραφέας αυτής εδώ της εργασίας.

Θα ήθελα να ευχαριστήσω τον κ. Β. Χριστοφίδη, υπο την καθοδήγηση του οποίου δούλεψα τον τελευταίο χρόνο. Οι πολύτιμες συμβουλές του έπαιξαν καταλυτικό ρόλο στην ολοκλήρωση της παρούσας εργασίας. Θα ήθελα επίσης να ευχαριστήσω τον κ. Γ. Γεωργακόπουλο για την βοήθεια του κατά την εκπόνηση της εργασίας, και τον κ. Ε. Μαρκάτο για τις διορθώσεις και τις παρατηρήσεις του.

Ευχαριστώ τον αδερφό μου, Αθανάσιο Κεμεντσιετσίδη, και τους φίλους μου Χρυσόστομο Θεωδορούδη και Αρσένη Πολυμενόπουλο που μου συμπαραστάθηκαν σε όλες τις δύσκολες στιγμές της ζωής μου, αν και μας χώριζαν, συχνά, πολλές εκατοντάδες χιλιομέτρων.

θα ήθελα να ευχαριστήσω την ομάδα Πληροφοριακών Συστημάτων του Ινστιτούτου Πληροφορικής καθώς και το Ιδρυμα Τεχνολογίας και Ερευνας στο σύνολο του, για την υλικοτεχνική υποδομή που μου παρείχε κατα την διάρκεια των σπουδών μου.

Τέλος θα ήθελα να ευχαριστήσω όλους τους φίλους και τους εχθρούς μου. Οι πρώτοι με όπλισαν με κουράγιο και οι δεύτεροι με πείσμα για να συνεχίσω.

Την εργασία αυτή την αφιερώνω σε μια αγάπη που χάθηκε...

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Το έντυπο υλικό αποτελεί αυτή τη στιγμή τον βασικό άξονα ανταλλαγής πληροφοριών. Η αποθήκευση αλλά και η αποδοτική διαχείριση των εγγράφων αποτελεί μία χρονοβόρα και επίπονη διαδικασία. Ποικίλες μέθοδοι χρησιμοποιήθηκαν στο παρελθόν για να δώσουν λύση σε αυτό το πρόβλημα. Οι βιβλιοθήκες, ήταν μία από τις πρώτες προσεγγίσεις. Τα έγγραφα οργανώνονται σε θεματικές ενότητες και ευρετηριάζονται για την γρήγορη ανεύρεση τους. Δυστυχώς η μέθοδος αυτή αποδείχθηκε μη αποτελεσματική καθώς διαπιστώθηκε ότι, ειδικά σε μεγάλες βιβλιοθήκες, λόγω του τεραστίου πλήθους των εγγράφων, η ανεύρεση ενός από αυτά αποτελούσε συχνά έργο ημερών. Το πρόβλημα κατέστη τόσο έντονο ώστε συχνά τα ευρετήρια των εγγράφων της βιβλιοθήκης, αποτελούσαν με την σειρά τους μία μικρή βιβλιοθήκη. Μία δεύτερη προσέγγιση, η οποία έλυσε τον πρόβλημα του χώρου και της άμεσης ανεύρεσης, ήταν η χρήση microfilm. Παρόλα αυτά, η άμεση ανεύρεση συναφών πληροφοριών από πολλές πηγές συγχρόνως, πηγές τις οποίες δεν γνώριζε ίσως ούτε ο ίδιος ο ενδιαφερόμενος, αποτελούσε μακρινό όνειρο.

Οι υπολογιστές από πολύ νωρίς θεωρήθηκαν ως ένα από τα καταλληλότερα εργαλεία, το οποίο θα έδινε λύση στο πρόβλημα της διαχείρισης εγγράφων. Η ηλεκτρονική αποθήκευση των εγγράφων, τώρα πια, είναι εξαιρετικά οικονομική από άποψη χώρου και χρόνου, και η ανάκτηση αυτών αποτελεί θέμα μερικών δευτερολέπτων. Επίσης καθίσταται δυνατή η άμεση πρόσβαση σε έγγραφα τα οποία βρίσκονται σε διαφορετική γεωγραφική περιοχή από την εκάστοτε.

Τα ηλεκτρονικά αποθηκευόμενα έγγραφα, στα οποία θα αναφερόμαστε από εδώ και πέρα χρησιμοποιώντας τον όρο ηλεκτρονικά έγγραφα ή απλώς έγγραφα, και η διαχείριση αυτών, απασχολούν σήμερα ολοένα και περισσότερους μηχανικούς και ερευνητές στον τομέα της αποθήκευσης και ανάκτησης πληροφοριών. Τα προβλήματα που ανακύπτουν

οφείλονται συχνά στις ολοένα και αυξανόμενες απαιτήσεις των χρηστών. Ενας επίσης σημαντικός λόγος είναι το γεγονός ότι η έννοια "ηλεκτρονικό έγγραφο" έχει διευρυνθεί αρκετά ώστε να εμπεριέχει κείμενα πολυμέσων, η διαχείριση των οποίων αποτελεί μη τετριμμένο έργο. Αυτή ακριβώς η διεύρυνση οδήγησε τους ερευνητές στη παρατήρηση ότι ένα έγγραφο, ανάλογα με το σημείο εστίασης του ενδιαφέροντος, έχει:

- Μία Λογική Δομή, η οποία παριστάνει το σκεπτικό της οργάνωσης του εγγράφου που ακολούθησε ο δημιουργός κατά την συγγραφή αυτού. Η λογική δομή ενός εγγράφου είναι, παραδείγματος χάριν, η οργάνωση αυτού σε κεφάλαια, υποκεφάλαια, παραγράφους, η χρήση υποσημειώσεων, αναφορών κοκ.

- Μία Εννοιολογική Δομή, η οποία παριστάνει την κοινή αντίληψη που έχουν οι ενδεχόμενοι αναγνώστες του ιδίου εγγράφου. Ο αναγνώστης μπορεί να κατανοήσει την θεματική ενότητα του εγγράφου, το συγκεκριμένο θέμα που πραγματεύεται το εν λόγω έγγραφο, την γλώσσα στην οποία είναι γραμμένο κτλ.

- Μία Φυσική Δομή, η οποία παριστάνει τον τρόπο εμφάνισης του εγγράφου σε κάποιο μέσο, όπως π.χ. στο χαρτί, στην οθόνη. Η φυσική δομή ενός εγγράφου είναι η γραμματοσειρά η οποία θα χρησιμοποιηθεί για την εμφάνιση του, το μέγεθος της γραμματοσειράς αυτής κτλ.

Και οι τρεις αυτές δομές βρίσκονται ενσωματωμένες σε κάθε έγγραφο. Ποια όμως είναι η σχέση ανάμεσα στη μία και στην άλλη δομή; Γενικά στην ίδια λογική δομή μπορούν να αντιστοιχιστούν πολλές φυσικές δομές. Ετσι μία λίστα αντικειμένων, μέρος της λογικής δομής ενός κειμένου, σαν αυτή που απαριθμεί παραπάνω τις τρεις δομές, μπορεί να έχει διαφορετικούς τρόπους εμφάνισης, χρησιμοποιώντας τελείες, αύξοντες αριθμούς κοκ. Επίσης η ίδια φυσική δομή μπορεί να αντιστοιχιστεί σε διαφορετικές λογικές δομές. Ετσι η πλάγια γραμματοσειρά, μέρος της φυσικής δομής ενός κειμένου, μπορεί να χρησιμοποιηθεί για την παράσταση τόσο της ειδικής ορολογίας ενός κειμένου, όσο και των βιβλιογραφικών αναφορών, που αποτελούν μέρη της λογικής δομής αυτού. Οι δύο αυτές δομές, φυσική και λογική, δεν είναι όμως εντελώς ανεξάρτητες. Είναι σύνηθες το φαινόμενο της αλλαγής σελίδας, μέρος της φυσικής δομής, με την εκκίνηση ενός νέου κεφαλαίου, το οποίο είναι μέρος της λογικής δομής. Οσον αφορά την σχέση ανάμεσα στην εννοιολογική και στη λογική αυτό που μπορεί να ειπωθεί με σιγουριά είναι ότι σχετίζονται κατά ένα μεγάλο βαθμό, σε αντίθεση με την εννοιολογική και την φυσική δομή οι οποίες δεν δείχνουν να σχετίζονται σε μεγάλο βαθμό.

Μέχρι πρόσφατα το λογισμικό, το οποίο κατασκευάζονταν για διαχείριση εγγράφων, έτεινε να αγνοεί μία τουλάχιστον από αυτές τις δομές, ή, ακόμα χειρότερα, έτεινε να μην τις διαχωρίζει. Ετσι από την μια πλευρά οι επονομαζόμενοι *Επεξεργαστές Κειμένων* έδιναν ιδιαίτερη έμφαση στην φυσική δομή του κειμένου και μερικοί από αυτούς και στην λογική δομή αγνοώντας πλήρως την εννοιολογική δομή. Απο την άλλη πλευρά τα *Συστήματα Διαχείρισης Εγγράφων* επικέντρωναν το ενδιαφέρον τους στην εννοιολογική και λογική δομή των εγγράφων αγνοώντας πλήρως την φυσική δομή.

Για να δοθούν λύσεις στα προβλήματα τα οποία ανέκυψαν, μία σειρά από πρότυπα δημιουργήθηκαν για τη διαχείριση και επεξεργασία των εγγράφων. Ενδεικτικά αναφέρονται τα ακόλουθα, SGML[ISO86], DSSSL[ISO96], SPDL[ISO95], ODA[ISO89] και το υπο μελέτη πρότυπο XML[Con97]. Η παρούσα εργασία ασχολείται με το πρότυπο της SGML.

Η Standard Generalized Markup Language, SGML, η οποία αποτελεί το πρότυπο ISO 8879, προσφέρει την δυνατότητα καθορισμού και των τριών αυτών δομών με ιδιαίτερη έμφαση στην λογική δομή. Ενας από τους κυρίους στόχους του προτύπου, δεν είναι μόνο ο καθορισμός των τριών δομών, αλλά και ο σαφής διαχωρισμός τους. Σε πρώτη φάση η SGML χρησιμοποιείται για τον *Καθορισμό Τύπου Εγγράφου.* Ο καθορισμός τύπου εγγράφου είναι στην ουσία ένας τρόπος καθορισμού της λογικής δομής ενός εγγράφου. Κάθε έγγραφο το οποίο έχει την ίδια λογική δομή με αυτή, την οποία ορίζει ένας συγκεκριμένος καθορισμός τύπου εγγράφου, αποτελεί περίπτωση εγγράφου του εν λόγω τύπου. Ο καθορισμός τύπου εγγράφου σε συνδυασμό με την εκάστοτε περίπτωση εγγράφου καθορίζει πλήρως την εννοιολογική δομή της περίπτωσης του εγγράφου. Αυτό γίνεται εμφανές και από το εξής παράδειγμα. Χρησιμοποιώντας το πρότυπο SGML μπορεί κανείς να ορίσει τον καθορισμό τύπου εγγράφου για ένα βιβλίο. Ορίζει ότι ένα βιβλίο αποτελείται από κεφάλαια, υποκεφάλαια και παραγράφους, περιέχει εικόνες, αναφορές κοκ., ορίζοντας έτσι την λογική δομή αυτού. Σε ένα συγκεκριμένο βιβλίο, γραμμένο στα γαλλικά με θέμα την ζωγραφική του Vincent van Gohg, το οποίο αποτελεί περίπτωση του προαναφερθέντος τύπου εγγράφου, είναι ξεκάθαρο τόσο σε ποια γλώσσα αυτό είναι γραμμένο όσο και η θεματική ενότητα του βιβλίου καθώς και το θέμα το οποίο πραγματεύεται, άρα είναι ξεκάθαρη η εννοιολογική δομή αυτού. Το πρότυπο προσφέρει όλους του απαραίτητους μηχανισμούς για την αποθήκευση της πληροφορίας, η οποία έχει να κάνει τόσο με την λογική όσο και με την εννοιολογική δομή των εγγράφων. Οσον αφορά την φυσική δομή του εγγράφου, το πρότυπο προσφέρει τη δυνατότητα εισαγωγής οδηγιών επεξεργασίας τόσο στο εσωτερικό του καθορισμού τύπου εγγράφου όσο και στο εσωτερικό της περίπτωσης του εγγράφου δίνοντας έτσι, στον εκάστοτε

διαχειριστή του εγγράφου, την δυνατότητα να ορίσει την φυσική δομή αυτού. Αν και η δυνατότητα ορισμού της φυσικής δομής προσφέρεται από το πρότυπο, σπάνια γίνεται χρήση της δυνατότητας αυτής, αφού στόχος του προτύπου είναι να μην εμπεριέχονται οδηγίες επεξεργασίας μέσα στο έγγραφο, προσφέροντας έτσι στα προγράμματα που διαχειρίζονται το έγγραφο, απόλυτη ελευθερία στον καθορισμό της φυσικής δομής αυτού.

Τόσο οι καθορισμοί τύπου όσο και οι αντίστοιχες περιπτώσεις εγγράφων αποθηκεύονται στον υπολογιστή ως απλοί χαρακτήρες. Καμία ειδική κωδικοποίηση δεν χρησιμοποιείται που θα έκανε την μεταφορά των κειμένων μεταξύ διαφορετικών αρχιτεκτονικών από δύσκολη ως αδύνατη. Ετσι τα έγγραφα ανεξαρτητοποιούνται από τη αρχιτεκτονική αλλά και από το ίδιο το πρόγραμμα το οποίο τα επεξεργάζεται κάθε φορά.

Πλήθος εφαρμογών, όπως επεξεργαστές κειμένου, μεταφραστές, υπάρχουν αυτή τη στιγμή και υποστηρίζουν το πρότυπο της SGML. Αντικειμενικός σκοπός της παρούσης εργασίας είναι η κατασκευή ενός συντακτικού και λεξικογραφικού μετα-αναλυτή. Ο μετα-αναλυτής δέχεται ως είσοδο ένα καθορισμό τύπου εγγράφου και παράγει ως έξοδο ένα συντακτικό και λεξικογραφικό αναλυτή των περιπτώσεων του δοθέντος τύπου. Επιθυμώντας την διατήρηση της ανεξαρτησίας από την αρχιτεκτονική, η παρούσα εργασία κάνει χρήση εργαλείων ευρέως διαδεδομένων έτσι ώστε ο μετα-αναλυτής να είναι διαθέσιμος όποτε και η SGML είναι διαθέσιμη. Τα εργαλεία αυτά είναι ο κατασκευαστής συντακτικών αναλυτών Bison[DS90] και ο κατασκευαστής λεξικογραφικών αναλυτών Flex[Pax88], τα οποία αποτελούν βελτιωμένες εκδόσεις των εργαλείων Yacc και Lex αντίστοιχα. Ο κώδικας είναι γραμμένος στη γλώσσα προγραμματισμού Ansi C [KR78].

1.1 Οργάνωση της εργασίας

Το δεύτερο κεφάλαιο αφιερώνεται στην παρουσίαση του προτύπου της SGML. Στο τρίτο κεφάλαιο γίνεται μια ανασκόπηση των υπαρχόντων συστημάτων, τα οποία υποστηρίζουν το πρότυπο. Το τέταρτο κεφάλαιο αφιερώνεται στη περιγραφή της προτεινόμενης προσέγγισης στο πρόβλημα της συντακτικής και λεξικογραφικής ανάλυσης εγγράφων SGML. Στο πέμπτο κεφάλαιο παρουσιάζεται μία εφαρμογή της παρούσας εργασίας, μέσα στα πλαίσια του ερευνητικού προγράμματος AQUARELLE. Η εργασία ολοκληρώνεται με το έκτο κεφάλαιο, όπου παρουσιάζονται τα συμπεράσματα καθώς και οι μελλοντικές επεκτάσεις της παρούσης εργασίας.

Κεφάλαιο 2

Το πρότυπο SGML

Το παρόν κεφάλαιο θα ξεκινήσει με μια σύντομη ιστορική αναδρομή στην οποία θα περιγραφεί η εξέλιξη του προτύπου και τα στάδια από τα οποία αυτή πέρασε. Κατόπιν θα γίνει μία περιγραφή των κυρίων χαρακτηριστικών του προτύπου, χαρακτηριστικά τα οποία συνέβαλαν στην καθιέρωση και στην αποδοχή αυτού, από την επιστημονική κοινότητα. Το υπόλοιπο του κεφαλαίου θα αφιερωθεί στην παρουσίαση του προτύπου.

2.1 Ιστορική αναδρομή

Στο παρελθόν είχαν αναπτυχθεί τρόποι κωδικοποίησης εγγράφων οι οποίοι χρησιμοποιούνταν για τη μορφοποίηση τους. Τα κωδικοποιημένα έγγραφα εμπεριείχαν κωδικούς ελέγχου ή/και μακροεντολές οι οποίες καθόριζαν πλήρως και κατά τρόπο **μοναδικό** την μορφοποίηση του εγγράφου. Αυτός ο τρόπος κωδικοποίησης ονομάστηκε *ειδικευμένη κωδικοποίηση* (specific coding). Αργότερα, τέλη της δεκαετίας του 60, υιοθετήθηκαν τρόποι *γενικευμένης κωδικοποίησης* (generic coding), που σκοπό είχαν την εισαγωγή ειδικών λέξεων, ετικετών (tags), στα περιεχόμενα εγγράφων ώστε να περιγράφεται η λογική δομή του κειμένου. Αντικειμενικός σκοπός της γενικευμένης κωδικοποίησης ήταν ο διαχωρισμός της λογικής δομής του εγγράφου από την φυσική δομή αυτού.

Το 1969 οι Goldfarb, Mosher και Lorie δημιούργησαν την Generalized Markup Language ή αλλιώς GML. Η GML εισήγαγε την έννοια του τυπικά ορισμένου Τύπου Εγγράφου, (Document Type). Σκοπός της GML ήταν η συγγραφή, μορφοποίηση και ανάκτηση κοινωνουμένων εγγράφων. Το 1978 ζητήθηκε από τον Goldfarb να ηγηθεί ενός προγράμματος για τη δημιουργία ενός προτύπου βασισμένο στην GML. Το 1980 έγινε η πρώτη δημοσίευση για το πρότυπο αυτό, πρότυπο με την ονομασία Standard Generalized

Markup Language (SGML). Μία σειρά από περίπου δέκα δημοσιεύσεις ακολούθησαν μέχρι το 1986, οπότε και δημοσιεύθηκε το επίσημο κείμενο του προτύπου με κωδικό ISO 8879:1986.

2.2 Χαρακτηριστικά της SGML

Υπάρχουν τρία χαρακτηριστικά της SGML τα οποία την κάνουν να ξεχωρίζει από τις υπόλοιπες γλώσσες κωδικοποίησης. Το πρώτο χαρακτηριστικό είναι ότι η SGML είναι μια περιγραφική γλώσσα κωδικοποίησης (descriptive mark-up language), και όχι μία διαδικαστική γλώσσα κωδικοποίησης (procedural mark-up language). Οι ειδικές λέξεις, ετικέτες (tags), που χρησιμοποιούνται για την κωδικοποίηση ενός εγγράφου SGML απλώς περιγράφουν την λογική δομή αυτού, χωρίς να λένε τίποτα για την φυσική δομή, κάτι που συμβαίνει σε άλλες γλώσσες κωδικοποίησης όπως η γλώσσα LaTex[Lam86]. Στο ακόλουθο παράδειγμα, το οποίο αποτελεί μέρος περίπτωσης εγγράφου ακολουθώντας την σύνταξη της SGML, η ετικέτα <line> μαρκάρει την αρχή μίας γραμμής ενώ η ετικέτα </line> μαρκάρει τον τερματισμό αυτής. Γενικά στο συντακτικό της SGML η ετικέτα αρχής (start tag) έχει τη μορφή $<$*όνομα ετικέτας*$>$, ενώ η ετικέτα τέλους (end tag) έχει τη μορφή $</$*όνομα ετικέτας*$>$.

Παράδειγμα 2.1 *:*

$< stanza >$
$< line >$ *Κοιτάς μακριά μα δεν μου λες τι βλέπεις* $< /line >$
$< line >$ *μιλάς σωστά μα δεν μου λες τι ξέρεις* $< /line >$
$< line >$ *έτσι κι εγώ δεν θα σου πω ποτέ μου* $< /line >$
$< line >$ *πως πάντα θα σε αγαπώ* $< /line >$
$< /stanza >$

Η ετικέτα $<$*line*$>$ φανερώνει ότι το κείμενο που ακολουθεί είναι μια γραμμή χωρίς να καθορίζει πουθενά αν αυτό θα εμφανιστεί με πλάγιους χαρακτήρες ή με απλούς χαρακτήρες, κάτι που συμβαίνει γενικά σε μια γλώσσα όπως η LaTex. Είναι θέμα της εκάστοτε εφαρμογής που αναλαμβάνει την εμφάνιση του κειμένου πώς θα χειριστεί τις γραμμές.

Ενα άλλο χαρακτηριστικό της SGML είναι η εισαγωγή της έννοιας του Τύπου Εγγράφου. Το έγγραφο δεν έχει απλώς κάποια λογική δομή αλλά αυτή η δομή πρέπει να υπακούει συγκεκριμένους κανόνες όπως αυτοί ορίζονται στον καθορισμό τύπου εγγράφου του οποίου το εν λόγω έγγραφο αποτελεί περίπτωση. Ετσι στο παραπάνω

παράδειγμα του ποιήματος ο απλός κανόνας που ακολουθείται είναι ότι μέσα σε μία στροφή ποιήματος υπάρχουν μόνο γραμμές, μία ή περισσότερες. Η προσέγγιση της SGML στο θέμα του τύπου εγγράφου θυμίζει την προσέγγιση των γλωσσών προγραμματισμού όπου κάθε μεταβλητή πρέπει να έχει κάποιο τύπο. Αυτός ο αυστηρός καθορισμός επιτρέπει την ομαδοποίηση των εγγράφων του ιδίου τύπου και την επεξεργασία τους από ένα πρόγραμμα, το οποίο εκμεταλλευόμενο τη γνώση της κοινής δομής αυτών, μπορεί να τα επεξεργαστεί πιο *έξυπνα* και αποδοτικά[1]. Η καθορισμένη δομή διευκολύνει και στην γρηγορότερη συγγραφή νέων εγγράφων και τα λάθη που προκύπτουν μπορούν εύκολα να διαπιστωθούν από ένα συντακτικό αναλυτή ικανό να αναγνωρίζει έγγραφα του εν λόγω τύπου.

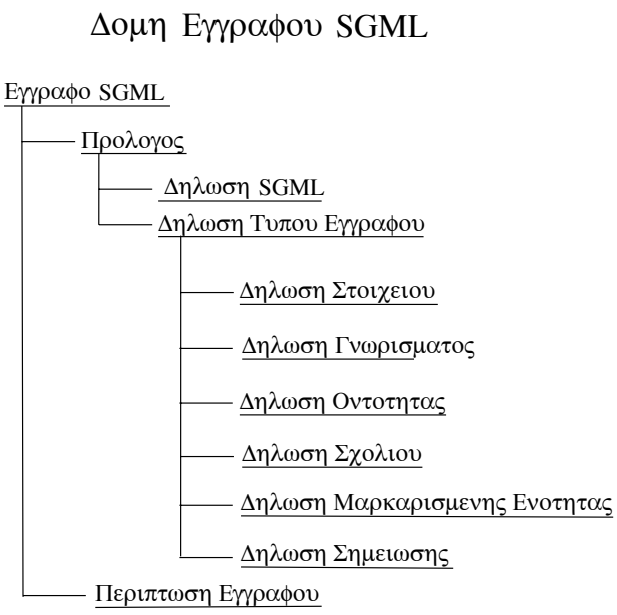
Τέλος ένα σημαντικό χαρακτηριστικό της γλώσσας είναι η ανεξαρτησία αυτής από την αρχιτεκτονική αλλά και η ανεξαρτησία της από οποιοδήποτε λογισμικό το οποίο θα αναλάβει να κατασκευάσει/επεξεργαστεί το έγγραφο. Το παραπάνω ποίημα είναι απλοί χαρακτήρες, όπως και όλα τα έγγραφα SGML, κάτι που μας εξασφαλίζει κατά μεγάλο μέρος την μεταφορά κειμένων μεταξύ διαφορετικών αρχιτεκτονικών. Παρόλα αυτά υπάρχουν περιπτώσεις στις οποίες πρέπει να χρησιμοποιηθούν ειδικοί χαρακτήρες στο κείμενο οι οποίοι εξαρτώνται από το σύστημα στο οποίο βρίσκεται αποθηκευμένο το έγγραφο. Για αυτούς τους μη μεταφέρσιμους χαρακτήρες το πρότυπο προσφέρει ένα τρόπο έμμεσης αναφοράς σε χαρακτήρες μέσω των *οντοτήτων*. Είναι δυνατόν να οριστεί μία οντότητα χαρακτήρα η οποία να συνδέεται με ένα μη μεταφέρσιμο χαρακτήρα μέσω του αύξοντα αριθμού που αυτός έχει στο σύνολο χαρακτήρων του δεδομένου συστήματος. Κάθε φορά που θέλει κανείς να *γράψει* τον χαρακτήρα χρησιμοποιεί το όνομα της οντότητας με την οποία ο χαρακτήρας συνδέθηκε. Κατά την μεταφορά του εγγράφου σε άλλη αρχιτεκτονική το μόνο που χρειάζεται να αλλάξει στο κείμενο είναι το σημείο στο οποίο γίνεται η σύνδεση της οντότητας με το συγκεκριμένο χαρακτήρα. Απλώς χρειάζεται να βρεθεί ο νέος αύξων αριθμός του ίδιου χαρακτήρα στο νέο σύνολο χαρακτήρων. Οι αναφορές δεν χρειάζεται να πειραχθούν. Οτι ισχύει για χαρακτήρες μπορεί να ισχύσει και για ολόκληρα αλφαριθμητικά, μέσω παρόμοιου μηχανισμού ο οποίος προσφέρεται από το πρότυπο.

[1] Αυτό θα γίνει πιο προφανές στο Κεφάλαιο 4, κατά την περιγραφή των κινήτρων τα οποία οδήγησαν στην κατασκευή της γεννήτριας μεταφραστών SGML.

2.3 Το πρότυπο της SGML

Η παρουσίαση του προτύπου της SGML θα γίνει μέσω της παρουσίασης της δομής ενός εγγράφου το οποίο είναι κωδικοποιημένο σύμφωνα με το πρότυπο. Η παρουσίαση των εννοιών, οι οποίες ορίζονται στο πρότυπο, μέσω του τρόπου με τον οποίο αυτές χρησιμοποιούνται για την κωδικοποίηση θα βοηθήσει στην καλύτερη κατανόηση των εννοιών αυτών. Επίσης για την πληρέστερη κατανόηση του προτύπου, πέρα από την λεκτική περιγραφή των εννοιών, θα χρησιμοποιηθούν και ένα σύνολο από παραγωγές BNF. Εφόσον το πρότυπο ορίζει στην ουσία μία περιγραφική γλώσσα κωδικοποίησης εγγράφων, η χρήση παραγωγών BNF είναι η πλέον αρμόζουσα για να παρασταθεί η δομή της περιγραφόμενης γλώσσας. Παρόμοια προσέγγιση χρησιμοποιήθηκε και από τον Goldfarb στο βιβλίο του *The SGML Handbook*[GR90].

Ενα έγγραφο το οποίο είναι κωδικοποιημένο σύμφωνα με το πρότυπο της SGLM θα ονομάζεται έγγραφο SGML. Οπως φαίνεται και από το Σχήμα 2.1 ένα έγγραφο SGML είναι ένα αρκετά σύνθετο έγγραφο.



Σχήμα 2.1: Γραφική παράσταση της δομής ενός εγγράφου SGML

Αποτελείται από δύο κυρίως μέρη, τον Πρόλογο SGML και τη Περίπτωση Εγγράφου. Ακολούθως θα παρουσιαστούν αναλυτικά κάθε ένα από τα δύο τμήματα, θα περιγραφεί η χρήση τους, η δομή και το είδος των περιεχομένων τους. Η δομή της παρουσίασης θα ακολουθήσει την δομή του εγγράφου SGML.

Παραγωγή 2.1 *:*

Εγγραφο SGML→ Πρόλογος SGML,
Περίπτωση Εγγράφου

2.4 Πρόλογος SGML

Ο πρόλογος SGML δηλώνει τι επιτρέπεται να εμφανίζεται στη περίπτωση του εγγράφου. Χωρίζεται σε δύο μέρη, την Δήλωση SGML, και την Δήλωση Τύπου Εγγράφου. Διαφορετικά συστήματα λογισμικού ορίζουν διαφορετικούς τρόπους διασύνδεσης ανάμεσα στην περίπτωση εγγράφου και στον πρόλογο SGML. Σε μερικές περιπτώσεις ο πρόλογος μπορεί να είναι ενσωματωμένος στο σύστημα λογισμικού που χειρίζεται τα έγγραφα, γεγονός που κάνει τον πρόλογο διαφανή στους χρήστες, επιτρέποντας την παράλειψη αυτού.

Παραγωγή 2.2 *:*

Πρόλογος SGML → Δήλωση SGML,
*(Δήλωση Τύπου Εγγράφου)**

Οπως φαίνεται από την παραπάνω παραγωγή ένας πρόλογος SGML μπορεί να έχει παραπάνω από μία Δηλώσεις Τύπου Εγγράφου. Μία από αυτές τις δηλώσεις θεωρείται η βασική και ονομάζεται Βασική Δήλωση Τύπου Εγγράφου.

2.4.1 Δήλωση SGML

Σκοπός της Δήλωσης SGML είναι ο καθορισμός της διαλέκτου SGML η οποία θα χρησιμοποιηθεί. Καθορίζεται έτσι η Κρυμένη Σύνταξη (Abstract Syntax) και η Συγκεκριμένη Σύνταξη (Concrete Syntax). Η κρυμένη σύνταξη καθορίζει τον τρόπο με οποίο θα γίνει η γενικευμένη κωδικοποίηση του κειμένου, χωρίς να ληφθεί υπόψη ο τρόπος με τον οποίο θα παρασταθούν οι ετικέτες της κωδικοποίησης, με ποιους δηλαδή χαρακτήρες, έξ ού και η ονομασία κρυμένη. Η συγκεκριμένη σύνταξη αντιστοιχεί στις έννοιες τις αφαιρετικής σύνταξης συγκεκριμένους χαρακτήρες, καθορίζοντας έτσι μια συγκεκριμένη μορφή της γενικευμένης κωδικοποίησης. Το πρότυπο προτείνει μία συγκεκριμένη σύνταξη, αλλά επιτρέπει την τροποποίηση αυτής σε περίπτωση που ο χρήστης το επιθυμεί. Ενα παράδειγμα είναι απαραίτητο για την καλύτερη κατανόηση των εννοιών.

Οπως φαίνεται και στο Σχήμα 2.2 η αφαιρετική σύνταξη καθορίζει ότι μία ετικέτα αρχής αποτελείται από τρία μέρη. Το χαρακτήρα εκκίνησης ετικέτας αρχής, ένα

Ετικεττα Αρχης

Αφαιρετικη Συνταξη	Χαρακτηρας εκινησης ετικετας αρχης	Ονομα ετικετας	Χαρακτηρας τερματισμου ετικετας αρχης
Συγκεκριμενη Συνταξη	<	Ονομα μεχρι 8 χαρακτηρες	>

Σχήμα 2.2: Αφαιρετική και Συγκεκριμένη Σύνταξη της ετικέτας αρχής

όνομα, και τον χαρακτήρα τερματισμού ετικέτας αρχής. Στη συγκεκριμένη σύνταξη καθορίζεται ότι ο χαρακτήρας εκκίνησης ετικέτας αρχής είναι ο <, το όνομα της ετικέτας είναι μέχρι 8 χαρακτήρες και ο χαρακτήρας τερματισμού ετικέτας αρχής είναι ο >.

Μέσα στα πλαίσια της συγκεκριμένης σύνταξης καθορίζεται επίσης το σύνολο των χαρακτήρων κειμένου οι οποίοι θα χρησιμοποιηθούν και το σύνολο των χαρακτήρων διαχωρισμού (delimiter characters) οι οποίοι θα χρησιμοποιηθούν. Τα δύο αυτά σύνολα χαρακτήρων δεν είναι ξένα μεταξύ τους. Υπάρχουν χαρακτήρες οι οποίοι, κάτω από ορισμένες προϋποθέσεις, θεωρούνται ως χαρακτήρες κειμένου, ενώ κάτω από άλλες προϋποθέσεις θεωρούνται χαρακτήρες διαχωρισμού. Χαρακτηριστικό παράδειγμα είναι ο χαρακτήρας < ο οποίος αν ακολουθείται από χαρακτήρα πεζού ή κεφαλαίου γράμματος θεωρείται ως ο ειδικός χαρακτήρας εκκίνησης ετικέτας αρχής, ενώ αν ακολουθείται από οποιοδήποτε άλλο χαρακτήρα δεν έχει καμία ειδική ερμηνεία. Εκτός από τα σύνολα χαρακτήρων, στην Δήλωση SGML, καθορίζονται οι κανόνες ονοματοδοσίας (Naming rules) που θα ακολουθηθούν, για την ονομασία παραδείγματος χάριν των ετικετών, καθώς επίσης και τα δεσμευμένα ονόματα. Επίσης δίδεται η δυνατότητα της ενεργοποίησης/απενεργοποίησης ενός συνόλου προαιρετικών δυνατοτήτων της SGML. Οι δυνατότητες αυτές χωρίζονται σε τρεις κατηγορίες οι οποίες θα περιγραφούν εν συντομία στη συνέχεια. Βασικό χαρακτηριστικό των εν λόγω δυνατοτήτων είναι ότι από την στιγμή που είναι προαιρετικές θα πρέπει να απενεργοποιούνται κατά την ανταλλαγή εγγράφων μεταξύ συστημάτων, αφού υπάρχει η πιθανότητα το σύστημα που δέχεται τα έγγραφα να μην τις υποστηρίζει. Η απενεργοποίηση των δυνατοτήτων συνεπάγεται πολλές φορές αλλαγή των εγγράφων.

1. Δυνατότητα Ελαχιστοποίησης Κωδικοποίησης

   Οι δυνατότητες αυτής της κατηγορίας είναι οι SHORTTAG, OMITTAG, SHORTREF,

DATATAG, RANK. Η SHORTTAG επιτρέπει στο δημιουργό του εγγράφου να παραλείψει κατά την πληκτρολόγηση του εγγράφου ορισμένες ετικέτες τέλους όταν αυτές ακολουθούνται αμέσως από άλλες ετικέτες αρχής ή τέλους, και επίσης επιτρέπει στο δημιουργό να εισαγάγει άδειες ετικέτες αρχής <> ή τέλους </> παραλείποντας το όνομα της ετικέτας, με την προϋπόθεση βέβαια ότι αυτή η παράλειψη δεν δημιουργεί αμφισημίες. Το παράδειγμα του ποιήματος, χρησιμοποιώντας αυτή την δυνατότητα, μπορεί να γραφτεί ως

Παράδειγμα 2.2 *:*

$< stanza >$
$<>$ *Κοιτάς μακριά μα δεν μου λες τι βλέπεις*
$<>$ *μιλάς σωστά μα δεν μου λες τι ξέρεις*
. . .

Δεν υπάρχει αμφισημία αφού από τον καθορισμό τύπου εγγράφου καθορίζεται, όπως θα δειχθεί στη συνέχεια, ότι μία stanza αποτελείται από lines οπότε το παραλειπόμενο όνομα lines από τις ετικέτες αρχής δεν δημιουργεί αμφισημία ως προς ποιο όνομα παραλείφθηκε.
Η δυνατότητα OMITTAG μας επιτρέπει της παράλειψη ετικετών αρχής ή τέλους με μόνο περιορισμό η παράλειψη αυτή να μην δημιουργεί αμφισημία. Οι δυνατότητες SHORTREF και DATATAG, παραβιάζοντας το συντακτικό της SGML επιτρέπουν σε κανονικούς χαρακτήρες κειμένου να παίξουν το ρόλο οντοτήτων και ετικετών αντίστοιχα. Τέλος η RANK προσφέρει ένα απλό τρόπο καθορισμού και παράστασης του επιπέδου ενθυλάκωσης με την προσθήκη δεικτών στις ετικέτες αρχής. Χρησιμοποιείται όταν, παραδείγματος χάριν, έχουμε μία παράγραφο μέσα σε μία άλλη παράγραφο, και το γεγονός της ενθυλάκωσης δεν είναι προφανές από την παρατήρηση των ετικετών αρχής και τέλους.

Παράδειγμα 2.3 *:*

$< parag1 >$ *Κοιτάς μακριά μα δεν μου λες τι βλέπεις...*
$< /parag >$
$< parag2 >$ *μιλάς σωστά μα δεν μου λες τι ξέρεις ...*
$< /parag >$

Χρησιμοποιούνται λοιπόν δείκτες, όπως φαίνεται και στο παράδειγμα, για να αναπαρασταθεί το γεγονός ότι η δεύτερη παράγραφος είναι υποπαράγραφος της πρώτης και όχι η επόμενη παράγραφος.

2. Δυνατότητες Συνδέσμων

   Η χρήση των εν λόγω δυνατοτήτων επιτρέπει στον χρήστη τον καθορισμό της επεξεργασίας την οποία θα υποστεί ένα έγγραφο κατά το μετασχηματισμό του σε ένα ισοδύναμο έγγραφο κάποιου άλλου τύπου, π.χ. το ίδιο έγγραφο αλλά φορμαρισμένο, με φυσική δομή αυστηρά καθορισμένη για κάποιο επεξεργαστή κειμένου. Οι δυνατότητες αυτής της κατηγορίας είναι οι SIMPLE, IMPLICIT, EXPLICIT.

3. Λοιπές Δυνατότητες

   Οι δυνατότητες αυτής της κατηγορίας είναι οι CONCUR, SUBDOC, FORMAL. Η CONCUR επιτρέπει την συντακτική ανάλυση μίας περίπτωσης εγγράφου για περισσότερους από ένα καθορισμούς τύπου εγγράφου. Ετσι ένα έγγραφο, ανάλογα με το ποιον καθορισμό τύπου εγγράφου θεωρούμε ενεργό, μπορεί να θεωρηθεί ότι έχει πολλές όψεις. Η SUBDOC επιτρέπει την εισαγωγή σε μία περίπτωση εγγράφου, ενός κειμένου το οποίο αποτελεί περίπτωση εγγράφου διαφορετικού καθορισμού τύπου εγγράφου από αυτόν του αρχικού. Τέλος η FORMAL καθορίζει και επιβάλλει ένα συγκεκριμένο τρόπο σύνταξης του κειμένου ορισμένων οντοτήτων.

Ενα έγγραφο SGML το οποίο έχει συνταχθεί χρησιμοποιώντας την εξ ορισμού συγκεκριμένη σύνταξη, και χωρίς να περιέχει καμία από τις προαιρετικές δυνατότητες οι οποίες περιγράφηκαν προηγουμένως, ονομάζεται *Ελάχιστο SGML Εγγραφο, (Minimal SGML Document).* Ενα έγγραφο SGML το οποίο έχει συνταχθεί χρησιμοποιώντας την εξ ορισμού συγκεκριμένη σύνταξη, και επιπλέον κάνει χρήση των δυνατοτήτων SHORTTAG και OMITTAG, ονομάζεται *Βασικό SGML Εγγραφο, (Basic SGML Document).* Τέλος ένα έγγραφο SGML το οποίο έχει συνταχθεί έχοντας ορίσει επιτούτου την συγκεκριμένη σύνταξη, ονομάζεται *Παραλλάσον SGML Εγγραφο, (Variant Conforming SGML Document).* Συχνά τα προγράμματα τα οποία υποστηρίζουν το πρότυπο, και ιδιαίτερα οι συντακτικοί και λεξικογραφικοί αναλυτές, πρέπει να αναφέρουν ποιο από τα τρία είδη εγγράφων αναγνωρίζουν.

2.4.2 Δήλωση Τύπου Εγγράφου

Ορίζει ένα καθορισμό τύπου εγγράφου (Document Type Definition), και αποδίδει ένα όνομα σε αυτόν. Ενας καθορισμός τύπος εγγράφου ορίζει τα στοιχεία (ELEMENT) από τα οποία αποτελείται ένα έγγραφο, τα γνωρίσματα (ATTRIBUTE) καθενός στοιχείου και τέλος ορίζει τη σειρά με την οποία τα στοιχεία που ορίστηκαν μπορούν να εμφανιστούν. Η

σειρά αυτή μπορεί να παρασταθεί με την μορφή ενός δένδρου με ένα μόνο ριζικό στοιχείο. Επίσης ορίζει ένα σύνολο από οντότητες (ENTITY) τις οποίες χρησιμοποιεί τόσο ο ίδιος ο καθορισμός τύπου εγγράφου όσο και οι περιπτώσεις του τύπου αυτού. Μέσα στον καθορισμό μπορούν να εμπεριέχονται επίσης οδηγίες επεξεργασίας για τις περιπτώσεις εγγράφων μέσω σημειώσεων (NOTATION). Μέρη του καθορισμού μπορούν, με τη χρήση μαρκαρισμένων ενοτήτων, άλλοτε να αγνοούνται και άλλοτε να λαμβάνονται υπόψη. Τέλος, ελεύθερο κείμενο μπορεί να εμφανίζεται μέσα στον καθορισμό με τη μορφή σχολίων. Οι παρακάτω παραγωγές συνοψίζουν όσα προαναφέρθηκαν.

Παραγωγή 2.3 :

Δήλωση Τύπου Εγγράφου → *<!DOCTYPE Ονομα Καθορισμού Τύπου Εγγράφου, Καθορισμός Τύπου Εγγράφου*

Παραγωγή 2.4 :

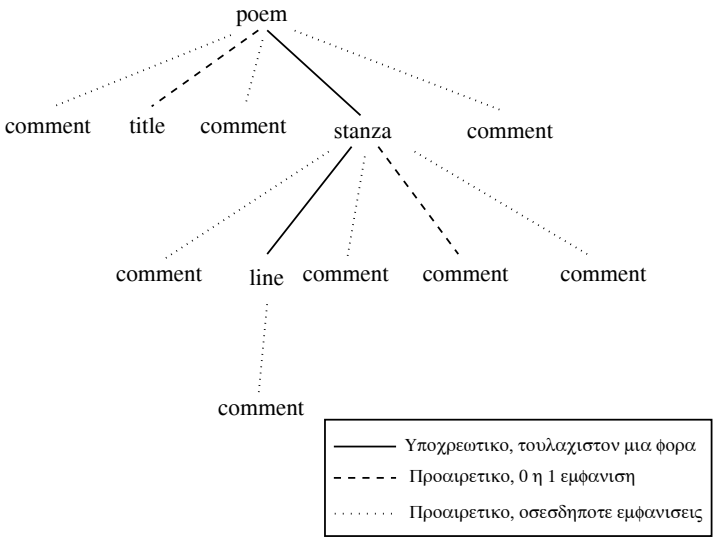
Καθορισμός Τύπου Εγγράφου → (*Δήλωση Στοιχείου*
| *Δήλωση Γνωρίσματος*
| *Δήλωση Οντότητας*
| *Δήλωση Σχολίου*
| *Δήλωση Μαρκαρισμένης Ενότητας*
| *Δήλωση Σημείωσης*)*

Ενα παράδειγμα Δήλωσης Τύπου Εγγράφου, σύμφωνα με το πρότυπο, είναι το ακόλουθο το οποίο περιγράφει την δομή που μπορούν να έχουν τα ποιήματα. Στο συγκεκριμένο παράδειγμα το Ονομα Καθορισμού Τύπου Εγγράφου είναι poem και ο καθορισμός τύπου εγγράφου περιέχει δηλώσεις στοιχείων στις γραμμές (3),(5),(6),(7),(8), μία δήλωση γνωρισμάτων στην γραμμή (4), και μία δήλωση σχολίου στη γραμμή (1).

Παράδειγμα 2.4 :

```
(1)<!-- This is a DTD for poems -->
(2)<!DOCTYPE poem [
(3)<!ELEMENT poem - - (title?, stanza+) +(comment)>
(4)<!ATTLIST poem status (draft | final) draft>
(5)<!ELEMENT title - - CDATA>
(6)<!ELEMENT stanza - - (line+,comment?)>
(7)<!ELEMENT line - - (#PCDATA)>
(8)<!ELEMENT comment - - (#PCDATA) -(comment)>
(9)]>
```

Σύμφωνα με την παραπάνω δήλωση ένα ποίημα αποτελείται από ένα προαιρετικό τίτλο και μία ή περισσότερες στροφές (γραμμή 3). Η στροφή αποτελείται από μία ή περισσότερες γραμμές και ένα προαιρετικό σχόλιο στο τέλος κάθε στροφής (γραμμή 6). Επίσης σχόλια μπορούνε να εμφανίζονται και σε οποιοδήποτε μέρος του ποιήματος, χωρίς περιορισμό (γραμμή 3). Αυτό που απαγορεύεται είναι η εμφάνιση σχολίων μέσα σε σχόλια (γραμμή 8). Τα παραπάνω απεικονίζονται και στο Σχήμα 2.3 στο οποίο παριστάνεται με γραφικό τρόπο η δομή ενός ποιήματος.



Σχήμα 2.3: Γραφική παράσταση της δομής του καθορισμού τύπου εγγράφου για ποίημα

Στην συνέχεια θα γίνει μία περιγραφή των δομικών στοιχείων της δήλωσης τύπου εγγράφου και θα γίνει κατανοητό το πώς όλη αυτή η πληροφορία, για την δομή του ποιήματος, αποτυπώνεται στο παραπάνω παράδειγμα με την χρήση της SGML.

2.4.3 Δήλωση Στοιχείου

Το στοιχείο (ELEMENT) είναι το δομικό συστατικό ενός εγγράφου. Κάθε στοιχείο δηλώνεται στο καθορισμό τύπου εγγράφου μέσω μίας δήλωσης στοιχείου. Σκοπός της δήλωσης στοιχείου είναι να αποδώσει σε κάθε στοιχείο ένα μοναδικό όνομα και να καθορίσει το είδος των περιεχομένων του στοιχείου. Ενα στοιχείο μπορεί να περιέχει αδόμητα δεδομένα με τη μορφή ενός αλφαριθμητικού και/ή άλλα στοιχεία. Στο Παράδειγμα 2.1 τα περιεχόμενα του στοιχείου $line$ είναι αλφαριθμητικού τύπου, ενώ τα περιεχόμενα του στοιχείου $stanza$ είναι ένα ή περισσότερα στοιχεία $line$. Το περιεχόμενο του κάθε στοιχείου καθορίζεται μέσω του μοντέλου περιεχομένων του στοιχείου, μία

αναπαράσταση η οποία πλησιάζει τις κανονικές εκφράσεις. Στις παραγράφους που ακολουθούν θα περιγραφούν τα δύο είδη μοντέλων περιεχομένων τα οποία υπάρχουν, το Αλφαριθμητικό Μοντέλο Περιεχομένων και το Μοντέλο Περιεχομένων Στοιχείων.

2.4.3.1 Αλφαριθμητικό Μοντέλο Περιεχομένων

Υπάρχουν τρία είδη αλφαριθμητικών περιεχομένων, με ονόματα RCDATA (Replacable Character DATA), CDATA (Character DATA), EMPTY. Το κάθε ένα από αυτά τα μοντέλα καθορίζει κατά τρόπο αυστηρό το τι ακριβώς θα πρέπει να αναγνωρίζει και τι θα πρέπει να αγνοεί ένας λεξικογραφικός αναλυτής κατά την διάρκεια της λεξικογραφικής ανάλυσης των περιεχομένων ενός στοιχείου εγγράφου. Στην περίπτωση του EMPTY, όπως και το όνομα φανερώνει, το περιεχόμενο του στοιχείου πρέπει να είναι άδειο. Μόνο χαρακτήρες κενού και αλλαγής γραμμής επιτρέπονται. Οσον αφορά τα RCDATA και CDATA επιτρέπονται όλοι οι χαρακτήρες. Ειδικότερα στην περίπτωση του RCDATA επιτρέπονται και αναφορές σε οντότητες και ο λεξικογραφικός αναλυτής αντικαθιστά την αναφορά με το κείμενο στο οποίο αυτή αναφέρεται, ενώ η ίδια αναφορά σε ένα περιεχόμενο τύπου CDATA θα θεωρηθεί ως κείμενο, δεν αναγνωρίζεται ως τέτοια, και καμία αντικατάσταση δεν λαμβάνει χώρα.

Εδώ θα πρέπει να σημειωθεί ότι υπάρχει και ένα τέταρτο είδος αλφαριθμητικών περιεχομένων με το όνομα PCDATA (Parsed Character DATA) αλλά παρουσιάζει αρκετές διαφορές από τα προηγούμενα είδη για αυτό και δεν ομαδοποιείται μαζί τους. Μία από τις βασικές διαφορές είναι ότι, αν τα περιεχόμενα ενός στοιχείου έχουν οριστεί να είναι του τύπου PCDATA τότε ο λεξικογραφικός αναλυτής αναγνωρίζει όλες τις ετικέτες αρχής και τέλους τις οποίες συναντά στο κείμενο. Αν όμως τα περιεχόμενα του στοιχείου έχουν οριστεί να είναι του τύπου RCDATA ή CDATA, τότε η μόνη ετικέτα που αναγνωρίζεται από το λεξικογραφικό αναλυτή μέσα στο κείμενο είναι η ετικέτα τέλους του στοιχείου μέσα στο μοντέλο περιεχομένων του οποίου βρισκόμαστε, ή η ετικέτα τέλους στοιχείου μέσα στο οποίο το δεδομένο στοιχείο είναι ενθυλακωμένο. Ολες οι άλλες ετικέτες αντιμετωπίζονται σαν κανονικό κείμενο. Αυτός είναι και ο λόγος για τον οποίο δεν μπορούν να υπάρχουν φωλιασμένα στοιχεία μέσα σε στοιχεία που έχουν μοντέλο περιεχομένων RCDATA και CDATA. Από την άλλη μεριά ενθυλάκωση στοιχείων επιτρέπεται στο αλφαριθμητικό είδος περιεχομένων PCDATA και για αυτό το είδος αυτό των περιεχομένων θεωρείται ειδική περίπτωση του μοντέλου περιεχομένων στοιχείων.

2.4.3.2 Μοντέλο Περιεχομένων Στοιχείων

Στην περίπτωση που το μοντέλο περιεχομένων ενός στοιχείου περιέχει άλλα στοιχεία τότε κάθε ένα από αυτά τα στοιχεία χαρακτηρίζεται από δύο τελεστές: *τον τελεστή εμφάνισης* και *τον τελεστή σύνδεσης*.

- Τελεστής Εμφάνισης
  Δηλώνει πόσες φορές μπορεί να εμφανίζεται ένα στοιχείο στη λίστα περιεχομένων. Υπάρχουν τρεις τελεστές, το σύμβολο συν (+), το σύμβολο αστερίσκος (*), το σύμβολο ερωτηματικό (?).
  Το σύμβολο συν (+) σημαίνει ότι μπορούν να υπάρξουν μία ή περισσότερες εμφανίσεις του στοιχείου, το σύμβολο αστερίσκος (*) σημαίνει ότι μπορεί το στοιχείο να απουσιάζει ή να υπάρξουν μία ή περισσότερες εμφανίσεις του. Το σύμβολο ερωτηματικό (?) σημαίνει ότι το στοιχείο μπορεί είτε να απουσιάζει είτε να εμφανίζεται μόνο μία φορά. Τέλος αν ο τελεστής εμφάνισης απουσιάζει τότε το εν λόγω στοιχείο θα πρέπει να εμφανίζεται μία μόνο φορά υποχρεωτικά.

- Τελεστής Σύνδεσης
  Δηλώνει τη σειρά με την οποία μπορούν να εμφανίζονται τα στοιχεία. Υπάρχουν τρεις τελεστές σύνδεσης, το σύμβολο κόμμα (,), το σύμβολο κάθετης μπάρας (|), το σύμβολο ampersand (&).
  Το σύμβολο κόμμα (,) σημαίνει ότι τα στοιχεία που συνδέει πρέπει να εμφανίζονται με τη σειρά που εμφανίζονται στο μοντέλο περιεχομένων Το σύμβολο κάθετης μπάρας (|) σημαίνει ότι μόνο ένα από τα στοιχεία που συνδέει μπορεί να εμφανιστεί. Τέλος το σύμβολο ampersand (&) σημαίνει ότι τα στοιχεία που συνδέει πρέπει να εμφανιστούν όλα αλλά με οποιαδήποτε σειρά.

Μέσα σε ένα μοντέλο περιεχομένων μπορεί να υπάρχει μόνο ένας από τους τελεστές σύνδεσης. Οπως προαναφέρθηκε ο τελεστής σύνδεσης συνδέει δύο στοιχεία αλλά μπορεί να συνδέει και ένα στοιχείο με ένα υπομοντέλο ή ακόμα και δύο υπομοντέλα. Μπορούν να εμφανίζονται υπομοντέλα περιεχομένων, με τα δικά τους στοιχεία, τελεστή σύνδεσης κοκ. και αυτό μπορεί να φτάσει σε οποιοδήποτε βάθος φωλιάσματος. Το παράδειγμα που ακολουθεί δείχνει μια δήλωση στοιχείου με κάπως πιο πολύπλοκο μοντέλο από αυτά που έχουν δειχθεί μέχρι τώρα

Παράδειγμα 2.5 *:*

$$< !ELEMENT\ a\ -\ -\ ((b,\ c)+\ |\ (d,\ e,\ f))\ >$$

Στο παράδειγμα τα περιεχόμενα του στοιχείου a μπορεί να είναι είτε τα στοιχεία b και c με αυτή την σειρά, για παραπάνω από μία φορές, είτε τα στοιχεία d, e, και f πάλι με αυτή τη σειρά αλλά μία εμφάνιση από το καθένα.

Ειδική περίπτωση του μοντέλου περιεχομένων στοιχείων είναι το περιεχόμενο του προς δήλωση στοιχείου να μπορεί να περιέχει οποιοδήποτε άλλο στοιχείο, γεγονός το οποίο δηλώνεται μέσω της δεσμευμένης λέξης ANY.

Παραγωγή 2.5 *:*

Δήλωση Στοιχείου → *<!ELEMENT Ονομα Στοιχείου,*
(Αλφαριθμητικό Μοντέλο Περιεχομένων
| Μοντέλο Περιεχομένων Στοιχείων)
Εξαιρέσεις

Παραγωγή 2.6 *:*

Αλφαριθμητικό Μοντέλο Περιεχομένων → *RCDATA* | *CDATA* | *EMPTY*

Παραγωγή 2.7 *:*

Μοντέλο Περιεχομένων Στοιχείων → *ANY* | *Μοντέλο Στοιχείων*

Παραγωγή 2.8 *:*

Μοντέλο Στοιχείων → *(Υποστοιχείο,(Τελεστής Σύνδεσης, Υποστοιχείο)*),*
Τελεστής Εμφάνισης

Παραγωγή 2.9 *:*

Υποστοιχείο → *(Ονομα Στοιχείου* | *Υπομοντέλο Στοιχείων), Τελεστής Εμφάνισης*
| *PCDATA*

Μία έννοια η οποία ορίζεται στο πρότυπο, και η οποία έχει άμεση σχέση με το μοντέλο περιεχομένων ενός στοιχείου, είναι οι Εξαιρέσεις. Στις παραγράφους που ακολουθούν θα μελετηθεί η έννοια των εξαιρέσεων και το πως αυτές επηρεάζουν τον μοντέλο περιεχομένων ενός στοιχείου.

2.4.3.3 Εξαιρέσεις

Ενα σημείο του προτύπου, για το οποίο πλήθος συζητήσεων έχουν γίνει, είναι αυτό των Εξαιρέσεων (Exceptions) ενός μοντέλου περιεχομένων. Στα περισσότερα είδη εγγράφων υπάρχουν κάποια στοιχεία τα οποία θα πρέπει να εμφανίζονται σε οποιοδήποτε σημείο του εγγράφου, όπως παραδείγματος χάριν, τα σχόλια. Στο παράδειγμα μας με το ποίημα θα ήταν επιθυμητό να μπορούμε να προσθέσουμε ένα σχόλιο σε οποιοδήποτε

σημείο του ποιήματος είτε σε κάποιο στίχο, είτε στον τίτλο κτλ. Για να προσφέρει αυτή την δυνατότητα το πρότυπο εισάγει την έννοια των εξαιρέσεων. Διακρίνει μάλιστα τις εξαιρέσεις σε δύο είδη, τους Εγκλεισμούς (Inclusions) και τους Αποκλεισμούς (Exclusions). Σε ένα, οποιοδήποτε, μοντέλο περιεχομένων μπορούν να υπάρχουν δύο λίστες. Στην λίστα των εγκλεισμών, η οποία δηλώνεται με ένα συν (+) μετά το τέλος της δήλωσης του μοντέλου περιεχομένων, αναφέρονται τα ονόματα των στοιχείων τα οποία μπορούν να εμφανιστούν οπουδήποτε μέσα στο μοντέλο περιεχομένων του προς δήλωση στοιχείου. Αντίστοιχα στην λίστα των αποκλεισμών, η οποία δηλώνεται με ένα μείον (-) μετά το τέλος της δήλωσης του μοντέλου περιεχομένων, αναφέρονται τα ονόματα των στοιχείων που αποκλείεται η εμφάνιση τους. Στη γραμμή 3 του Παραδείγματος 4.8, κάνοντας χρήση των εξαιρέσεων καθορίστηκε ότι σε οποιοδήποτε σημείο ενός ποιήματος μπορούν να υπάρξουν σχόλια. Ενα χαρακτηριστικό των εξαιρέσεων, το οποίο δεν αναφέρθηκε μέχρι τώρα, είναι ότι οι εξαιρέσεις κληρονομούνται από όλα τα στοιχεία που βρίσκονται στο υποδέντρο της δομής του εγγράφου με ριζικό το στοιχείο στο οποίο ορίζονται οι εξαιρέσεις. Αυτό φαίνεται και στο Σχήμα 2.3 το οποίο αναπαριστά τη δομή αυτή. Ετσι στον καθορισμό τύπου εγγράφου του Παραδείγματος 4.8 χρειάστηκε μόνο στην γραμμή 3 να οριστεί ότι οπουδήποτε μέσα στο στοιχείο $poem$ μπορούν να εμφανιστούν σχόλια και η εξαίρεση αυτή κληρονομήθηκε και από το στοιχείο $stanza$ και από το στοιχείο $line$ τα οποία στο δέντρο του Σχήματος 2.3 εμφανίζονται ως απόγονοι του στοιχείου $poem$.

Παρόλο που οι εξαιρέσεις μας βοηθούν να περιγράψουμε πιο πολύπλοκα έγγραφα, εισάγουν ένα μεγάλο πρόβλημα, αυτό της κατασκευής γρήγορων και αποδοτικών συντακτικών αναλυτών που μπορούν να αναγνωρίζουν τα έγγραφα που δημιουργούνται. Επίσης έχει αμφισβητηθεί από πολλούς [COM] η χρησιμότητα των εξαιρέσεων με αποτέλεσμα το πρότυπο XML [Con97] το οποίο αποτελεί συνέχεια της SGML να μην υποστηρίζει εξαιρέσεις. Ανεξαρτήτως των αμφισβητήσεων οι οποίες έχουν διατυπωθεί, η ιδέα των εξαιρέσεων δεν μπορεί να απορριφθεί εξ ολοκλήρου καθώς έχει αποδειχθεί [Eng97] ότι για ένα καθορισμό τύπου εγγράφου με εξαιρέσεις δεν μπορεί, γενικά, να γραφτεί ένα ισοδύναμος καθορισμός χωρίς εξαιρέσεις, γεγονός το οποίο αποδεικνύει την εκφραστική ισχύ που αυτές προσφέρουν.

Τελειώνοντας την περιγραφή της δήλωσης στοιχείου θα ήταν παράλειψη να μην γίνει λόγος για το πρόβλημα της αμφισημίας[ASU86] στο μοντέλο περιεχομένων της δήλωσης στοιχείου, και τα προβλήματα που αυτή εισάγει [BKW93, BK94b, BK94a, BKW94]. Το φαινόμενο της αμφισημίας του μοντέλου περιεχομένων ενός στοιχείου μπορεί να εμφανιστεί μόνο όταν στην δήλωση στοιχείου, το οριζόμενο στοιχείο έχει

μοντέλο περιεχομένων στοιχείων. Επίσης πρέπει να σημειωθεί ότι υπάρχουν δύο είδη αμφισημίας, η αμφισημία ως προς τα σύμβολα και η αμφισημία ως προς τους τελεστές. Ακολούθως περιγράφονται τα δύο είδη αυτά είδη.

2.4.3.4 Αμφισημία ως προς τα σύμβολα

Το παράδειγμα που ακολουθεί παρουσιάζει τον ορισμό ενός στοιχείου με αμφισημία ως προς τα σύμβολα στο μοντέλο περιεχομένων του, και θα βοηθήσει στην καλύτερη εξήγηση του προβλήματος. Το παράδειγμα περιέχει μία δήλωση στοιχείου, όπως αυτή θα εμφανιζόταν στο καθορισμό τύπου εγγράφου, καθώς και ένα απόσπασμα από μία πιθανή περίπτωση εγγράφου.

Παράδειγμα 2.6 *:*

<!ELEMENT a - - *(b, c?, b?)* >*

(1)<a>

*(2) περιεχόμενο στοιχείου b *

(3)<c> περιεχόμενο στοιχείου c </c>

*(4) περιεχόμενο στοιχείου b *

(5)...

(6)

Απο τη δήλωση του στοιχείου εξάγεται το συμπέρασμα ότι στα στιγμιότυπα ενός καθορισμού τύπου εγγράφου, σε περίπτωση συνάντησης της ετικέτας αρχής του στοιχείου a αυτό που πρέπει να ακολουθήσει είναι ένα στοιχείο b στην συνέχεια προαιρετικά ένα στοιχείο c, κατόπιν ένα προαιρετικό στοιχείο b, και όλη η ακολουθία, λόγω του αστερίσκου της δήλωσης, μπορεί να επαναληφθεί οσεσδήποτε φορές. Οταν ο συντακτικός αναλυτής της περίπτωσης εγγράφου συναντήσει το δεύτερο στοιχείο b (γραμμή 4), και με την προϋπόθεση ότι δεν θα δει τι ακολουθεί μετά (no lookahead), δεν μπορεί να καταλάβει αν πρόκειται για το δεύτερο b του μοντέλου ή αν το δεύτερο b παραλείφθηκε, και χρησιμοποιώντας τον τελεστή εμφάνισης αστερίσκο, επαναλαμβάνεται το πρώτο b του μοντέλου. Για να γίνει το παράδειγμα πιο κατανοητό θα μαρκαριστούν [Glu61][B$^+$71] [BS86] τα σύμβολα του μοντέλου περιεχομένων έτσι ώστε να μπορούν να διακριθούν οι δύο εμφανίσεις του στοιχείου b. Το μοντέλο μπορεί τώρα να γραφεί ως $(b_1,\ c_1?,\ b_2?)*$. Τότε η *λέξη* bcb η οποία αντιστοιχεί στο περιεχόμενο του στοιχείου a μπορεί να παραχθεί από το μοντέλο με δύο τρόπους, είτε ως $b_1c_1b_2$, είτε ως $b_1c_1b_1$. Αυτού

του είδους η αμφισημία χαρακτηρίζεται, όπως προαναφέρθηκε, ως αμφισημία ως προς τα σύμβολα, για να την διακρίνουμε από ένα δεύτερο είδος αμφισημίας, την αμφισημία ως προς τους τελεστές η οποία θα μελετηθεί στη συνέχεια.

2.4.3.5 Αμφισημία ως προς τους τελεστές

Ενα παράδειγμα αμφισημίας ως προς τους τελεστές είναι το ακόλουθο. Το παράδειγμα περιέχει μία δήλωση στοιχείου, όπως αυτή θα εμφανιζόταν στο καθορισμό τύπου εγγράφου, καθώς και ένα απόσπασμα από μία πιθανή περίπτωση εγγράφου.

Παράδειγμα 2.7 *:*

<!ELEMENT a - - (b+ | c+)+ >

<a>
* περιεχόμενο στοιχείου b *
* περιεχόμενο στοιχείου b *
**

Στο παράδειγμα ορίζεται ότι μετά το στοιχείο a μπορεί να ακολουθήσει μία σειρά από b ή μία σειρά από c και όλο το μοντέλο, λόγω του εξωτερικού συν (+), μπορεί να επαναληφθεί οσεσδήποτε φορές. Στο απόσπασμα από μία πιθανή περίπτωση εγγράφου, φαίνεται ένα στοιχείο a να έχει ως περιεχόμενο του δύο εμφανίσεις του στοιχείου b. Η αμφισημία του παραδείγματος έγκειται στο γεγονός ότι δεν είναι δυνατόν να γίνει αντιληπτό αν τα δύο b οφείλονται στην διπλή εφαρμογή του συν στο εσωτερικό του μοντέλου και στην μονή εφαρμογή του εξωτερικού συν ή στην μονή εφαρμογή του εσωτερικού συν και στην διπλή εφαρμογή του εξωτερικού. Το πρότυπο θεωρεί ότι ένα μοντέλο είναι αμφίσημο **μόνο** όταν αυτό παρουσιάζει αμφισημία ως προς τα σύμβολα. Ετσι ένα αναλυτής πιστοποίησης δεν θα πρέπει να αναφέρει το δεύτερο είδος αμφισημίας.

Τελειώνοντας με την περιγραφή των αμφισημιών στα μοντέλα περιεχομένων κρίνεται απαραίτητο να σχολιαστεί η γραμματική η οποία παράγεται από τα μοντέλα περιεχομένων στοιχείων, γραμματική η οποία καθορίζει την δομή των περιπτώσεων εγγράφων, και η οποία σύμφωνα με το πρότυπο πρέπει να είναι LL(1). Η ερμηνεία που δίνει το πρότυπο για τις LL(1) γραμματικές είναι ότι κάθε φορά που ο συντακτικός αναλυτής βλέπει μία ετικέτα αρχής θα πρέπει να ξέρει σε ποιο σημείο του μοντέλου αντιστοιχεί χωρίς να χρειάζεται να δει επόμενες ετικέτες. Περιορίζοντας τις οριζόμενες γραμματικές στη κλάση γραμματικών LL(1), και σε συνδυασμό με την απουσία αμφισημίας στα

μοντέλα περιεχομένων, οι δημιουργοί του προτύπου αποσκοπούν στο να εξασφαλίσουν τον ορισμό μη αμφίσημων γραμματικών. Η αμφισημία της γραμματικής όμως είναι ανεξάρτητη από την αμφισημία των επιμέρους μοντέλων περιεχομένων τα οποία την ορίζουν, αφού υπάρχει πιθανότητα μη αμφίσημα μοντέλα περιεχομένων να ορίσουν μία αμφίσημη γραμματική όπως θα δειχθεί στην συνέχεια. Οι ακόλουθες παρατηρήσεις θα αποσαφηνίσουν τα παραπάνω. Η πρώτη παρατήρηση η οποία πρέπει να γίνει είναι ότι οι γραμματικές οι οποίες ορίζονται, δεν μπορούν να χαρακτηριστούν ως LL(1), και αυτό γιατί ορισμένες δυνατότητες του προτύπου, όπως οι εξαιρέσεις και ο τελεστής σύνδεσης ampersand (&), δεν υπάρχουν στις LL(1) γραμματικές, αλλά ούτε συναντώνται στη θεωρία γλωσσών γενικότερα. Μια δεύτερη παρατήρηση είναι ότι η δυνατότητα, την οποία προσφέρει το πρότυπο, για παράλειψη ετικετών αρχής και τέλους οδηγεί στο προαναφερθέν πρόβλημα του ορισμού αμφίσημης γραμματικής από ένα σύνολο μη αμφίσημων μοντέλων περιεχομένων. Αυτό φαίνεται και από το ακόλουθο απλό παράδειγμα.

Παράδειγμα 2.8 *:*

<!ELEMENT a - - (b | c) >

<a>
<> περιεχόμενο στοιχείου b ή c </>
**

Στο παράδειγμα, το μοντέλο περιεχομένων του στοιχείου a είναι σαφώς μη αμφίσημο και ως προς τα σύμβολα και ως προς τους τελεστές. Στη ακόλουθη περίπτωση εγγράφου έχει παραλειφθεί η ετικέτα αρχής και τέλους του στοιχείου που βρίσκεται στο περιεχόμενο του στοιχείου a. Ποιο είναι όμως το στοιχείο αυτό, το b ή το c; Ενα μοντέλο περιεχομένων, το οποίο περνάει επιτυχώς τον έλεγχο για αμφισημίες, παρήγαγε μη αμφίσημη γραμματική από τις παρενέργειες που δημιουργεί η παράλειψη ετικετών. Το λογικό ερώτημα που τίθεται τότε είναι, γιατί να μην ελεγχθεί και η γραμματική για αμφισημίες, όπως έγινε με τα μοντέλα περιεχομένων. Η απάντηση είναι απλή. Γιατί η απόφαση της ύπαρξης ή μη αμφισημίας σε μια γραμματική είναι **μη επιλύσιμο πρόβλημα** [HU79].

2.4.4 Δήλωση Γνωρίσματος

Κάθε στοιχείο μπορεί να έχει ένα ή περισσότερα γνωρίσματα. Η δήλωση γνωρισμάτων ξεκινάει με την αναφορά του ονόματος του στοιχείου του οποίου τα γνωρίσματα πρόκειται να οριστούν. Κατόπιν ακολουθεί η λίστα με τους ορισμούς των γνωρισμάτων.

Κάθε γνώρισμα έχει ένα όνομα, ένα τύπο, και τέλος ένα πεδίο που καθορίζει το τι πρέπει να γίνει στην περίπτωση που ο δημιουργός της περίπτωσης εγγράφου δεν δώσει τιμή στο γνώρισμα.

Παραγωγή 2.10 *:*

Δήλωση Γνωρίσματος → *<!ATTLIST Ονομα Στοιχείου Λίστα Γνωρισμάτων>*

Παραγωγή 2.11 *:*

Λίστα Γνωρισμάτων → *(Γνώρισμα)+*

Παραγωγή 2.12 *:*

Γνώρισμα → *Ονομα Γνωρίσματος,*
Τύπος Γνωρίσματος,
Τύπος Εμφάνισης Γνωρίσματος

Τα γνωρίσματα χρησιμοποιούνται για να περιγράψουν ιδιότητες των στοιχείων οι οποίες δεν είναι προφανείς από το περιεχόμενο τους. Ετσι όπως βλέπουμε και από το παράδειγμα ορισμού του τύπου εγγράφου poem το στοιχείο poem έχει ένα γνώρισμα με το όνομα status το οποίο φανερώνει αν το ποίημα βρίσκεται στην τελική του μορφή ή αν ακόμα ο συγγραφέας κάνει διορθώσεις σε αυτό. Μία τέτοια πληροφορία μπορεί να αποδειχθεί ιδιαίτερα χρήσιμη σε ένα υποτιθέμενο πρόγραμμα το οποίο επεξεργάζεται τα ποιήματα αφού μπορεί σε περίπτωση που χρειαστεί να παραχθεί μία εκτύπωση αυτών, να επιλέξει μόνο αυτά τα οποία βρίσκονται στην τελική τους μορφή. Μια συνήθης χρήση των γνωρισμάτων είναι αυτή της αναφοράς από ένα σημείο του εγγράφου σε κάποιο άλλο σημείο αυτού. Υπάρχει η δυνατότητα, μέσω ειδικού τύπου γνωρίσματος, απόδοσης ταυτότητας στο στοιχείο στο οποίο αναφέρεται το γνώρισμα έτσι ώστε τα υπόλοιπα στοιχεία να μπορούν να αναφέρονται σε αυτό. Η αναφορά σε κάποιο άλλο στοιχείο γίνεται και αυτή μέσω ειδικού τύπου γνωρίσματος. Το πρότυπο προσφέρει δεκαέξι τύπους γνωρισμάτων, και κάθε τύπος έχει το αντίστοιχο πεδίο τιμών. Οι τύποι αυτοί είναι CDATA, ENTITY, ENTITIES, ID, IDREF, IDREFS, NAME, NAMES, NMTOKEN, NMTOKENS, NUMBER, NUMBERS, NUTOKEN, NUTOKENS, NOTATION, NAME TOKEN GROUP.

Οσον αφορά τον Τύπο Εμφάνισης Γνωρίσματος το πρότυπο προσφέρει πέντε δυνατές επιλογές. Η πρώτη επιλογή REQUIRED ορίζει ότι πρέπει οπωσδήποτε να αποδοθεί τιμή στο γνώρισμα σε κάθε εμφάνιση του στοιχείου στο οποίο ανήκει. Η δεύτερη επιλογή IMPLIED ορίζει ότι το γνώρισμα είναι προαιρετικό. Η τρίτη επιλογή CURRENT ορίζει ότι το γνώρισμα σε περίπτωση που δεν του αποδοθεί τιμή θα πρέπει να πάρει την ίδια

τιμή με την τιμή που πήρε το πιο πρόσφατο γνώρισμα του ιδίου Τύπου Γνωρίσματος ανεξαρτήτως σε πιο στοιχείο ανήκε. Η τέταρτη επιλογή CONREF ορίζει ότι σε περίπτωση που αποδοθεί τιμή στο γνώρισμα το μοντέλο περιεχομένων του στοιχείου πρέπει να μείνει κενό, ενώ στην περίπτωση μη απόδοσης τιμής το μοντέλο περιεχομένων είναι αυτό που ορίστηκε στο καθορισμό τύπου εγγράφου. Τέλος μπορεί ο δημιουργός του καθορισμού τύπου εγγράφου να ορίσει μία τιμή η οποία ισχύει στην περίπτωση που δεν αποδοθεί τιμή στο γνώρισμα.

Η εκχώρηση τιμών στα γνωρίσματα ενός στοιχείου γίνεται στο εσωτερικό της ετικέτας αρχής του στοιχείου. Ετσι αν ο δημιουργός ενός ποιήματος επιθυμεί να δηλώσει ότι το ποίημα το οποίο επεξεργάζεται βρίσκεται στην τελική του μορφή, αποδίδοντας έτσι στο γνώρισμα status του στοιχείου poem την τιμή final, θα πρέπει να γράψει μία δήλωση της μορφής

Παράδειγμα 2.9 *:*

<poem status=final>

Οταν για ένα στοιχείο έχουν δηλωθεί παραπάνω από ένα γνωρίσματα τότε η εκχώρηση των τιμών μπορεί να γίνει με οποιαδήποτε σειρά, δηλαδή δεν υπάρχει η έννοια της σειράς μεταξύ των γνωρισμάτων. Εκχώρηση τιμών δεν επιτρέπεται στο εσωτερικό των ετικετών τέλους.

2.4.5 Δήλωση Οντότητας

Μία οντότητα είναι ένα κομμάτι κειμένου στο οποίο έχει αποδοθεί ένα όνομα. Το κομμάτι αυτό κειμένου μπορεί να είναι είτε ένα αλφαριθμητικό μικρού ή μεγάλου μήκους είτε ένα σύνολο χαρακτήρων το οποίο δεν μπορεί να παραχθεί εύκολα με το διαθέσιμο πληκτρολόγιο είτε ένα άλλο αρχείο κειμένου το οποίο βρίσκεται κάπου στο σύστημα αρχείων. Το όνομα της οντότητας χρησιμοποιείται όταν γίνεται αναφορά σε αυτή. Το πρότυπο χωρίζει της οντότητες σε δύο μεγάλες κατηγορίες, τις παραμετρικές οντότητες και τις γενικές οντότητες.

Παραγωγή 2.13 *:*

Δήλωση Οντότητας $\rightarrow$ *<!ENTITY Ονομα Οντότητας Κείμενο Οντότητας >*

Οι παραμετρικές οντότητες χρησιμοποιούνται μόνο μέσα στα πλαίσια της Δήλωσης SGML, από τον δημιουργό του Καθορισμού Τύπου Εγγράφου, και ο δημιουργός της περίπτωσης εγγράφου δεν χρειάζεται, και τις περισσότερες φορές δεν πρέπει, να γνωρίζει την ύπαρξη τους.

Οι γενικές οντότητες είναι αυτές που κατά κύριο λόγο χρησιμοποιούνται κυρίως για την ονομασία κομματιών κειμένου που χρησιμοποιούνται συχνά. Ετσι ο δημιουργός της περίπτωσης εγγράφου γλιτώνει χρόνο κατά την πληκτρολόγηση αφού αντί να ξαναγράφει το ίδιο κείμενο απλώς κάνει μια αναφορά σε αυτό και είναι ευθύνη του λεξικογραφικού αναλυτή που θα διαβάσει το έγγραφο να αντικαταστήσει την αναφορά με το ίδιο το κείμενο στο οποίο αναφέρεται.

2.4.6 Δήλωση Σχολίου

Μία δήλωση σχολίου μπορεί να εμφανιστεί σε οποιοδήποτε σημείο τόσο κατά την δήλωση τύπου εγγράφου όσο και κατά την συγγραφή της περίπτωσης εγγράφου.

Παραγωγή 2.14 *:*

Δήλωση Σχολίου $\rightarrow$ *<!-- Κείμενο Σχολίου -->*

2.4.7 Δήλωση Μαρκαρισμένης Ενότητας

Το πρότυπο προσφέρει την δυνατότητα μαρκαρίσματος μίας ενότητας κειμένου και τον χαρακτηρισμό αυτής με μία από πέντε δεσμευμένες λέξεις έτσι ώστε το λογισμικό το οποίο χειρίζεται το έγγραφο να ξέρει πως να χειριστεί την ενότητα αυτή. Οι πέντε δυνατοί χαρακτηρισμοί είναι CDATA, IGNORE, INCLUDE, RCDATA, TEMP. Μία ενότητα με τον χαρακτηρισμό IGNORE θα πρέπει να αγνοηθεί από το λογισμικό χειρισμού του εγγράφου ενώ μία ενότητα με χαρακτηρισμό INCLUDE ή TEMP όχι. Από την άλλη μεριά ενότητες μαρκαρισμένες ως CDATA ή RCDATA θα πρέπει να θεωρηθούν ότι αποτελούνται από χαρακτήρες και οι περιορισμοί που ισχύουν για αυτές είναι οι ίδιοι με τους περιορισμούς οι οποίοι ισχύουν για το μοντέλο περιεχομένων ενός στοιχείου με τους ίδιους χαρακτηρισμούς. Η δήλωση μαρκαρισμένης ενότητας χρησιμοποιείται στο επίπεδο της δήλωσης SGML αλλά και στο επίπεδο της περίπτωσης εγγράφου. Εκεί μάλιστα είναι ιδιαίτερα χρήσιμη για την διατήρηση των εκδόσεων (versioning) ενός εγγράφου. Συγκεκριμένα είναι δυνατόν να υπάρχουν μέσα στο ίδιο αρχείο αποθηκευμένες όλες οι εκδόσεις από τις οποίες πέρασε ένα κείμενο και κάθε φορά με κατάλληλο μηχανισμό μία μόνο έκδοση να είναι ενεργή, συνήθως η τελευταία, χρησιμοποιώντας το χαρακτηρισμό INCLUDE για αυτήν, και όλες οι υπόλοιπες ανενεργές χαρακτηρίζοντάς τις ως IGNORE κομμάτια κειμένου.

Παραγωγή 2.15 *:*

Δήλωση Μαρκαρισμένης Ενότητας → *<![Χαρακτηρισμός Ενότητας [Κείμενο Ενότητας]]>*

2.4.8 Δήλωση Σημείωσης

Πολλές φορές στις περιπτώσεις εγγράφων εκτός από κανονικό κείμενο ο δημιουργός μπορεί να ενσωματώσει ένα αρχείο εικόνας ή ήχου, ένα αρχείο LaTex ή κάτι παρόμοιο, το οποίο δεν πρέπει να αντιμετωπιστεί ως κανονικό κείμενο SGML ούτε από το συντακτικό και λεξικογραφικό αναλυτή ούτε από οποιοδήποτε άλλο πρόγραμμα το οποίο χειρίζεται το έγγραφο. Είναι επιθυμητό, με κάποιο τρόπο, να φανεί μέσα στο κείμενο ότι, παραδείγματος χάριν, το περιεχόμενο του στοιχείου picture δεν είναι χαρακτήρες αλλά ένα δυαδικό αρχείο και ότι είναι ευθύνη ενός πακέτου γραφικών να χειριστεί αυτό το αρχείο. Για να επιτευχθεί αυτό στην SGML χρησιμοποιούνται οι σημειώσεις. Θα πρέπει ο δημιουργός του καθορισμού τύπου εγγράφου να καθορίσει, χρησιμοποιώντας δηλώσεις σημειώσεων, τους διάφορους τύπους δεδομένων που μπορούν να συναντηθούν μέσα στις περιπτώσεις εγγράφων και πιθανώς να χρειαστεί να καθορίσει και το πρόγραμμα το οποίο θα αναλάβει την διαχείριση των δεδομένων αυτών.

Παραγωγή 2.16 *:*

Δήλωση Σημείωσης → *<!NOTATION Ονομα Σημείωσης, Καθορισμός Σημείωσης >*

2.5 Περίπτωση Εγγράφου

Στο τμήμα αυτό υπάρχει το ”πραγματικό έγγραφο” που ακολουθεί τον τύπο εγγράφου που περιγράφηκε στο τμήμα του προλόγου. Η περίπτωση εγγράφου περιέχει κείμενο, ετικέτες και αναφορές σε οντότητες. Ολες οι απαραίτητες δηλώσεις οι οποίες αφορούν την λογική δομή του εγγράφου έχουν προηγηθεί. Ο δημιουργός της περίπτωσης εγγράφου μπορεί να κάνει τριών ειδών δηλώσεις. Οι δύο από αυτές έχουν ήδη σχολιαστεί και είναι η δήλωση σχολίου και η δήλωση μαρκαρισμένης οντότητας. Η τρίτη δήλωση την οποία συναντάμε μόνο στην περίπτωση εγγράφου είναι η δήλωση οδηγίας επεξεργασίας (processing instruction).

2.5.1 Δήλωση Οδηγίας Επεξεργασίας

Η δήλωση οδηγίας επεξεργασίας είναι ένας τρόπος ενσωμάτωσης στο κείμενο, οδηγιών για την φυσική δομή αυτού. Οι οδηγίες επεξεργασίας μπορούν να θεωρηθούν

σαν σχόλια ειδικής μορφής. Οπως ακριβώς τα σχόλια έχουν νόημα, τις περισσότερες φορές, μόνο για αυτόν ο οποίος τα γράφει, ενώ όλοι οι υπόλοιποι τείνουν να τα αγνοούν, έτσι και οι οδηγίες επεξεργασίας έχει νόημα μόνο για ορισμένα προγράμματα, τα οποία τις αναζητούν μέσα στο κείμενο για να πάρουν οδηγίες για το τι πρέπει να κάνουν. Τα υπόλοιπα προγράμματα απλώς τις αγνοούν. Οπως και τα σχόλια έτσι και οι οδηγίες επεξεργασίας μπορούν να εμφανίζονται σε οποιοδήποτε σημείο του κειμένου.

Παραγωγή 2.17 *:*

$\mathit{Δήλωση\ Οδηγίας\ Επεξεργασίας} \rightarrow \ <?\ \mathit{Κείμενο\ Οδηγίας\ Επεξεργασίας}\ >$

Κεφάλαιο 3

Υπάρχοντα Συστήματα

Στο παρόν κεφάλαιο θα γίνει μία περιγραφή των υπαρχόντων συστημάτων λογισμικού τα οποία υποστηρίζουν το πρότυπο. Σε πρώτη φάση τα συστήματα θα κατηγοριοποιηθούν βάσει των χαρακτηριστικών τους. Στη συνέχεια το ενδιαφέρον θα επικεντρωθεί στους υπάρχοντες συντακτικούς και λεξικογραφικούς αναλυτές, οι οποίοι θα κατηγοριοποιηθούν με τη σειρά τους βάσει της αρχιτεκτονικής τους. Τέλος θα γίνει μία αναφορά στους πιο σημαντικούς από αυτούς τους αναλυτές.

3.1 Κατηγοριοποίηση Συστημάτων

Παρόλο που το πρότυπο δημοσιεύθηκε το 1986 αρκετός καιρός πέρασε μέχρι να βγουν τα πρώτα συστήματα τα οποία να το υποστηρίζουν. Μέχρι το 1993 ελάχιστες εταιρίες είχαν έτοιμα προϊόντα, αν και αυτά ήταν υψηλής ποιότητας, μιας και προέρχονταν από εταιρίες όπως η IBM [IBM], η XEROX [XER], η SoftQuad [Sof]. Πλήθος ερευνητικών προγραμμάτων έχουν ξεκινήσει από τότε από μεγάλους οργανισμούς αλλά και από ιδιώτες όπως, παραδείγματος χάριν, το ιδιωτικό ερευνητικό πρόγραμμα το οποίο οδήγησε στην κατασκευή του SGML Parser (SP), ο οποίος θα περιγραφεί στην συνέχεια. Τα συστήματα που υπάρχουν μπορούν να κατηγοριοποιηθούν βάσει είτε της πλατφόρμας πάνω στην οποία τρέχουν, είτε του επιπέδου υποστήριξης του προτύπου το οποίο προσφέρουν, είτε των λειτουργιών τις οποίες προσφέρουν.

3.1.1 Κατηγοριοποίηση βάσει της πλατφόρμας

Οπως αναφέρθηκε στο προηγούμενο κεφάλαιο ένα από τα κύρια χαρακτηριστικά του προτύπου είναι η ανεξαρτησία αυτού από την αρχιτεκτονική. Παρόλα αυτά, αυτό

δεν σημαίνει ότι τα υπάρχοντα συστήματα διατηρούν αυτήν την ανεξαρτησία. Συνήθως τα συστήματα εξαρτώνται από κάποια αρχιτεκτονική και τίθεται τότε το ερώτημα κατά πόσο αυτά τα συστήματα είναι ικανά να επεξεργαστούν έγγραφα από άλλες πλατφόρμες, έχοντας ως δεδομένο την ύπαρξη τουλάχιστον διαφορετικού συνόλου χαρακτήρων. Αν και το ίδιο το πρότυπο προβλέπει διαφορές μεταξύ συνόλων χαρακτήρων, αυτό δεν είναι δεδομένο για τα συστήματα.

3.1.2 Κατηγοριοποίηση βάσει του επιπέδου υποστήριξης

Σύμφωνα με το πρότυπο, για να θεωρηθεί ένα σύστημα ότι το υποστηρίζει, δεν είναι αναγκαίο να είναι σε θέση να επεξεργαστεί πλήρως ένα έγγραφο SGML, ούτε καν είναι υποχρεωμένο να μπορεί να αναγνωρίζει και να αναφέρει λάθη που πιθανώς υπάρχουν στα έγγραφα. Στο πρότυπο ορίζονται διαφορετικά επίπεδα υποστήριξής του. Υπάρχουν δυνατότητες όπως αυτή της DATATAG ή της OMITTAG οι οποίες είναι προαιρετικές, όπως έχει ήδη αναφερθεί. Επίσης υπάρχουν υπηρεσίες πιστοποίησης (validation services), επίσης προαιρετικές, και καθορίζουν το βαθμό στον οποίο ένα σύστημα μπορεί να αναγνωρίζει και να αναφέρει τα διάφορα είδη λαθών. Θεωρείται λοιπόν επιβεβλημένο κάθε σύστημα να δηλώνει τις δυνατότητες, το σύνολο χαρακτήρων, και τις υπηρεσίες πιστοποίησης που προσφέρει.

3.1.3 Κατηγοριοποίηση βάσει των λειτουργιών

Αρκετές διαφορές εμφανίζονται και στις λειτουργίες των διαφόρων συστημάτων. Υπάρχουν συστήματα που αποτελούνται μόνο από συντακτικό και λεξικογραφικό αναλυτή, άλλα που αποτελούνται από επεξεργαστή κειμένου ειδικά σχεδιασμένο για την γρήγορη συγγραφή εγγράφων SGML, ή συστήματα τα οποία αναλαμβάνουν να μετατρέψουν έγγραφα SGML σε κάποια άλλη μορφή, όπως παραδείγματος χάριν, σε μορφή HTML. Οι μεγάλοι οργανισμοί λογισμικού προσφέρουν, όπως είναι φυσικό, πιο ολοκληρωμένα περιβάλλοντα για την δημιουργία, διαχείριση, αποθήκευση και ανάκτηση εγγράφων SGML.

3.2 Συντακτικοί και λεξικογραφικοί αναλυτές SGML

Ο σκοπός κάθε συντακτικού και λεξικογραφικού αναλυτή SGML (SGML parser) είναι απλός, να αναγνωρίζει τις ετικέτες που εμφανίζονται σε μία περίπτωση εγγράφου. Σύμφωνα με το πρότυπο, δεν είναι υποχρεωμένος ο αναλυτής να αναγνωρίζει λάθη στην

σύνταξη. Η μόνη απαίτηση που τίθεται είναι να έχει την ικανότητα να επεξεργάζεται σωστά ένα έγγραφο, θεωρώντας ότι αυτό δεν έχει λάθη. Από την άλλη μεριά, ένας συντακτικός και λεξικογραφικός αναλυτής πιστοποίησης (validating SGML parser) πρέπει να έχει την δυνατότητα εντοπισμού και αναφοράς συντακτικών λαθών, όταν αυτά υπάρχουν. Προαιρετικά μπορεί να αναφέρει και λάθη που δεν έχουν σχέση με την σύνταξη, όπως παραδείγματος χάριν, αμφισημίες που παρουσιάζονται σε μοντέλα περιεχομένων.

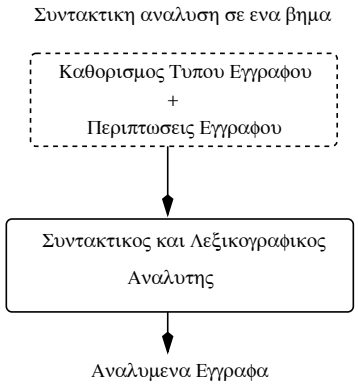
Αυτή τη στιγμή ένας μικρός αριθμός από συντακτικούς και λεξικογραφικούς αναλυτές πιστοποίησης είναι διαθέσιμος, και οι περισσότεροι από αυτούς ανήκουν στην κατηγορία του μη εμπορικού λογισμικού (shareware). Οι αναλυτές οι οποίοι θα μελετηθούν στην συνέχεια, ανήκουν στην κατηγορία του μη εμπορικού λογισμικού αφού, λόγω του υψηλού τους κόστους, οι αναλυτές της κατηγορίας του εμπορικού λογισμικού δεν ήταν διαθέσιμοι. Με βάση την αρχιτεκτονική με την οποία είναι σχεδιασμένοι, μπορούν να χωριστούν σε δύο κατηγορίες:

- Κατηγορία 1

  Σε αυτή την κατηγορία ανήκουν οι αναλυτές οι οποίοι κατασκευάζονται εξ ολοκλήρου χρησιμοποιώντας μια γλώσσα προγραμματισμού, όπως η C++. Αναλυτής της κατηγορίας αυτής είναι ο *Almaden Research Center SGML Parser, (ARC-SGML)* [Gol92], ο οποίος είναι ένας από τους πρώτους που κατασκευάστηκαν. Αποτέλεσε την βάση για την δημιουργία του *SGMLS* [Cla94], ο οποίος με την σειρά του μετεξελίχθηκε στον *SGML Parser, (SP)* [Cla97]. Την στιγμή αυτή ο SP θεωρείται ως ένας από τους καλύτερους, αν όχι ο καλύτερος, αναλυτής. Στην ίδια κατηγορία κατατάσσεται και ο *Yorktown Advanced SGML Parser, (YASP)* [Ric97]. Ο YASP είναι επίσης ένα από τα πιο ανταγωνιστικά προϊόντα. Σχεδόν όλοι οι αναλυτές της κατηγορίας αυτής είναι ανεξάρτητοι αρχιτεκτονικής.

  Η συντακτική ανάλυση επιτελείται, συνήθως, ως εξής: Ο εκάστοτε αναλυτής δέχεται ως είσοδο ένα έγγραφο SGML, αποτελούμενο από το καθορισμό τύπου εγγράφου και μία ή περισσότερες περιπτώσεις εγγράφων. Σε πρώτη φάση συλλέγει όλη την πληροφορία για την δομή των εγγράφων από τον καθορισμό τύπου εγγράφου, πληροφορία την οποία αποθηκεύει σε κατάλληλες δομές. Κατόπιν ξεκινάει η ανάγνωση των περιπτώσεων εγγράφων και η πιστοποίηση της ορθότητας αυτών, χρησιμοποιώντας βέβαια την αποθηκευόμενη πληροφορία. Η όλη διαδικασία επιτελείται σε ένα βήμα (βλέπε Σχήμα 3.1).

- Κατηγορία 2

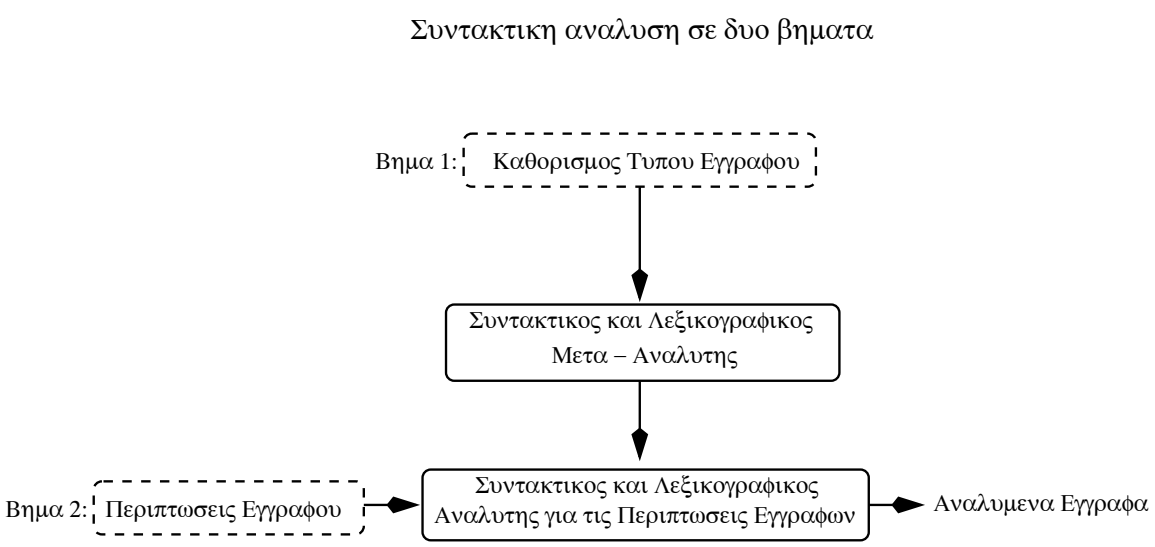


Σχήμα 3.1: Γραφική παράσταση της διαδικασίας συντακτικής ανάλυσης σε ένα βήμα

Σε αυτή την κατηγορία ανήκουν οι αναλυτές οι οποίοι κατασκευάζονται χρησιμοποιώντας κάποιο έτοιμο εργαλείο κατασκευής συντακτικών και λεξικογραφικών αναλυτών, όπως ο *YACC*[Joh75]. Λογισμικό της κατηγορίας αυτής είναι ο *Amsterdam SGML Parser, (ASP-SGML)* [WE89]. Χρησιμοποιεί τον κατασκευαστή συντακτικών αναλυτών *LLGen*[GJ88], προϊόν της ίδιας ομάδας ερευνητών.

Σε αυτή την κατηγορία αναλυτών η συντακτική ανάλυση συνήθως γίνεται ως εξής: Ο αναλυτής δέχεται αρχικά ως είσοδο τον καθορισμό τύπου εγγράφου μόνο. Κατόπιν παράγει ως έξοδο έναν συντακτικό αναλυτή για τις περιπτώσεις του τύπου που δόθηκε ως είσοδος. Οι περιπτώσεις εγγράφων δίδονται ως είσοδος στον παραγόμενο αναλυτή για συντακτική ανάλυση. Η όλη διαδικασία επιτελείται, εδώ, σε δύο βήματα (βλέπε Σχήμα 3.2). Η προσέγγιση αυτή είναι καλύτερη από αυτή που είδαμε στην προηγούμενη κατηγορία, καθώς ο παραγόμενος αναλυτής μπορεί να προσαρμοστεί και να βελτιστοποιηθεί για την πιο αποδοτική συντακτική ανάλυση των περιπτώσεων εγγράφων. Παρόλα αυτά η διαδικασία αυτή είναι πιο χρονοβόρα, και εισάγει μεγάλη καθυστέρηση ειδικά στις περιπτώσεις που οι καθορισμοί τύπου εγγράφου αλλάζουν συχνά, αφού σε κάθε αλλαγή πρέπει πρώτα να παράγεται ο νέος συντακτικός αναλυτής των περιπτώσεων και μετά να αναλυθούν συντακτικά οι ίδιες οι περιπτώσεις.

Στις παραγράφους που ακολουθούν θα περιγραφούν, εν συντομία, μερικά από τα συστήματα τα οποία έχουν προαναφερθεί, η αρχιτεκτονική τους και η λειτουργικότητα που αυτά προσφέρουν.



Σχήμα 3.2: Γραφική παράσταση της διαδικασίας συντακτικής ανάλυσης σε δύο βήματα

3.2.1 SGML Parser, (SP)

Ο SP κατατάσσεται όπως προαναφέρθηκε στην πρώτη κατηγορία αναλυτών. Είναι γραμμένος εξ ολοκλήρου σε C++, γεγονός που τον κάνει μεταφέρσιμο μεταξύ διαφορετικών αρχιτεκτονικών. Είναι μη εμπορικό προϊόν και διατίθεται στο δίκτυο μαζί με τον κώδικα. Υποστηρίζει όλες τις δυνατότητες της SGML, πλην ελαχίστων και, πέρα από το πρόγραμμα της συντακτικής ανάλυσης, το πακέτο λογισμικού περιέχει και μία ομάδα συνοδευτικών προγραμμάτων. Προσφέρει προγραμματιστική διεπιφάνεια χρήσης, που αποτελείται από ένα σύνολο πολύ γενικών ρουτινών τις οποίες ο χρήστης μπορεί να χρησιμοποιήσει ενώ για πιο προχωρημένες λειτουργίες οι ρουτίνες ναι μεν υπάρχουν αλλά πρέπει να τις ανακαλύψει ο ίδιος ο χρήστης καθώς δεν προσφέρεται εγχειρίδιο χρήσης για αυτές. Αν και ο ίδιος ο δημιουργός του SP, James Clark, δεν προσφέρει υποστήριξη για το πρόγραμμα του, μία άλλη εταιρία, με την οποία ο ίδιος δεν συνδέεται, έχει αναλάβει την υποστήριξη. Επειδή ο SP αποδείχθηκε ως ένας από τους καλύτερους αναλυτές, πολλά πακέτα, ακόμα και μεγάλων εταιριών τον χρησιμοποιούν για την συντακτική ανάλυση των εγγράφων SGML και πρόσφατα εταιρία κατασκεύασε και διεπιφάνεια χρήσης του SP σε Java.

3.2.2 Yorktown Advanced SGML Parser, (YASP)

Ο YASP κατατάσσεται επίσης στην πρώτη κατηγορία αναλυτών. Είναι εξ ολοκλήρου γραμμένος σε C, γεγονός που τον κάνει επίσης μεταφέρσιμο μεταξύ αρχιτεκτονικών. Υποστηρίζει αρκετές δυνατότητες της SGML και, ως μη εμπορικό προϊόν, διατίθεται στο δίκτυο μαζί με τον κώδικα. Δεν περιέχει προγραμματιστική διεπιφάνεια χρήσης αλλά άντ' αυτού ο δημιουργός του αναλυτή προσφέρει ένα εγχειρίδιο με όλες τις δομές στις οποίες αποθηκεύεται η πληροφορία μετά την συντακτική ανάλυση. Από κει και πέρα είναι ευθύνη του χρήστη να χρησιμοποιήσει αυτές τις δομές, να εξαγάγει την πληροφορία που αυτός θέλει, και να δημιουργήσει έτσι την δικιά του προγραμματιστική διεπιφάνεια χρήσης.

3.2.3 Amsterdam SGML Parser, (ASP-SGML)

Ο ASP-SGML κατατάσσεται στην δεύτερη κατηγορία αναλυτών. Είναι χτισμένος πάνω στον πρόγραμμα κατασκευής συντακτικών αναλυτών LLGen. Το γεγονός ότι ο LLGen δεν είναι μεταφέρσιμος και υπάρχει μόνο για αρχιτεκτονική UNIX, και VAX/VMS, περιορίζει και την μεταφερσιμότητα του ASP-SGML. Υποστηρίζει αρκετές δυνατότητες της SGML και προσφέρει προγραμματιστική διεπιφάνεια χρήσης χρησιμοποιώντας μηχανισμό ειδοποίησης (notification mechanism) κατά την διάρκεια της συντακτικής ανάλυσης για να αλληλεπιδρά ο συντακτικός αναλυτής με το πρόγραμμα χρήσης του.

Κεφάλαιο 4

Γεννήτρια Μεταφραστών SGML

Στο παρόν κεφάλαιο θα περιγραφεί μια νέα προσέγγιση στην ανάλυση εγγράφων SGML, η οποία αποτελεί και το σκοπό της παρούσης εργασίας. Το κεφάλαιο θα ξεκινήσει με την περιγραφή των κινήτρων, κίνητρα τα οποία αποτέλεσαν το έναυσμα για την δημιουργία αυτής της εργασίας. Ακολούθως θα περιγραφεί η γενική αρχιτεκτονική του συστήματος και κατόπιν θα ακολουθήσει μία λεπτομερής περιγραφή των λειτουργικών μονάδων του.

4.1 Τα Κίνητρα

Εχοντας περιγράψει τα υπάρχοντα συστήματα στο προηγούμενο κεφάλαιο, ένα ερώτημα που λογικά τίθεται είναι γιατί να κατασκευαστεί ένα καινούργιο σύστημα. Ποια η συνεισφορά αυτού στο χώρο των συντακτικών αναλυτών. Επιχειρηματολογώντας υπέρ της παρούσας προσέγγισης θα μπορούσαν να ειπωθούν τα εξής. Εχοντας μελετήσει τις υπάρχουσες προσεγγίσεις, εντοπίστηκαν τόσο τα δυνατά σημεία όσο και οι αδυναμίες αυτών. Αυτές οι αδυναμίες αποτέλεσαν το έναυσμα για την παρούσα προσέγγιση. Τα δυνατά σημεία των υπολοίπων προσεγγίσεων χρησιμοποιήθηκαν ως μέτρο αξιολόγησης και σύγκρισης της παρούσας υλοποίησης με τις υπόλοιπες, ενώ η κάλυψη των αδυναμιών των υπολοίπων συστημάτων είναι αυτή που καθιστά την παρούσα υλοποίηση χρήσιμη. Στις παραγράφους που ακολουθούν θα τεκμηριωθεί ο παραπάνω ισχυρισμός.

4.1.1 Φορμαλισμός στη συντακτική/λεξικογραφική ανάλυση

Μια από τις βασικές επιλογές οι οποίες έγιναν κατά τα αρχικά στάδια σχεδιασμού ήταν ο αναλυτής να χτιστεί πάνω σε ένα υπάρχον εργαλείο κατασκευής αναλυτών, να

ανήκει δηλαδή στην δεύτερη κατηγορία αναλυτών. Η χρήση εργαλείων κατασκευής αναλυτών βοηθάει στην κατασκευή ενός δομημένου αναλυτή, διαχωρίζοντας την συντακτική από τη λεξικογραφική ανάλυση. Λόγω του τρόπου περιγραφής της γραμματικής είναι επίσης πιο εύκολο να αποδειχθεί ότι αυτό που αναγνωρίζει ο αναλυτής είναι η ζητούμενη γλώσσα, κάτι που είναι δύσκολο να αποδειχθεί όταν η συντακτική ανάλυση γίνεται από κάποιο κομμάτι κώδικα.

Το γεγονός ότι η παρούσα προσέγγιση ανήκει στην δεύτερη κατηγορία αναλυτών αποτελεί ένα ακόμη θετικό σημείο, δεδομένου ότι, έτσι επιτυγχάνεται η κατασκευή πιο αποδοτικών αναλυτών, αφού ο παραγόμενος αναλυτής προσαρμόζεται και βελτιστοποιείται για το συγκεκριμένο τύπο εγγράφου. Σε αυτό το σημείο, υπερτερεί η παρούσα προσέγγιση από τους SP και YASP. Επίσης υπερτερεί του ASP-SGML, ο οποίος ανήκει στην ίδια κατηγορία, αφού σε αντίθεση με αυτόν, η παρούσα προσέγγιση είναι ανεξάρτητη αρχιτεκτονικής και αυτό οφείλεται στα εργαλεία κατασκευής αναλυτών τα οποία επιλέχθηκαν, Bison [DS90] και Flex [Pax88].

Τέλος, σημαντική συνεισφορά της παρούσης εργασίας είναι ο διαχωρισμός της συντακτικής από τη λεξικογραφική ανάλυση ενός εγγράφου SGML. Η SGML είναι μία αρκετά πολύπλοκη γλώσσα, στην οποία συχνά μπερδεύονται οι συντακτικές με τις λεξικογραφικές συνιστώσες. Αυτό έχει οδηγήσει αρκετούς [BKW93][COM] στο συμπέρασμα ότι υπάρχοντα εργαλεία, όπως ο Bison και ο Flex, δεν είναι κατάλληλα, και δεν θα έπρεπε να χρησιμοποιηθούν, αφού απαιτούνται κατασκευές στις οποίες οι δύο αυτές συνιστώσες δεν είναι τόσο αυστηρά διαχωρισμένες. Από την άλλη μεριά έχει διαπιστωθεί [NBL93] ότι, παρόλο που υπάρχει αυτή η δυσκολία, θα ήταν πολύ χρήσιμο να χρησιμοποιηθούν τέτοια εργαλεία, έτσι ώστε το πρότυπο να αξιολογηθεί και να μπορέσουν να προταθούν αλλαγές στην γραμματική αυτού. Ενα πρότυπο, όπως αναφέρεται στο ίδιο άρθρο, δεν πρέπει να εισάγει δυσκολίες και επιπλοκές στην κατασκευή λογισμικού το οποίο το υποστηρίζει, και η χρήση ευρέως γνωστών εργαλείων θα βοηθήσει στον εντοπισμό των δυσκολιών αυτών. Με δεδομένα τα παραπάνω, η παρούσα εργασία δεν αποτελεί λοιπόν μία ακόμη υλοποίηση, αλλά μέσω της εμπειρίας η οποία αποκτήθηκε μπορεί να συνεισφέρει στο ευρύτερο τομέα της διαχείρισης ηλεκτρονικών εγγράφων με τη χρήση της SGML.

4.1.2 Ευελιξία σε τροποποιήσεις της γραμματικής

Ακόμη ένα επιχείρημα υπέρ της επιλογής χρησιμοποίησης κατασκευαστή αναλυτών είναι ότι στην περίπτωση αναθεώρησης του προτύπου, αναθεώρηση η οποία μπορεί

να οδηγήσει σε αλλαγή της δομής της γραμματικής αυτού, είναι γενικά πιο εύκολο να αλλαχθεί το αρχείο του κατασκευαστή ώστε να αντικατοπτρίζει τις αλλαγές από το να αλλαχθεί ένα αρχείο κώδικα το οποίο κάνει την συντακτική ανάλυση, επιχείρημα το οποίο αναφέρεται και στο [ASU86]. Με αυτό το τρόπο η προτεινόμενη προσέγγιση είναι πιο ευέλικτη και μπορεί να εξελίσσεται παράλληλα με το πρότυπο. Απόδειξη των παραπάνω είναι και το γεγονός ότι με ελάχιστες αλλαγές ο αναλυτής που κατασκευάστηκε μπορεί να υποστηρίξει το πρότυπο XML [Con97], πρότυπο το οποίο θεωρείται διάδοχος αυτού της SGML.

4.1.3 Ανοικτή αρχιτεκτονική προγραμματιστικής διεπιφάνειας χρήσης

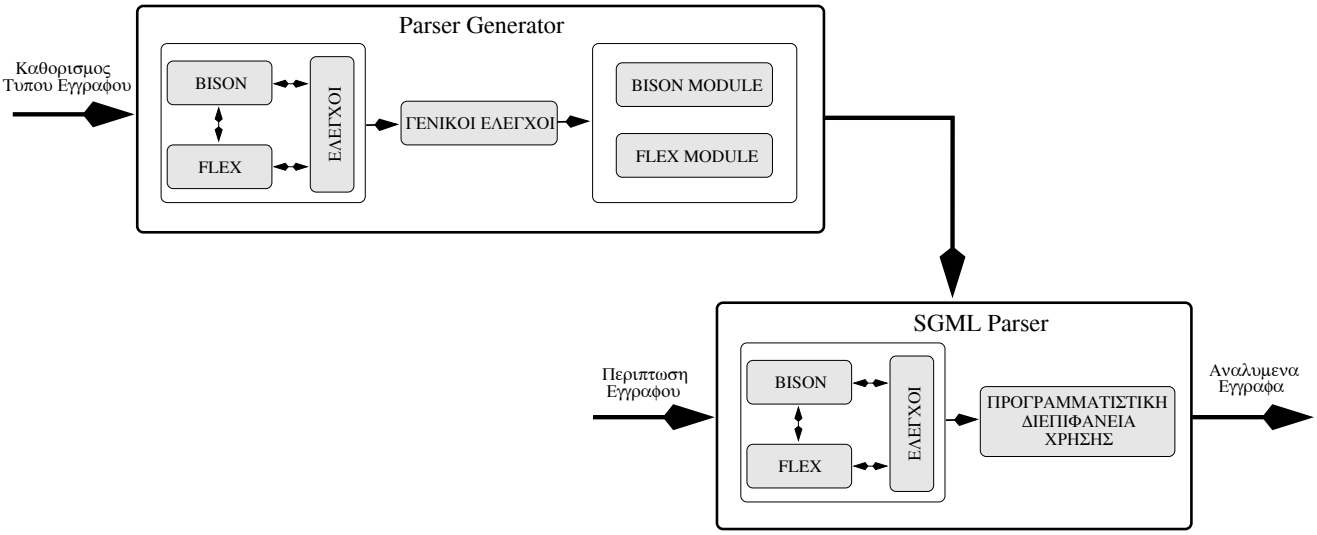
Σημαντικό επίσης χαρακτηριστικό ενός αναλυτή είναι η προγραμματιστική διεπιφάνεια χρήσης και πόσο εύκολα ο χρήστης μπορεί να την χρησιμοποιήσει ή ακόμα και να ορίσει μία δικιά του. Ο SP προσφέρει ένα σύνολο πολύ γενικών συναρτήσεων στον χρήστη ενώ για προχωρημένες λειτουργίες τον αναγκάζει να μελετήσει κομμάτια κώδικα του SP, για να ανακαλύψει αυτό που ζητά. Ο YASP όπως έχει αναφερθεί απλώς περιγράφει τις δομές στις οποίες αποθηκεύει την πληροφορία και αφήνει τα υπόλοιπα στο χρήστη. Τέτοιες προσεγγίσεις, αν και δίνουν τελικά στο χρήστη την απαραίτητη πληροφορία, είναι λανθασμένες αφού οποιαδήποτε αλλαγή είτε στον κώδικα του SP είτε στις δομές του YASP θα ανάγκαζε τον χρήστη να αλλάξει όλες ίσως τις εφαρμογές του οι οποίες βασίζονται σε αυτά τα προγράμματα. Η προγραμματιστική διεπιφάνεια χρήσης θα πρέπει να εισάγει ένα επίπεδο αφαίρεσης τέτοιο ώστε οι αλλαγές να *φιλτράρονται* και να μην φτάνουν ποτέ στο επίπεδο του χρήστη. Αν και αυτή την στιγμή η προγραμματιστική διεπιφάνεια χρήσης του συστήματος βρίσκεται υπό κατασκευή, ο δρόμος που ακολουθείται είναι εντελώς διαφορετικός από αυτούς των προαναφερθέντων προγραμμάτων. Αντικειμενικός σκοπός είναι να μην χρειαστεί ποτέ ο χρήστης να ασχοληθεί με αρχεία κώδικα. Ενα σύνολο συναρτήσεων θα προσφέρει στον χρήστη όλη την πληροφορία που αυτός θέλει είτε για την συγκεκριμένη περίπτωση εγγράφου είτε για τον καθορισμό τύπου εγγράφου του οποίου περίπτωση αποτελεί το έγγραφο. Επίσης θα προσφέρεται η δυνατότητα στον χρήστη να ενσωματώνει δικές του συναρτήσεις στις ενέργειες σημασιολογικού περιεχομένου (semantic actions) του αρχείου Bison του παραγόμενου αναλυτή, ώστε να μπορεί ο χρήστης να ενσωματώσει πλήρως τις συναρτήσεις του σε αυτόν.

4.1.4 Ταχεία παραγωγή πρωτοτύπου

Η ταχεία παραγωγή πρωτοτύπου (rapid prototyping) είναι αναγκαία, ιδιαίτερα κατά τα αρχικά στάδια του κύκλου ζωής ενός συστήματος, αφού έχει αποδειχθεί ότι συμβάλλει στην κατασκευή καλύτερων συστημάτων. Η χρήση εργαλείων κατασκευής συντακτικών και λεξικογραφικών, όπως ο Bison και Flex, βοήθησε στην ταχεία παραγωγή πρωτοτύπου της γεννήτριας μεταφραστών SGML, κάτι που θα ήταν δύσκολο στην περίπτωση που η υλοποίηση γινόταν εξ ολοκλήρου με χρήση μίας γλώσσας προγραμματισμού όπως η C++. Απο την άλλη μεριά, η ακολουθούμενη προσέγγιση θεωρείται οτι οδηγεί γενικά στην κατασκευή πιο αργών αναλυτών κάτι που υποστηρίζεται και στο [ASU86]. Εχοντας όμως ως δεδομένη τη δεκάχρονη και πλέον πορεία των εν λόγω εργαλείων, και τη συνεχή βελτίωση αυτών σε θέματα ταχύτητας, η επιλογή τους τελικά δικαιώνεται απο κάθε άποψη.

4.2 Αρχιτεκτονική Συστήματος

Στις παραγράφους που ακολουθούν θα γίνει μια περιγραφή της αρχιτεκτονική του συστήματος και θα ακολουθήσει μια ανάλυση των λειτουργικών μονάδων (modules) από τις οποίες αποτελείται, τα προβλήματα που υπήρξαν και πώς αυτά αντιμετωπίστηκαν.



Σχήμα 4.1: Γραφική παράσταση της αρχιτεκτονικής της γεννήτριας αναλυτών SGML

Οπως φαίνεται και στο Σχήμα 4.1 το όλο σύστημα αποτελείται από δύο κύρια μέρη.

Το πρώτο μέρος είναι ο μετα-αναλυτής ο οποίος δέχεται ως είσοδο τον καθορισμό τύπου εγγράφου και παράγει ως έξοδο το δεύτερο μέρος της αρχιτεκτονικής, τον αναλυτή για τις περιπτώσεις του εν λόγω τύπου.

Ο μετα-αναλυτής αποτελείται από τρεις κυρίως λειτουργικές μονάδες. Η πρώτη αποτελείται, με την σειρά της, από το ζεύγος εργαλείων Bison και Flex και μία λειτουργική μονάδα η οποία επιτελεί ελέγχους σημασιολογικού περιεχομένου κατά της διάρκεια της συντακτικής και λεξικογραφικής ανάλυσης. Η δεύτερη λειτουργική μονάδα του μετα-αναλυτή είναι υπεύθυνη για την επιτέλεση ενός συνόλου από ελέγχους, έλεγχοι οι οποίοι επιτελούνται μετά την ολοκλήρωση της συντακτικής και λεξικογραφικής ανάλυσης της εισόδου. Η τρίτη και τελευταία λειτουργική μονάδα του μετα-αναλυτή είναι υπεύθυνη για την παραγωγή των αρχείων εξόδου Bison και Flex, αρχεία τα οποία θα χρησιμοποιηθούν για την δημιουργία του αναλυτή για τις περιπτώσεις του καθορισμού τύπου εγγράφου που δόθηκε ως είσοδος.

Οσον αφορά τον παραγόμενο αναλυτή, αυτός αναγνωρίζει *Βασικά Εγγραφα SGML*, όπως αυτά ορίστηκαν στο Κεφάλαιο 2 και αποτελείται από δύο λειτουργικές μονάδες. Η πρώτη είναι παρόμοια με αυτή του μετα-αναλυτή, αφού αποτελείται από το ζεύγος εργαλείων Bison και Flex και μία λειτουργική μονάδα ελέγχων. Η δεύτερη λειτουργική μονάδα του αναλυτή είναι ένα σύνολο από συναρτήσεις οι οποίες αποτελούν την προγραμματιστική διεπιφάνεια χρήσης του αναλυτή.

Ολες οι προαναφερόμενες λειτουργικές μονάδες θα περιγραφούν με λεπτομέρεια στην συνέχεια.

4.3 Μετα-αναλυτής

4.3.1 Bison και Flex του Μετα-αναλυτή

Μια πολύ ηρημένη μορφή της γραμματικής η οποία αναγνωρίζει ένα πρόλογο SGML δόθηκε στο Κεφάλαιο 2 κατά την περιγραφή του προτύπου. Τόσο η συντακτική όσο και η λεξικογραφική ανάλυση βασίστηκε σε ένα προσχέδιο της γραμματικής το οποίο προτάθηκε από τον Goldfarb [GR90]. Δεν αντιμετωπίστηκαν ιδιαίτερα προβλήματα κατά την δημιουργία της γραμματικής για την συντακτική ανάλυση του προλόγου και συγκεκριμένα του καθορισμού τύπου εγγράφου και αυτά που υπήρξαν είχαν να κάνουν κυρίως με τον χειρισμό των διαστημάτων και κενών χαρακτήρων, σημείο στο οποίο η SGML υστερεί έναντι άλλων γλωσσών, αφού οι κενοί χαρακτήρες έχουν σημασία στην SGML και η λεξικογραφική λειτουργική μονάδα του μετα-αναλυτή δεν μπορεί απλώς

να τους αγνοεί. Το πρόβλημα γίνεται ακόμη πιο έντονο στη λεξικογραφική λειτουργική μονάδα του παραγόμενου αναλυτή όπως θα δειχθεί στη συνέχεια.

Για την ορθή παραγωγή μηνυμάτων λάθους, σε περίπτωση συντακτικού λάθους, χρησιμοποιήθηκε ένας μηχανισμός εμπλουτισμού των παραγωγών με σημεία ελέγχου (checkpoints). Ετσι σε περίπτωση συντακτικού λάθους το μήνυμα το οποίο αντιστοιχεί στο τελευταίο σημείο ελέγχου εμφανίζεται στην οθόνη. Προτιμήθηκε αυτή η προσέγγιση από τη χρησιμοποίηση των παραγωγών λάθους που προσφέρει ο Bison και αυτό γιατί οι παραγωγές αυτές δημιουργούν συχνά αμφισημίες στην γραμματική παράγοντας shift/reduce και reduce/reduce conflicts, κάτι που δεν είναι επιθυμητό. Οσον αφορά τα σημασιολογικά λάθη, η λειτουργική μονάδα ελέγχου, η οποία θα περιγραφεί στην συνέχεια, δημιουργήθηκε για αυτόν ακριβώς το σκοπό, για τον εντοπισμό και την αναφορά σημασιολογικών λαθών.

4.3.2 Λειτουργική μονάδα Γενικών Ελέγχων

Οταν η συντακτική και λεξικογραφική ανάλυση τελειώσει τότε αν δεν έχει προκύψει κάποιο συντακτικό λάθος, η λειτουργική μονάδα ελέγχου καλείται για να επιτελέσει ένα σύνολο από ελέγχους απαραίτητους για την πιστοποίηση της εισόδου. Ο καθορισμός τύπου εγγράφου που δόθηκε ως είσοδος ελέγχεται για:

- Πολλαπλές δηλώσεις του ιδίου στοιχείου
- Πολλαπλές δηλώσεις του ιδίου χαρακτηριστικού για ένα συγκεκριμένο στοιχείο
- Πολλαπλές δηλώσεις της ιδίας σημείωσης
- Πολλαπλές δηλώσεις της ιδίας οντότητας. Στην περίπτωση αυτή ο χρήστης ειδοποιείται ότι, σύμφωνα με το πρότυπο, όταν μία οντότητα δηλωθεί πολλές φορές η πρώτη δήλωση είναι αυτή που ισχύει.
- Ελέγχονται οι δηλώσεις γνωρισμάτων για να διαπιστωθεί αν υπάρχει δήλωση χαρακτηριστικών για ένα στοιχείο το οποίο δεν έχει δηλωθεί.
- Τέλος ελέγχονται τα μοντέλα περιεχομένων των δηλωθέντων στοιχείων για ύπαρξη αμφισημίας, όπως αυτή ορίζεται στο πρότυπο.

Αυτός ο τελευταίος έλεγχος είναι και ο πιο σημαντικός και στην συνέχεια θα περιγραφεί με λεπτομέρεια, αφού πρώτα δοθεί μια γενική εικόνα αυτού. Συγκεκριμένα για να επιτελεστεί ο έλεγχος αμφισημίας του μοντέλου ενός οποιουδήποτε στοιχείου, σαν πρώτο βήμα

το μοντέλο περιεχομένων του δίδεται ως είσοδος σε μία λειτουργική μονάδα η οποία αναλαμβάνει να μετατρέψει το μοντέλο σε ένα ισοδύναμο αλλά απλούστερο μοντέλο. Αυτή η μετατροπή δεν εισάγει ούτε αναιρεί αμφισημίες του αρχικού μοντέλου, ισχυρισμός ο οποίος θα αποδειχθεί στη συνέχεια. Το απλούστερο μοντέλο θα χρησιμοποιηθεί από την λειτουργική μονάδα κατασκευής αυτομάτων, η οποία θα κατασκευάσει το αυτόματο του μοντέλου αυτού και βάση του αυτομάτου αυτού θα αποφασιστεί η ύπαρξη ή όχι αμφισημίας στο αρχικό μοντέλο. Εφόσον διαπιστωθεί ότι δεν υπάρχουν αμφισημίες, το απλουστευμένο μοντέλο θα χρησιμοποιηθεί από την λειτουργική μονάδα παραγωγής του αρχείου εξόδου Bison, για την παραγωγή μιας βελτιστοποιημένης γραμματικής, γεγονός το οποίο θα οδηγήσει στην γρηγορότερη και αποδοτικότερη συντακτική ανάλυση των περιπτώσεων εγγράφων.

4.3.3 Λειτουργική Μονάδα Απλοποίησης Μοντέλου

Ο δημιουργός ενός καθορισμού τύπου εγγράφου έχει ως ευθύνη του, να *μεταφράσει* την δομή που θα ήθελε να έχουν τα κείμενα του σε ένα σύνολο από δηλώσεις στοιχείων, με τα αντίστοιχα μοντέλα περιεχομένων, δηλώσεις σύμφωνες πάντοτε με το πρότυπο της SGML. Οπως φάνηκε κατά την περιγραφή του προτύπου στο κεφάλαιο 2, το μοντέλο περιεχομένων ενός στοιχείου είναι μία παράσταση η οποία είναι στην ουσία κανονικές εκφράσεις (regular expressions), αν εξαιρέσουμε βέβαια τις εξαιρέσεις και τον τελεστή ampersand (&). Το γεγονός ότι ο δημιουργός του τύπου δεν είναι υποχρεωτικά γνώστης της θεωρίας γλωσσών, οδηγεί στο συμπέρασμα ότι οι εκφράσεις οι οποίες δίνει δεν είναι πάντοτε οι καλύτερες δυνατές. Είναι λοιπόν επιθυμητό να ευρεθεί η βέλτιστη δυνατή, ισοδύναμη πάντα παράσταση, και μάλιστα χωρίς αυτή η μετατροπή να εισάγει ή να αφαιρεί αμφισημίες από το αρχικό μοντέλο. Το βέλτιστο μοντέλο δεν είναι απλώς πιο εύκολο στην διαχείριση, επεξεργασία και μελέτη του, αλλά επιπλέον προσφέρει την δυνατότητα παραγωγής ενός μικρότερου αναλυτή για τον συγκεκριμένο καθορισμό τύπου εγγράφου.

Πως όμως θα επιτευχθεί αυτή η βελτιστοποίηση; Για να επιτευχθεί το ζητούμενο, θα χρησιμοποιηθούν ένα σύνολο από κανόνες βελτιστοποίησης μοντέλων. Οι κανόνες αυτοί προτάθηκαν αρχικά από τον H. H. Rath και αργότερα ο Arjan Loeffen πρότεινε μία επέκτασή τους [Loe94]. Μετά της απαρίθμηση τους θα ακολουθήσει η απόδειξη της εγκυρότητας μερικών από αυτούς, κυρίως αυτών που δεν είναι ίσως τόσο προφανής.

Πίνακας 4.1 *:*

(A)	→	*A*
(A)?	→	*A?*
*(A)**	→	*A**
(A)+	→	*A+*
(A?)	→	*A?*
(A)*	→	*A**
(A+)	→	*A+*

Πίνακας 4.2 *:*

(A)?*	→	*A**
(A)**	→	*A**
(A)+*	→	*A**
(A?)?	→	*A?*
*(A?)**	→	*A**
(A?)+	→	*A**
(A+)?	→	*A**
*(A+)**	→	*A**
(A+)+	→	*A+*

Πίνακας 4.3 *:*

(A, (B, C))	→	*(A, B, C)*
(A | (B | C))	→	*(A | B | C)*

Πίνακας 4.4 *:*

(A? | B)?	→	*(A | B)?*
(A? | B)+	→	*(A | B)**
(A+ | B)+	→	*(A | B)+*
*(A+ | B)**	→	*(A | B)**
(A | B)+*	→	*(A | B)**
*(A? | B)**	→	*(A | B)**
(A | B)**	→	*(A | B)**

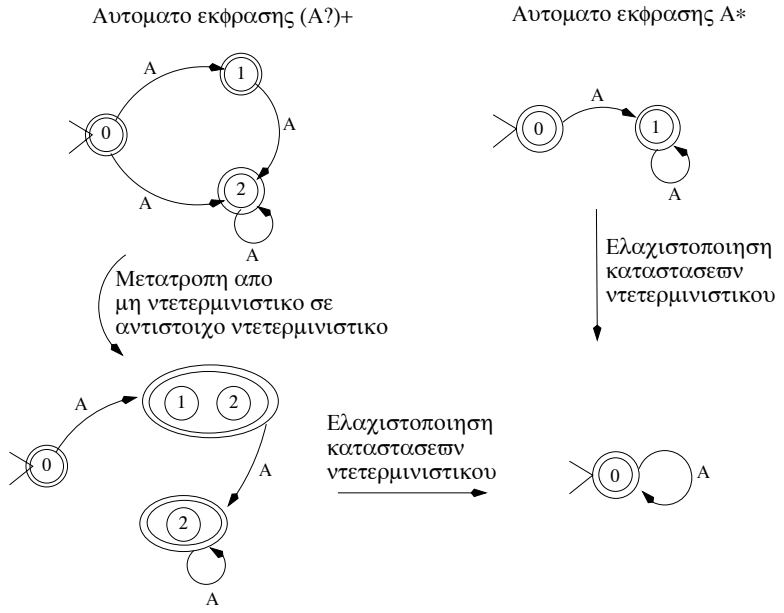
Ενα πολύ σημαντικό χαρακτηριστικό των παραπάνω κανόνων είναι, όπως έχει αναφερθεί, ότι δεν εισάγουν και δεν αναιρούν αμφισημίες του μοντέλου περιεχομένων. Ο όρος αμφισημία εδώ χρησιμοποιείται ως συνώνυμο του όρου αμφισημία ως προς σύμβολα όπως αυτή ορίστηκε στο κεφάλαιο 2. Από την άλλη μεριά οι κανόνες αυτοί αναιρούν τις αμφισημίες ως προς τελεστές. Αυτό είναι επιθυμητό καθώς στην κατασκευή του αυτομάτου η οποία θα επακολουθήσει, και στην εξέταση αυτών ως προς τον ντετερμινισμό τους, πιθανή ύπαρξη μη ντετερμινισμού θα πρέπει να οφείλεται μόνο σε αμφισημία ως προς σύμβολα.

Στον πίνακα 4.1 υπάρχει ένα σύνολο κανόνων οι οποίοι κυρίως αποσκοπούν στην αφαίρεση των επιπλέον παρενθέσεων από το μοντέλο περιεχομένων. Στον πίνακα 4.2 οι κανόνες οι οποίοι υπάρχουν αποσκοπούν στην συνένωση δύο τελεστών εμφάνισης σε ένα, εξαλείφοντας συγχρόνως τις παρενθέσεις. Με τους κανόνες του πίνακα αυτού εξαλείφεται και η ύπαρξη αμφισημίας βάσει τελεστή. Η εγκυρότητα των κανόνων αυτών δεν είναι ίσως προφανής, για αυτό θα αποδειχθεί για μερικούς από αυτούς. Για την επίτευξη της απόδειξης θα χρησιμοποιηθούν τα αυτόματα τα οποία εκφράζουν το αριστερό και το δεξί μέρος κάθε κανόνα. Τα αυτόματα αυτά παράγονται χρησιμοποιώντας τον αλγόριθμο του Thompson[ASU86]. Από το Thompson αυτόματο το οποίο παράγεται αφαιρούνται οι κενές μεταβάσεις, που ο αλγόριθμος αυτός εισάγει, και παράγεται ένα νέο ισοδύναμο αυτόματο το οποίο ονομάζεται Glushkov[Glu61] αυτόματο. Η όλη διαδικασία θα περιγραφεί αναλυτικότερα στην επόμενη παράγραφο που αναφέρεται στην

λειτουργική μονάδα κατασκευής αυτομάτων. Εδώ απλώς γίνεται χρήση των αποτελεσμάτων της λειτουργικής αυτής μονάδας. Επίσης για την επίτευξη της απόδειξης θα γίνει χρήση του αλγορίθμου μετατροπής ενός μη ντετερμινιστικού αυτομάτου στο αντίστοιχο ντετερμινιστικό καθώς και του αλγορίθμου ελαχιστοποίησης των καταστάσεων ενός ντετερμινιστικού αυτομάτου.

Απόδειξη 4.1 *:*

Το σκεπτικό της απόδειξης είναι το εξής: Με δεδομένα τα δύο αρχικά αυτόματα θα αποδειχθεί ότι με μια σειρά μετασχηματισμών και τα δύο αυτόματα καταλήγουν στο ίδιο κοινό αυτόματο αποδεικνύοντας έτσι την ισοδυναμία των αρχικών αυτομάτων και κατ'επέκταση των εκφράσεων στις οποίες αυτά αντιστοιχούν. Θα αποδειχθεί αρχικά η εγκυρότητα του κανόνα $(A?)+ \rightarrow A*$. Η απόδειξη απεικονίζεται και στο σχήμα 4.2.



Σχήμα 4.2: Γραφική παράσταση της απόδειξης της ισοδυναμίας των δύο αυτομάτων που αντιστοιχούν στις εκφράσεις $(A?)+$ και $A*$

Το πρώτο βήμα της απόδειξης είναι από τις δύο εκφράσεις να βρεθούν τα αντίστοιχα αυτόματα. Αυτό επιτυγχάνεται, όπως έχει προαναφερθεί, εφαρμόζοντας αρχικά τον αλγόριθμο του Thompson και παίρνοντας στη συνέχεια το ισοδύναμο αυτόματο του Glushkov. Τα αυτόματα που παράγονται φαίνονται στην πρώτη σειρά του σχήματος. Το πρώτο αυτόματο το οποίο αντιστοιχεί στην κανονική έκφραση $(A?)+$ είναι μη ντετερμινιστικό, αφού από την κατάσταση μηδέν υπάρχουν δύο μεταβάσεις με το σύμβολο A. Αν

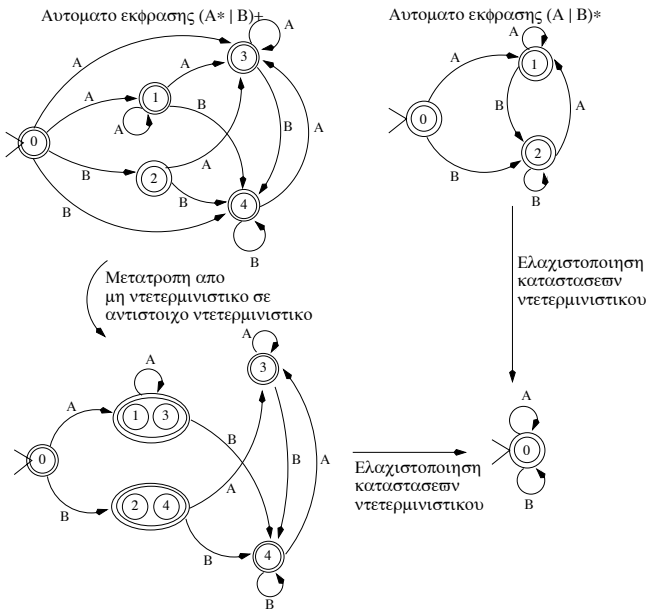
συμβολιστεί με e η κενή λέξη και δοθεί η λέξη A ως είσοδος τότε, με βάση την έκφραση $(A?)_+$, αυτή μπορεί να παραχθεί με επιλογή ενός μόνο A και οσεσδήποτε κενές λέξεις πριν ή μετά από αυτό, παραδείγματος χάριν, e e A e ή A e e e. Αρα ο μη ντετερμινισμός του αυτομάτου οφείλεται στην ύπαρξη αμφισημίας λόγω τελεστών, άρα μπορεί να αναιρεθεί αφού δεν αποτελεί αντικείμενο πιστοποίησης, σύμφωνα με το πρότυπο. Ο μη ντετερμινισμός αναιρείται με τη μετατροπή του αυτομάτου στο αντίστοιχο ντετερμινιστικό. Κατόπιν εφαρμόζεται ο αλγόριθμος ελαχιστοποίησης καταστάσεων στο ντετερμινιστικό αυτόματο. Ο ίδιος αλγόριθμος εφαρμόζεται στο αυτόματο της έκφρασης $A*$. Με αυτό το τρόπο και τα δύο αρχικά αυτόματα καταλήγουν στο ίδιο κοινό αυτόματο, άρα η απόδειξη ολοκληρώθηκε. Οι υπόλοιποι κανόνες του πίνακα αποδεικνύονται με παρόμοιο τρόπο. ■

Στον πίνακα 4.3 οι κανόνες οι οποίοι παρατίθενται είναι αρκετά απλοί. Σύμφωνα με αυτούς τους κανόνες ένα υπομοντέλο περιεχομένων μπορεί να ενωθεί με το μοντέλο στο οποίο περιέχεται, με τη προϋπόθεση ότι το υπομοντέλο δεν έχει τελεστή εμφάνισης, εάν έχουν τον ίδιο τελεστή σύνδεσης. Η ιδιότητα αυτή δεν ισχύει για τον τελεστή σύνδεσης ampersand γιατί για αυτόν δεν ισχύει η προσεταιριστική ιδιότητα. Στον πίνακα 4.4 οι κανόνες οι οποίοι παρατίθενται είναι λιγότερο προφανείς. Η χρήση αυτών των κανόνων αποσκοπεί στο να αφαιρέσει, αν είναι δυνατό, τους τελεστές εμφάνισης των στοιχείων ενός μοντέλου περιεχομένων, και να αφήσει μόνο τον τελεστή εμφάνισης του ιδίου του μοντέλου. Στη συνέχεια θα αποδειχθεί η εγκυρότητα του κανόνα $(A* \mid B)_+ \rightarrow (A \mid B)*$.

Απόδειξη 4.2 *:*

Η απόδειξη ακολουθεί το ίδιο σκεπτικό με την προηγούμενη. Τα αρχικά αυτόματα φαίνονται στην πρώτη σειρά του σχήματος 4.3. Οι μετασχηματισμοί οδηγούν και τα δύο αρχικά αυτόματα στο ίδιο τελικό και η απόδειξη έχει ολοκληρωθεί. Οι υπόλοιποι κανόνες του πίνακα 4.4 αποδεικνύονται με παρόμοιο τρόπο. ■

Η μεθοδολογία η οποία ακολουθείται για την εφαρμογή των κανόνων, δοθέντος ενός μοντέλου περιεχομένων, είναι η εξής: Το μοντέλο περιεχομένων, όντας μία κανονική έκφραση, μπορεί να παρασταθεί από ένα συντακτικό δέντρο (syntax tree). Στο δέντρο αυτό γίνεται μία κατά βάθος διάσχιση. Κατά την διάσχιση αυτή σε κάθε κόμβο ελέγχεται η δυνατότητα εφαρμογής κάποιου από τους προτεινόμενους κανόνες. Οταν η διάσχιση του δένδρου ολοκληρωθεί ελέγχεται αν έχει επιτελεστεί κάποια αλλαγή στο δένδρο. Αν αυτό συμβαίνει μία νέα κατά βάθος διάσχιση αρχίζει, αλλιώς η διαδικασία βελτιστοποίησης τερματίζεται. Ο λόγος για τον οποίο, σε περίπτωση αλλαγής του δένδρου, η διαδικασία



Σχήμα 4.3: Γραφική παράσταση της απόδειξης της ισοδυναμίας των δύο αυτομάτων που αντιστοιχούν στις εκφράσεις (A∗ | B)+ και (A | B)∗

επαναλαμβάνεται είναι ότι μπορεί κάποια αλλαγή στην τελευταία διάσχιση να επιτρέπει την εφαρμογή ενός κανόνα σε κάποιο σημείο του δέντρου, αλλαγή η οποία δεν ήταν δυνατό να επιτελεστεί πριν, αλλά οι τελευταίες αλλαγές που έγιναν την καθιστούν τώρα δυνατή. Αυτό μπορεί να φανεί και από το ακόλουθο παράδειγμα.

Παράδειγμα 4.1 *:*

Μοντέλο περιεχομένων: $(((A*)? \mid ((((B \mid C)+)+)*) \mid (((D))?))*)$

Πρώτη διάσχιση: $(A* \mid (B \mid C)* \mid D?)*$
Δεύτερη διάσχιση: $(A \mid (B \mid C) \mid D)*$
Τρίτη διάσχιση: $(A \mid B \mid C \mid D)*$

Κατά την πρώτη διάσχιση αφαιρούνται οι περιττές παρενθέσεις και συνενώνονται τελεστές εμφάνισης. Κατά την δεύτερη διάσχιση καθίσταται δυνατή η εφαρμογή κανόνων του πίνακα 4.4 απαλείφοντας έτσι τελεστές οι οποίοι εμφανίζονται στα στοιχεία και υπομοντέλα του μοντέλου και κρατώντας μόνο τον τελεστή εμφάνισης όλου του μοντέλου. Εφόσον τώρα στο εσωτερικό του μοντέλου το υπομοντέλο $(B \mid C)$ δεν έχει τελεστή εμφάνισης και επιπλέον έχει τον ίδιο τελεστή σύνδεσης με το μοντέλο στο οποίο περιέχεται, μπορεί να γίνει χρήση των κανόνων του πίνακα 4.3 και συγκεκριμένα του

δεύτερου κανόνα. Η τέταρτη διάσχιση δεν αλλάζει την έκφραση οπότε η διαδικασία τερματίζεται.

Τελειώνοντας την περιγραφή αυτής της λειτουργικής μονάδας μένει να διευκρινιστεί το εξής. Θα μπορούσε κάποιος να αναρωτηθεί γιατί να γίνουν αυτές οι απλοποιήσεις στο μοντέλο περιεχομένων, μέσω των προτεινόμενων κανόνων, και μετά να κατασκευαστεί το αυτόματο για τον έλεγχο της αμφισημίας και να μην γίνει κατευθείαν η κατασκευή του αυτομάτου, με βάση το αρχικό μοντέλο, και η ελαχιστοποίηση αυτού με τις μεθόδους που χρησιμοποιήθηκαν στις αποδείξεις των κανόνων. Το αποτέλεσμα όπως φάνηκε και από τις αποδείξεις είναι το ίδιο. Η απάντηση στην παραπάνω ερώτηση είναι η εξής. Οπως αναφέρθηκε και στη αρχή της παραγράφου, κύριο χαρακτηριστικό των κανόνων είναι η διατήρηση της αμφισημίας. Από την άλλη μεριά μέθοδοι όπως αυτή της μετατροπής μη ντετερμινιστικού αυτομάτου στο αντίστοιχο ντετερμινιστικό αναιρούν πάντα πιθανές αμφισημίες οπότε ο παραπέρα έλεγχος για ύπαρξη αμφισημίας καθίσταται περιττός. Αντικειμενικός σκοπός της παρούσας προσέγγισης είναι πάντοτε η ελαχιστοποίηση της αρχικής έκφρασης, διατηρώντας όμως τις ιδιότητες της, όσον αφορά την αμφισημία.

4.3.4 Λειτουργική μονάδα κατασκευής αυτομάτου

Το αυτόματο, όπως έχει επανειλημμένα αναφερθεί, χρησιμοποιείται μεταξύ άλλων, για να πιστοποιηθεί αν ένα μοντέλο περιεχομένων παρουσιάζει αμφισημία ως προς σύμβολα. Οπως είναι γνωστό από την θεωρία αυτομάτων[ASU86] [HU79][Pap92], για οποιαδήποτε κανονική έκφραση, είναι δυνατό να κατασκευαστεί ένα αυτόματο το οποίο αναγνωρίζει την ίδια γλώσσα με αυτή που ορίζει η έκφραση, και επιπλέον το αυτόματο αυτό να έχει και την επιθυμητή ιδιότητα του ντετερμινισμού. Χρειάζεται λοιπόν ιδιαίτερη προσοχή κατά την κατασκευή του αυτομάτου ώστε να μην κατασκευάζεται πάντοτε ένα ντετερμινιστικό αυτόματο, αλλά να κατασκευάζεται ένα αυτόματο, ντετερμινιστικό ή μή, διατηρώντας έτσι την ύπαρξη ή μη της αμφισημίας του μοντέλου.

Ο αλγόριθμος του Thompson είναι ένας από τους πιο διαδεδομένους αλγορίθμους κατασκευής αυτομάτων από μια κανονική έκφραση ο οποίος παράγει ένα μη ντετερμινιστικό αυτόματο με κενές μεταβάσεις. Παρόμοιοι αλγόριθμοι περιγράφονται στα [HU79] [AO83][SSS88] και τα αυτόματα που δημιουργούνται είναι ισόμορφα με αυτό του Thompson. Με δεδομένο το Thompson αυτόματο μιας έκφρασης πρέπει σαν επόμενο βήμα να αφαιρεθούν οι κενές μεταβάσεις. Ενα από τα κύρια χαρακτηριστικά ενός Thompson αυτομάτου είναι ότι καμία κατάσταση αυτού δεν είναι αποτελεί συγχρόνως προορισμό

κενών και μη κενών μεταβάσεων, αλλά οι μεταβάσεις είναι μόνο του ενός ή του άλλου είδους. Μία κατάσταση η οποία δεν έχει εισερχόμενες κενές μεταβάσεις θα ονομάζεται Glushkov κατάσταση, ενώ μία κατάσταση η οποία έχει εισερχόμενες κενές μεταβάσεις θα ονομάζεται μη-Glushkov κατάσταση. Αν όλες οι μη-Glushkov καταστάσεις απαλειφθούν τότε το Thompson μετατρέπεται σε ένα ισοδύναμο αυτόματο το οποίο ονομάζεται Glushkov αυτόματο και το οποίο όπως αποδεικνύεται στα [BK94a][BKW93], έχει την επιθυμητή ιδιότητα της διατήρησης των πιθανών αμφισημιών της κανονικής έκφρασης. Η αφαίρεση των μη-Glushkov καταστάσεων επιτυγχάνεται με την ακόλουθη μέθοδο.

Εστω μία μη-Glushkov κατάσταση K, και έστω $(K_1, e, K), \ldots, (K_m, e, K)$ οι εισερχόμενες και $(K, x_1, L_1), \ldots, (K, x_n, L_n)$ οι εξερχόμενες μεταβάσεις της K. Η κατάσταση K μπορεί να απαλειφθεί μαζί με όλες τις μεταβάσεις της, και στο νέο αυτόματο εισάγονται οι μεταβάσεις (K_i, x_j, L_j), για $1 \leq i \leq m$και$1 \leq j \leq n$. Επιπλέον αν η απαλειφθείσα κατάσταση K είναι τελική κατάσταση τότε οι καταστάσεις $\mathrm{K}_1, \ldots, K_m$ γίνονται τελικές. Επαναλαμβάνοντας την παραπάνω διαδικασία για όλες τις μη-Glushkov καταστάσεις, παράγεται ένα αυτόματο το οποίο είναι το Glushkov αυτόματο. Το αυτόματο αυτό ελέγχεται για να διαπιστωθεί αν είναι μη ντετερμινιστικό. Αν είναι, τότε το αρχικό μοντέλο περιεχομένων περιέχει αμφισημία και ο αναλυτής αναφέρει στο χρήστη το γεγονός.

4.3.5 Λειτουργική μονάδα παραγωγής αρχείου Bison

Εχοντας ολοκληρώσει τους ελέγχους της εισόδου για σημασιολογικά λάθη και αμφισημίες σειρά τώρα έχει η λειτουργική μονάδα η οποία είναι υπεύθυνη για την παραγωγή του αρχείου Bison, αρχείο το οποίο περιέχει την γραμματική βάσει της οποίας γίνεται η συντακτική ανάλυση των περιπτώσεων εγγράφου. Για την κατασκευή της γραμματικής χρησιμοποιείται η πληροφορία η οποία συγκεντρώθηκε από το μεταναλυτή και αφορά τις δηλώσεις στοιχείων και τις δηλώσεις γνωρισμάτων. Επίσης χρησιμοποιείται το απλουστευμένο μοντέλο του μοντέλου περιεχομένων των στοιχείων και ο λόγος είναι ότι το απλουστευμένο μοντέλο, όπως έχει προαναφερθεί, προσφέρει την δυνατότητα παραγωγής μικρότερης γραμματικής, πιο απλής, επιταχύνοντας έτσι την συντακτική ανάλυση των περιπτώσεων εγγράφων. Για την καλύτερη κατανόηση της διαδικασίας αυτόματης παραγωγής του Bison αρχείου θα χρησιμοποιηθεί ένα απόσπασμα από ένα καθορισμό τύπου εγγράφου το οποίο περιέχει μία δήλωση στοιχείου και μία δήλωση γνωρίσματος και θα παρουσιαστεί το σύνολο των παραγωγών οι οποίες δημιουργούνται με αυτόματο τρόπο.

Παράδειγμα 4.2 *:*

<!ELEMENT a - - (b, c?, d)+ >
<!ATTLIST a status (final | draft) draft>

Στο παράδειγμα, το μοντέλο περιεχομένων του στοιχείου a αποτελείται από τα στοιχεία b, c, d από τα οποία το στοιχείο c είναι προαιρετικό. Ολο το μοντέλο περιεχομένων έχει τελεστή εμφάνισης το συν. Επίσης για το στοιχείο a ορίζεται ένα γνώρισμα, το status, με δύο πιθανές τιμές final ή draft και με την τιμή draft να αποτελεί την εξ ορισμού τιμή του γνωρίσματος. Γενικά για οποιοδήποτε στοιχείο η διαδικασία κατασκευής των παραγωγών χωρίζεται σε τρία στάδια. Πρώτα κατασκευάζεται η παραγωγή που αφορά την συντακτική ανάλυση του στοιχείου, μετά οι παραγωγές για την συντακτική ανάλυση των γνωρισμάτων του και τέλος οι παραγωγές για την συντακτική ανάλυση του μοντέλου περιεχομένων του.

4.3.5.1 Κατασκευή παραγωγών στοιχείου

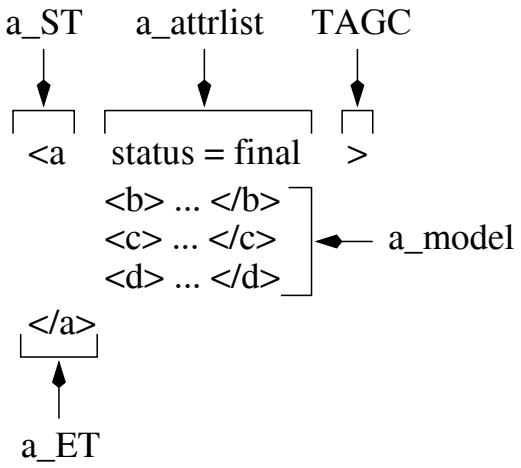
Η δομή της γραμματικής η οποία παράγεται για ένα οποιοδήποτε στοιχείο είναι γενικά της μορφής

Παραγωγή 4.1 *:*

<όνομα στοιχείου> → *<όνομα στοιχείου>_ST,*
<όνομα στοιχείου>_attrlist,
TAGC,
<όνομα στοιχείου>_model
<όνομα στοιχείου>_ET

Τα *<όνομα στοιχείου>_ST*, *<όνομα στοιχείου>_ET* και *TAGC* είναι τερματικά σύμβολα ενώ τα σύμβολα *<όνομα στοιχείου>_model*, *<όνομα στοιχείου>_attrlist* είναι μη τερματικά. Το τι ακριβώς περιέχουν τα μη τερματικά αυτά σύμβολα και σε ποιες παραγωγές αντιστοιχούν θα μελετηθεί στη συνέχεια. Το τι αντιπροσωπεύει κάθε ένα από αυτά τα σύμβολα φαίνεται στο Σχήμα 4.4, όπου παρίσταται με γραφικό τρόπο η συντακτική ανάλυση του στοιχείου a, για υποθετική περίπτωση εγγράφου, και γίνεται μία αντιστοίχιση ανάμεσα στα σύμβολα της γραμματικής και στα συντακτικά μέρη της περίπτωσης εγγράφου.

Από το σχήμα γίνεται αντιληπτό ότι η συντακτική ανάλυση της ετικέτας αρχής χωρίζεται σε τρία μέρη. Αυτό οφείλεται στο γεγονός ότι στο εσωτερικό μίας ετικέτας αρχής εκχωρούνται οι τιμές των γνωρισμάτων ενός στοιχείου. Οπότε μετά την αρχική



Σχήμα 4.4: Γραφική παράσταση της συντακτικής ανάλυσης του στοιχείου a

αναγνώριση του τερματικού συμβόλου *<όνομα στοιχείου>_ST*, επακολουθεί η συντακτική ανάλυση μίας ή περισσοτέρων αναθέσεων τιμών σε γνωρίσματα του στοιχείου, και η λίστα των αναθέσεων τερματίζεται με την συνάντηση του συμβόλου *TAGC*, σύμβολο το οποίο σηματοδοτεί και το τέλος της ετικέτας αρχής. Η συντακτική ανάλυση του στοιχείου συνεχίζεται με το μοντέλο του, και ολοκληρώνεται με τη συνάντηση της ετικέτας τέλους. Η συντακτική ανάλυση αυτής είναι αρκετά απλή, πρόκειται για ένα τερματικό σύμβολο, καθώς δεν μπορούν να υπάρχουν αναθέσεις τιμών σε γνωρίσματα, σε μια ετικέτα τέλους. Σύμφωνα με τα όσα περιγράφηκαν, η συγκεκριμένη γραμματική η οποία παράγεται για το στοιχείο a του παραδείγματος είναι

Παράδειγμα 4.3 *:*

a → *a_ST*
a_attrlist
TAGC
a_model
a_ET
;

4.3.5.2 Κατασκευή παραγωγών γνωρισμάτων

Η δομή της γραμματικής η οποία παράγεται για την συντακτική ανάλυση του μη τερματικού *<ονομα στοιχείου>_attrlist* είναι στην γενική της περίπτωση η ακόλουθη

Παραγωγή 4.2 :

<όνομα στοιχείου>_attrlist → <όνομα στοιχείου>_attrs | / empty */*

Παραγωγή 4.3 :

<όνομα στοιχείου>_attrs → <όνομα στοιχείου>_attrs, <ονομα στοιχείου>_attr
| <όνομα στοιχείου>_attr

Παραγωγή 4.4 :

<όνομα στοιχείου>_attr → <όνομα στοιχείου>_<ονομα γνωρίσματος>
| <όνομα στοιχείου>_<ονομα γνωρίσματος>
...
| <όνομα στοιχείου>_<ονομα γνωρίσματος>

Παραγωγή 4.5 :

<όνομα στοιχείου>_<ονομα γνωρίσματος> → <όνομα γνωρίσματος> = <τιμή>

Η παραγωγή 4.2 δείχνει ότι η λίστα αναθέσεων τιμών γνωρισμάτων, εκτός από την περίπτωση στην οποία υπάρχουν γνωρίσματα με τύπο εμφάνισης γνωρίσματος REQUIRED, μπορεί να είναι άδεια. Σε αυτή την περίπτωση τα γνωρίσματα τα οποία έχουν δηλωθεί στον καθορισμό τύπου εγγράφου για το συγκεκριμένο στοιχείο παίρνουν τις εξ ορισμού τιμές τους. Η παραγωγή 4.3 σε συνδυασμό με την παραγωγή 4.4 δείχνει ότι η σειρά με την οποία γίνονται οι δηλώσεις ανάθεσης τιμών μπορεί να είναι οποιαδήποτε. Τα γνωρίσματα δεν έχουν κάποια σχετική σειρά μεταξύ τους, την οποία πρέπει να ακολουθεί ο χρήστης. Εδώ πρέπει να σημειωθεί ότι η παραγωγή 4.3 δεν δημιουργείται όταν ένα στοιχείο έχει μόνο ένα γνώρισμα. Στο παράδειγμα που ακολουθεί παρουσιάζονται οι παραγωγές οι οποίες δημιουργούνται αυτόματα για τα γνωρίσματα του στοιχείου a.

Παράδειγμα 4.4 :

```
a_attrlist → a_attr | /* empty */
             ;
a_attr →     a_status
             ;
a_status →     STATUS = FINAL
             | STATUS = DRAFT
             ;
```

4.3.5.3 Κατασκευή παραγωγών μοντέλου περιεχομένων

Από την παραγωγή 4.1 το μόνο σύμβολο το οποίο απόμεινε να εξεταστεί είναι το μη τερματικό σύμβολο *<όνομα στοιχείου>_model*. Η δομή της γραμματικής για τη συντακτική ανάλυση του μοντέλου περιεχομένων ενός στοιχείου δεν είναι τόσο τυποποιημένη, όσο τα προηγούμενα παραδείγματα, αφού αυτή εξαρτάται από τα περιεχόμενα του μοντέλου, τα πιθανά υπομοντέλα και στοιχεία από τα οποία αυτό αποτελείται, καθώς επίσης και από τους τελεστής σύνδεσης και εμφάνισης αυτών. Οπως περιγράφηκε και στο κεφάλαιο 2, υπάρχουν τρεις τελεστές σύνδεσης. Από αυτούς, ο τελεστής ampersand, δεν υποστηρίζεται από την προτεινόμενη προσέγγιση οπότε απομένει η εξέταση των δύο υπολοίπων τελεστών, του τελεστή κόμμα και του τελεστή κάθετη μπάρα. Οι δύο όμως αυτοί τελεστές είναι ίδιοι με τους τελεστές σύνδεσης των συμβόλων στην γραμματική BNF. Αρα ο τελεστής σύνδεσης του μοντέλου περιεχομένων καθορίζει κατά μοναδικό τρόπο τον τελεστή της παραγωγής BNF η οποία κατασκευάζεται για την συντακτική ανάλυση του μοντέλου. Η κατάσταση διαφέρει όσον αφορά τους τελεστές εμφάνισης, αφού δεν υπάρχουν τελεστές εμφάνισης στην γραμματική BNF. Ετσι οι τελεστές εμφάνισης των μοντέλων μεταφράζονται σε μία ή περισσότερες παραγωγές. Συγκεκριμένα

Παραγωγή 4.6 *:*

<όνομα στοιχείου>_Optmodel → *<όνομα στοιχείου>_model*
| */* empty */*

Παραγωγή 4.7 *:*

<όνομα στοιχείου>_Plusmodel → *<όνομα στοιχείου>_Plusmodel, <ονομα στοιχείου>_model*
| *<όνομα στοιχείου>_model*

Παραγωγή 4.8 *:*

<όνομα στοιχείου>_Repmodel → *<όνομα στοιχείου>_Plusmodel*
| */* empty */*

Η παραγωγή 4.6 δημιουργείται για κάθε μοντέλο περιεχομένων, υπομοντέλο περιεχομένων ή στοιχείο, το οποίο έχει σαν τελεστή εμφάνισης το ερωτηματικό. Αντίστοιχα η παραγωγή 4.7 δημιουργείται κάθε φορά που ο τελεστής εμφάνισης είναι το σύμβολο συν. Τέλος αν ένα μοντέλο έχει σαν τελεστή εμφάνισης τον αστερίσκο, τότε δημιουργούνται δύο παραγωγές, οι 4.7 και 4.8. Στο παράδειγμα που ακολουθεί παρουσιάζεται η γραμματική η οποία παράγεται για το μοντέλο περιεχομένων του στοιχείου a, κάνοντας χρήση των παραπάνω γενικών παραγωγών. Κατ'αρχήν θα χρησιμοποιηθεί η Παραγωγή

4.7. Αυτό γιατί το μοντέλο περιεχομένων του στοιχείου a έχει τελεστή εμφάνισης το συν. Με αυτό τον τρόπο παράγονται οι δύο πρώτες παραγωγές του παραδείγματος. Στην τρίτη παραγωγή υπάρχει το καθεαυτό μοντέλο περιεχομένων με την παράθεση των στοιχείων. Επειδή όμως το στοιχείο c είναι προαιρετικό δεν παρατίθεται το ίδιο αλλά, με χρήση της Παραγωγής 4.6 που χρησιμοποιείται για προαιρετικά στοιχεία, παράγεται η τέταρτη παραγωγή του παραδείγματος.

Παράδειγμα 4.5 *:*

a_model → *a_Plusmodel1*
;

a_Plusmodel1 → *a_Plusmodel1, a_model1*
| *a_model1*
;

a_model1 → *b, a_Optmodel2, d*
;

a_Optmodel2 → *c*
| */* empty */*
;

Τελειώνοντας την περιγραφή της παρούσης λειτουργικής μονάδας πρέπει να σημειωθούν τα εξής. Πέρα από τη αυτοματοποίηση της παραγωγής της γραμματικής, ιδιαίτερη προσοχή δόθηκε στο να μην παρουσιάζει η παραγόμενη γραμματική αμφισημίες, όχι ως προς το πρότυπο, αλλά ως προς το εργαλείο Bison. Η πιθανή ύπαρξη κάποιων Bison αμφισημιών, εννοώντας τα shift/reduce και reduce/reduce conflicts, θα καταστούσε την όλη διαδικασία της αυτοματοποίησης άχρηστη καθώς ο χρήστης θα ήταν αναγκασμένος να επέμβει στην παραγόμενη γραμματική. Το προαναφερθέν πρόβλημα λύθηκε με τη χρήση καταλλήλων κανόνων ονοματοδοσίας των παραγομένων μή τερματικών συμβόλων και με τον περιορισμό του πλήθους των κενών παραγωγών.

4.3.6 Λειτουργική μονάδα παραγωγής αρχείου Flex

Αυτή η λειτουργική μονάδα είναι υπεύθυνη για την παραγωγή του αρχείου Flex, το οποίο περιέχει τις δηλώσεις για την λεξικογραφική ανάλυση των περιπτώσεων εγγράφων. Δηλώσεις Flex παράγονται για την αναγνώριση των ετικετών αρχής και τέλους των διαφόρων στοιχείων για τα ονόματα των γνωρισμάτων καθώς επίσης και για τις τιμές που αυτά μπορούν να πάρουν, όταν οι τιμές αυτές ορίζονται στον καθορισμό τύπου εγγράφου αυστηρά, όπως συμβαίνει για το γνώρισμα status που παίρνει μόνο τις τιμές final και draft.

Στο παραγόμενο αρχείο Flex αποθηκεύεται πληροφορία για τις οντότητες και το κείμενο στο οποίο αυτές αντιστοιχούν, και αυτό γιατί, όπως έχει ήδη αναφερθεί, είναι ευθύνη του λεξικογραφικού αναλυτή όταν συναντήσει μία οντότητα να την αντικαταστήσει με το κείμενο το οποίο της αντιστοιχεί. Επίσης υπάρχουν όλες οι απαραίτητες δηλώσεις Flex για την διαχείριση δηλώσεων σχολίων, μαρκαρισμένων ενοτήτων, οδηγιών επεξεργασίας. Ενα από τα βασικότερα προβλήματα το οποίο αντιμετωπίστηκε κατά την παρούσα εργασία, και το οποίο επηρέασε άμεσα την κατασκευή της παρούσας λειτουργικής μονάδας, ήταν η δυσκολία λεξικογραφικής ανάλυσης των περιπτώσεων εγγράφων. Στις παραγράφους που ακολουθούν θα αναφερθούν τα πιο σημαντικά προβλήματα, τα οποία ανέκυψαν κατά την λεξικογραφική ανάλυση των περιπτώσεων εγγράφων. Επίσης θα γίνει μια σύντομη περιγραφή του τρόπου με τον οποίο αυτά τα προβλήματα αντιμετωπίστηκαν.

4.3.6.1 Χαρακτήρες Διαχωρισμού

Οπως αναφέρθηκε και στο Κεφάλαιο 2, κατά την περιγραφή του SGML προλόγου, ένα σύνολο από χαρακτήρες ορίζονται στην SGML και αποτελούν τους λεγόμενους χαρακτήρες διαχωρισμού. Οι χαρακτήρες αυτοί, κάτω από ορισμένες συνθήκες, αποκτούν ιδιαίτερη σημασία, επηρεάζοντας την λεξικογραφική ανάλυση, ενώ κάτω από άλλες συνθήκες συμπεριφέρονται σαν κανονικοί χαρακτήρες κειμένου. Χαρακτηριστικό παράδειγμα είναι ο χαρακτήρας & ο οποίος χρησιμοποιείται στις περιπτώσεις εγγράφων, για τις αναφορές σε οντότητες. Σύμφωνα με το πρότυπο, για να γίνει αναφορά σε μία οντότητα πρέπει να υπάρχει ο χαρακτήρας & ακολουθούμενος από το όνομα της οντότητας, χωρίς μεσολάβηση κενών διαστημάτων. Το παράδειγμα που ακολουθεί θα κάνει τα παραπάνω πιο εμφανή.

Παράδειγμα 4.6 *:*

(1)<!ENTITY CSD "Computer Science Department">
(2)<!ENTITY UOC "University of Crete">

(3) The &CSD in the & UOC is one of the best in the world

Στο παράδειγμα υπάρχει ένα απόσπασμα από ένα καθορισμό τύπου εγγράφου (γραμμές 1 και 2), και ένα απόσπασμα από μία περίπτωση εγγράφου (γραμμή 3). Στον καθορισμό τύπου εγγράφου ορίζονται δύο οντότητες με ονόματα *CSD* και *UOC* έτσι ώστε

ο δημιουργός μίας περίπτωσης εγγράφου να μπορεί, αντί να επαναλαμβάνει συνέχεια τις αντίστοιχες εκφράσεις *Computer Science Department* και *University of Crete*, να χρησιμοποιεί τις αναφορές γλιτώνοντας έτσι χρόνο κατά την πληκτρολόγηση, τακτική η οποία χρησιμοποιείται συχνά για τις εκφράσεις οι οποίες επαναλαμβάνονται συχνά σε ένα κείμενο. Είναι ευθύνη του λεξικογραφικού αναλυτή, όπως προαναφέρθηκε, να αντικαταστήσει τις αναφορές με το κείμενο στο οποίο αυτές αναφέρονται. Επανερχόμενοι στο παράδειγμα και σύμφωνα με τα όσα έχουν λεχθεί, στην γραμμή 3 υπάρχει **μόνο** μία αναφορά σε οντότητα, η αναφορά στην οντότητα *CSD*, και ο λεξικογραφικός αναλυτής θα πρέπει να αντικαταστήσει την αναφορά με το κείμενο στο οποίο αυτή αντιστοιχεί. Από την άλλη πλευρά, λόγω της μεσολάβησης κενού διαστήματος μεταξύ του & και του *UOC*, δεν θεωρείται ότι υπάρχει αναφορά στην οντότητα *UOC* στην γραμμή αυτή. Η διαδικασία της λεξικογραφικής ανάλυσης περιπλέκεται ακόμη περισσότερο καθώς η ερμηνεία η οποία δόθηκε προηγουμένως για την λεξικογραφική ανάλυση της γραμμής 3 είναι σωστή μόνο αν η γραμμή 3 εμφανίζεται στα περιεχόμενα ενός στοιχείου, το μοντέλο περιεχομένων του οποίου έχει δηλωθεί στο καθορισμό τύπου εγγράφου ως PCDATA ή RCDATA. Στην περίπτωση που η γραμμή 3 εμφανίζεται στα περιεχόμενα στοιχείου με μοντέλο περιεχομένων CDATA ακόμα και η ακολουθία *&CSD* δεν θεωρείται ως αναφορά σε οντότητα αλλά ως απλό κείμενο και καμία αντικατάσταση δεν λαμβάνει χώρα.

Στο ίδιο σκεπτικό, το σύμβολο $<$ ακολουθούμενο από ένα πεζό ή κεφαλαίο χαρακτήρα θεωρείται ως το σύμβολο εκκίνησης μίας ετικέτας αρχής ενώ ακολουθούμενο από οποιοδήποτε άλλο χαρακτήρα δεν έχει καμία ειδική ερμηνεία. Η εξαίρεση στον παραπάνω κανόνα είναι όταν καθόλα νόμιμες συντακτικά ετικέτες πρέπει να θεωρούνται ως κείμενο όταν εμφανίζονται μέσα σε στοιχεία με μοντέλο περιεχομένων CDATA ή RCDATA και αυτό γιατί η μόνη ετικέτα η οποία αναγνωρίζεται σε αυτές τις καταστάσεις είναι η ετικέτα τέλους είτε του παρόντος στοιχείου είτε οποιουδήποτε στοιχείου μέσα στο οποίο το παρόν στοιχείο είναι ενθυλακωμένο.

Ενα ακόμη χαρακτηριστικό παράδειγμα *διπλής συμπεριφοράς* χαρακτήρα, είναι αυτό των κενών χαρακτήρων όπως ο χαρακτήρας διαστήματος και ο χαρακτήρας αλλαγής γραμμής. Το ακόλουθο παράδειγμα θα βοηθήσει στην εξήγηση.

Παράδειγμα 4.7 *:*

$< stanza > \hookleftarrow$

$< line > \hookleftarrow$

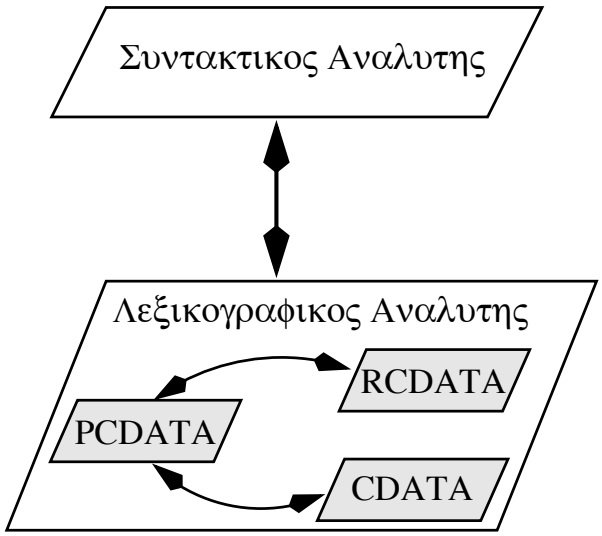
$< /line >$

$\ldots$

Στην πρώτη γραμμή του παραδείγματος ο χαρακτήρας αλλαγής γραμμής πρέπει να *καταναλωθεί* από το λεξικογραφικό αναλυτή χωρίς να επιστραφεί τιμή στον συντακτικό λόγω του μοντέλου περιεχομένων του στοιχείου $stanza$, το οποίο στο Παράδειγμα 4.8 ορίστηκε ως (line+,comment?). Από την άλλη μεριά ο χαρακτήρας αλλαγής γραμμής στην δεύτερη γραμμή δεν πρέπει απλώς να *καταναλωθεί* από το λεξικογραφικό αναλυτή, αλλά πρέπει να *επιστραφεί* η τιμή PCDATA στον συντακτικό αναλυτή αφού το μοντέλο του στοιχείου $line$ ορίστηκε ως PCDATA και σύμφωνα με το πρότυπο ως PCDATA θεωρούνται κανένας ή περισσότεροι χαρακτήρες, ακόμα και κενοί.

Για να αντιμετωπιστούν όλα αυτά τα προβλήματα, δημιουργήθηκαν ξεχωριστές καταστάσεις λεξικογραφικής ανάλυσης, μία για κάθε είδος αλφαριθμητικών περιεχομένων, CDATA, RCDATA, PCDATA. Οι καταστάσεις αυτές μπορούν να θεωρηθούν σαν αυτόνομοι λεξικογραφικοί αναλυτές, όπου ο καθένας αναγνωρίζει διαφορετικά πράγματα, προσφέροντας επίσης την δυνατότητα περάσματος από τον ένα στον άλλο, χωρίς να επηρεάζει αυτό την λειτουργία του κοινού συντακτικού αναλυτή. Συγκεκριμένα οι μεταβάσεις οι οποίες επιτρέπονται είναι μόνο από PCDATA σε RCDATA και αντίστροφα και από PCDATA σε CDATA και αντίστροφα. Το σε ποια κατάσταση βρίσκεται ο λεξικογραφικός αναλυτής εξαρτάται από το μοντέλο περιεχομένων του κάθε στοιχείου. Επίσης, συχνά για το ίδιο κανόνα, ο λεξικογραφικός αναλυτής χρειάστηκε να επιτελεί ελέγχους για το ποια τιμή θα επιστρέψει στον συντακτικό αναλυτή. Τα παραπάνω απεικονίζονται και στο Σχήμα 4.5.

Η μετάβαση από κατάσταση PCDATA, η οποία είναι και η εξ ορισμού, σε κατάσταση RCDATA ή CDATA γίνεται μέσω της αναγνώρισης ετικέτας αρχής ενός στοιχείου του οποίο το μοντέλο περιεχομένων έχει δηλωθεί ως RCDATA ή CDATA. Η επιστροφή σε κατάσταση PCDATA γίνεται μέσω της αναγνώρισης μίας έγκυρης ετικέτας τέλους. Οπως προαναφέρθηκε έγκυρη ετικέτα είναι η ετικέτα τέλους είτε του παρόντος στοιχείου είτε οποιουδήποτε στοιχείου μέσα στο οποίο το παρόν στοιχείο είναι ενθυλακωμένο. Το πρόβλημα αναγνώρισης μίας έγκυρης ετικέτας τέλους λύθηκε με τη χρήση στοίβας, στην οποία καταγράφονται ετικέτες στοιχείων. Οι ετικέτες αυτές ανήκουν σε στοιχεία για τα οποία έχει *συναντηθεί* η ετικέτα αρχής τους αλλά όχι η ετικέτα τέλους, και άρα δεν έχει ολοκληρωθεί ακόμη η συντακτική ανάλυση των περιεχομένων τους. Ετσι αναγνωρίζονται τα στοιχεία μέσα στα οποία το παρόν στοιχείο είναι ενθυλακωμένο.



Σχήμα 4.5: Γραφική παράσταση της αλληλεπίδρασης μεταξύ συντακτικού και λεξικογραφικού αναλυτή και παράσταση των καταστάσεων τελευταίου

4.3.6.2 Παράλειψη ετικετών

Ο παραγόμενος αναλυτής ήταν επιθυμητό να προσφέρει τις δυνατότητες OMITTAG και SHORTTAG όπως αυτές ορίστηκαν στο Κεφάλαιο 2. Για να υποστηριχθούν οι δυνατότητες αυτές έγιναν τα εξής. Ο συντακτικός αναλυτής κατασκευάστηκε έτσι ώστε στις παραγωγές του όλες οι ετικέτες να υπάρχουν υποχρεωτικά. Αυτό γιατί η ύπαρξη παραγωγών με προαιρετικές ετικέτες δημιουργούσε Bison αμφισημίες, δηλαδή shift/reduce και reduce/reduce conflicts. Από την άλλη μεριά κατασκευάστηκε μηχανισμός ο οποίος έδινε στον λεξικογραφικό αναλυτή την δυνατότητα να αντιλαμβάνεται, σε πρώτη φάση, πότε μία ετικέτα έχει παραλειφθεί, και σε δεύτερη φάση, να την εισάγει στην στοίβα του Flex έτσι ώστε ο ίδιος ο λεξικογραφικός αναλυτής να μπορεί να την αναγνωρίσει στην συνέχεια, και να την επιστρέψει στο συντακτικό αναλυτή. Ο κατασκευασθείς μηχανισμός χρησιμοποιεί την στοίβα του Flex, την στοίβα η οποία κατασκευάστηκε για την καταγραφή των ετικετών και περιγράφηκε προηγουμένως, καθώς και πληροφορία από τα αυτόματα που κατασκευάστηκαν από τον μετα-αναλυτή, για να επιτύχει το σκοπό του.

Ενα πρόβλημα άμεσα συνδεδεμένο με το προηγούμενο ήταν ο χειρισμός του χαρακτήρα τερματισμού αρχείου, EOF. Συγκεκριμένα, το πρόβλημα είναι ότι συχνά το αρχείο το οποίο περιέχει την περίπτωση εγγράφου τερματίζεται και ένας αριθμός από ετικέτες οι οποίες έχουν παραλειφθεί θα έπρεπε να συμπληρωθούν, πριν ο λεξικογραφικός

αναλυτής ενημερώσει τον συντακτικό ότι η είσοδος έχει τελειώσει. Οι ετικέτες αυτές συμπληρώνονται από τον μηχανισμό ο οποίος περιγράφηκε προηγουμένως, η συντακτική ανάλυση ολοκληρώνεται επιτυχώς και τότε ο λεξικογραφικός αναλυτής ενημερώνει τον συντακτικό ότι έχει ολοκληρωθεί η είσοδος.

4.4 Αναλυτής

4.4.1 Προγραμματιστική διεπιφάνεια χρήσης αναλυτή

Οπως αναφέρθηκε και στην αρχή του κεφαλαίου, ένα από τα θετικά σημεία της παρούσας υλοποίησης είναι η δυνατότητα να μπορεί ο εκάστοτε χρήστης να ενσωματώνει τις δικές του συναρτήσεις στις ενέργειες σημασιολογικού περιεχομένου του αναλυτή. Η παρούσα εργασία αναπτύχθηκε μέσα στα πλαίσια του προγράμματος AQUARELLE [INR] και για τις ανάγκες του προγράμματος αυτού, η συγκεκριμένη διεπιφάνεια χρήσης η οποία δημιουργήθηκε αποσκοπεί στην φόρτωση της δομής και των περιεχομένων της περίπτωσης εγγράφου σε μία βάση δεδομένων βασισμένη στο ιεραρχικό μοντέλο δεδομένων. Περισσότερες λεπτομέρειες για την διεπιφάνεια αυτή θα δοθούν στο επόμενο κεφάλαιο. Εδώ θα μελετηθεί η γενική διεπιφάνεια χρήσης την οποία είναι επιθυμητό να προσφέρει ο αναλυτής ανεξαρτήτως εφαρμογής η οποία τον χρησιμοποιεί. Η υπό σχεδιασμό διεπιφάνεια χρήσης είναι επιθυμητό να προσφέρει δύο είδη πληροφοριών, πληροφορίες για τον καθορισμό τύπου εγγράφου, του οποίου το παρόν έγγραφο αποτελεί περίπτωση, και πληροφορίες για την υπό εξέταση περίπτωση εγγράφου. Στις παραγράφους που ακολουθούν θα δοθεί μία σύντομη περιγραφή των απαραιτήτων διαδικασιών.

4.4.1.1 Υπηρεσίες καθορισμού τύπου εγγράφου

- Γενικές Υπηρεσίες

  -- SessionOpen, SessionClose
  Οι συναρτήσεις αυτές χρησιμοποιούνται για την εκκίνηση/τερματισμό ενός καναλιού επικοινωνίας μεταξύ μίας εφαρμογής και του αναλυτή.

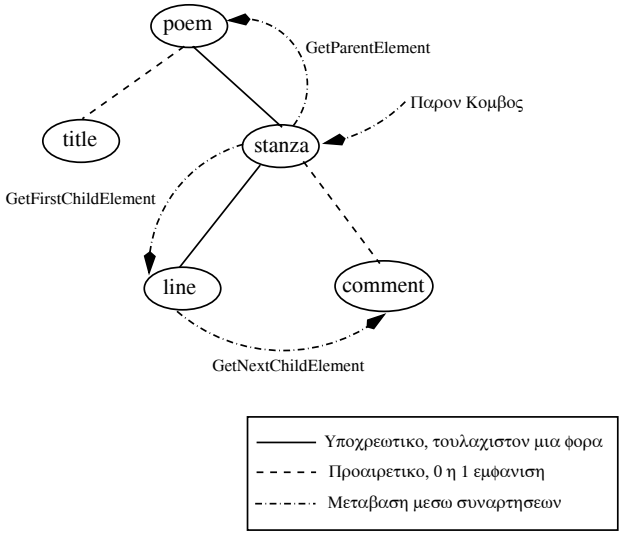
- Υπηρεσίες Πλοήγησης και Συλλογής Πληροφοριών

  -- GetRootElement, GetParentElement, GetFirstChildElement, GetNextChildElement, GetPrevChildElement

Οι συναρτήσεις αυτές χρησιμοποιούνται για την πλοήγηση μέσα στο καθορισμό τύπου εγγράφου. Η δομή ενός τύπου, όπως αναφέρθηκε και στο Κεφάλαιο 2, μπορεί να παρασταθεί από ένα δένδρο του οποίου οι κόμβοι είναι τα στοιχεία που ορίζονται στον τύπο. Κάθε κόμβος πατρικός αυτού του δένδρου έχει ως θυγατρικούς του εκείνους τους κόμβους των οποίων τα στοιχεία εμφανίζονται στο μοντέλο περιεχομένων του πατρικού κόμβου. Ετσι για τον ακόλουθο καθορισμό τύπου εγγράφου το δέντρο το οποίο παριστάνει της δομή του φαίνεται στο Σχήμα 4.6.

Παράδειγμα 4.8 *:*

(1)<!-- This is another DTD for poems -->
(2)<!DOCTYPE poem [
(3)<!ELEMENT poem - - (title?, stanza+) >
(4)<!ATTLIST poem status (draft | final) draft>
(5)<!ELEMENT title - - CDATA>
(6)<!ELEMENT stanza - - (line+,comment?)>
(7)<!ELEMENT line - - (#PCDATA)>
(8)<!ELEMENT comment - - (#PCDATA) >
(9)]>



Σχήμα 4.6: Γραφική παράσταση της δομής του καθορισμού τύπου εγγράφου για ένα ποίημα

Οι συναρτήσεις αυτές χρησιμοποιούνται για τη διάσχιση του εν λόγω δένδρου. Κατά της διάρκεια της διάσχισης αυτής είναι δυνατή η συλλογή πληροφοριών για τον κόμβο που τώρα επισκεπτόμαστε. Αν υποτεθεί ότι ο κόμβος υπό επίσκεψη είναι ο *stanza* τότε μία κλήση της *GetParentElement* θα οδηγήσει στον κόμβο *poem* ενώ μία κλήση της *GetFirstChildElement* θα οδηγήσει στον κόμβο *line*. Αν την κλήση της *GetFirstChildElement* ακολουθήσει μία κλήση της *GetNextChildElement* τότε ο υπό επίσκεψη κόμβος θα είναι αυτός του στοιχείου *comment*. Οι προτεινόμενες συναρτήσεις δεν ορίζουν μία συγκεκριμένη διάσχιση του εν λόγω δένδρου, αλλά επιτρέπουν στο χρήστη να διασχίσει το δένδρο κατά την δική του βούληση.

-- GetElementName, GetElementOccurence, GetElementConnector
Οι συναρτήσεις αυτές χρησιμοποιούνται για την συλλογή πληροφοριών για το στοιχείο, στον κόμβο του οποίου τώρα επισκεπτόμαστε.

-- GetAttributeCount, GetFirstAttribute, GetNextAttribute, GetAttributeName, GetAttributeDefaultValue, GetAttributeCount, GetAttributeList
Οι συναρτήσεις αυτές χρησιμοποιούνται για την συλλογή πληροφοριών, πληροφορίες οι οποίες αφορούν τον αριθμό και το είδος των χαρακτηριστικών που έχει το στοιχείο στον κόμβο του οποίου βρισκόμαστε.

-- GetFirstEntity, GetNextEntity, FindFileEntity, ExpandEntity, IsEntityDefined
Ο συναρτήσεις αυτές μας δίνουν πληροφορίες για τις οντότητες οι οποίες ορίζονται στο καθορισμό τύπου εγγράφου.

4.4.1.2 Υπηρεσίες περίπτωσης εγγράφου

- Γενικές Υπηρεσίες

-- SessionOpen, SessionClose
Οι συναρτήσεις αυτές χρησιμοποιούνται για την εκκίνηση/τερματισμό ενός καναλιού επικοινωνίας μεταξύ μίας εφαρμογής και του αναλυτή.

-- SetInstancePath, SetCatalogPath, SetErrorFile, FindInstance, LoadInstance, CloseInstance, SaveInstance
Οι συναρτήσεις αυτές χρησιμοποιούνται για το καθορισμό του μονοπατιού στο οποίο βρίσκονται οι περιπτώσεις SGML εγγράφων. Χρησιμοποιούνται επίσης για την ανεύρεση, φόρτωση και αποθήκευση των εγγράφων.

-- ValidateInstance

Χρησιμοποιείται για τον έλεγχο της ορθότητας της περίπτωσης εγγράφου, σε σχέση πάντοτε με το πρότυπο, καθώς και σε σχέση με το καθορισμό τύπου εγγράφου.

- Υπηρεσίες Πλοήγησης και Συλλογής Πληροφοριών

-- GetParentElement, GetFirstChildElement, GetNextChildElement, GetPrevChildElement, GetFirstElement, GetLastElement, GetNextElement, GetPrevElement, FindElement

Οι συναρτήσεις αυτές χρησιμοποιούνται για την πλοήγηση μέσα στο έγγραφο. Η δομή ενός έγγραφου μπορεί να παρασταθεί από ένα δένδρο του οποίου οι κόμβοι είναι τα στοιχεία του εγγράφου. Στο δένδρο κάθε πατρικός κόμβος έχει ως θυγατρικούς του εκείνους τους κόμβους των οποίων τα στοιχεία εμφανίζονται στο περιεχόμενο του πατρικού κόμβου. Οι συναρτήσεις αυτές χρησιμοποιούνται για τη διάσχιση του εν λόγω δένδρου. Κατά της διάρκεια της διάσχισης αυτής είναι δυνατή η συλλογή πληροφοριών για τον κόμβο που τώρα επισκεπτόμαστε. Οι προτεινόμενες συναρτήσεις μπορούν να χρησιμοποιηθούν όπως ακριβώς οι συναρτήσεις οι οποίες προσφέρονται για την διάσχιση του δένδρου της δομής του καθορισμού τύπου εγγράφου, με μόνη διαφορά ότι εδώ το δένδρο διάσχισης, όπως έχει προαναφερθεί, παριστάνει τη δομή της περίπτωσης εγγράφου.

-- GetElementContents, GetElementName, GetElementType

Οι συναρτήσεις αυτές χρησιμοποιούνται για την συλλογή πληροφοριών για το στοιχείο του οποίου το κόμβο τώρα επισκεπτόμαστε.

-- GetAttributeCount, GetFirstAttribute, GetNextAttribute, GetAttributeType, GetAttributeName, GetAttributeDefaultValue, GetAttributeCount, GetAttributeList

Οι συναρτήσεις αυτές χρησιμοποιούνται για την συλλογή πληροφοριών, πληροφορίες οι οποίες αφορούν τον αριθμό και το είδος των χαρακτηριστικών που έχει το στοιχείο στον κόμβο του οποίου βρισκόμαστε.

-- GetFirstEntity, GetNextEntity, FindFileEntity, ExpandEntity, IsEntityDefined

Ο συναρτήσεις αυτές μας δίνουν πληροφορίες για τις οντότητες που εμφανίζονται στη παρούσα περίπτωση εγγράφου.

Κεφάλαιο 5

Μία εφαρμογή AQUARELLE

Η παρούσα υλοποίηση χρησιμοποιείται στο πρόγραμμα AQUARELLE για την συντακτική και λεξικογραφική ανάλυση SGML εγγράφων καθώς και για την φόρτωση τόσο της δομής όσο και των περιεχομένων των περιπτώσεων εγγράφων στο Σύστημα Σημασιολογικού Ευρετηριασμού (ΣΣΕ) [Ins97][CD94], προϊόν του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Ερευνας. Το ΣΣΕ είναι βασισμένο στην γλώσσα TELOS[MBJK90]. Στη συνέχεια θα γίνει μία σύντομη περιγραφή του ερευνητικού προγράμματος AQUARELLE και θα ακολουθήσει μία εισαγωγή τόσο στο ΣΣΕ όσο και στην γλώσσα TELOS και τέλος θα παρουσιαστεί η μεθοδολογία η οποία ακολουθείται για την φόρτωση των περιπτώσεων εγγράφων στο ΣΣΕ.

5.1 AQUARELLE

Αντικειμενικός σκοπός του ερευνητικού προγράμματος AQUARELLE είναι ο διαμοιρασμός πολιτισμικής πληροφορίας. Το πρόγραμμα απευθύνεται στους ερευνητές πολιτισμικής πληροφορίας, σε μουσεία, εκδοτικούς οίκους και πολιτισμικούς οργανισμούς. Προσφέρει στους χρήστες την δυνατότητα πρόσβασης στο σύνολο των πληροφοριών που αφορούν την ευρωπαϊκή πολιτισμική κληρονομιά όπως μνημεία, μουσικά όργανα, γλυπτά, πίνακες κτλ. Οι πληροφορίες οργανώνονται σε θεματικές ενότητες και βρίσκονται αποθηκευμένες σε ένα σύνολο απο ετερογενείς πηγές δεδομένων, με την εκάστοτε πηγή να βρίσκεται, πιθανώς, σε διαφορετική γεωγραφική περιοχή. Οι χρήστες μπορούν να θέτουν ερωτήσεις σε μία ή περισσότερες, απο τις διαθέσιμες, πηγές. Η διατύπωση των επερωτήσεων διευκολύνεται με τη χρήση πολύγλωσσων θυσαυρών όρων.

Ακολούθως θα αναφερθούν τα κυρίως μέρη της αρχιτεκτονικής του συστήματος

και θα περιγραφούν μερικές απο τις βασικές επιλογές, οι οποίες έγιναν κατά την διάρκεια του σχεδιασμού της αρχιτεκτονικής, και οι οποίες συνδέονται άμεσα με την παρούσα εργασία. Οι βασικές λειτουργικές μονάδες του προγράμματος AQUARELLE είναι τέσσερις.

- Οι εξυπηρέτες αρχείου (archive servers), στους οποίους φυλλάσσεται η διαμοιραζόμενη πολιτισμική πληροφορία. Ο εξυπηρέτης αρχείου είναι στην ουσία η πηγή των δεδομένων και, όπως προαναφέρθηκε, υπάρχουν πολλές ετερογενείς πηγές δεδομένων, άρα πολλοί ετερογενείς εξυπηρέτες αρχείου.

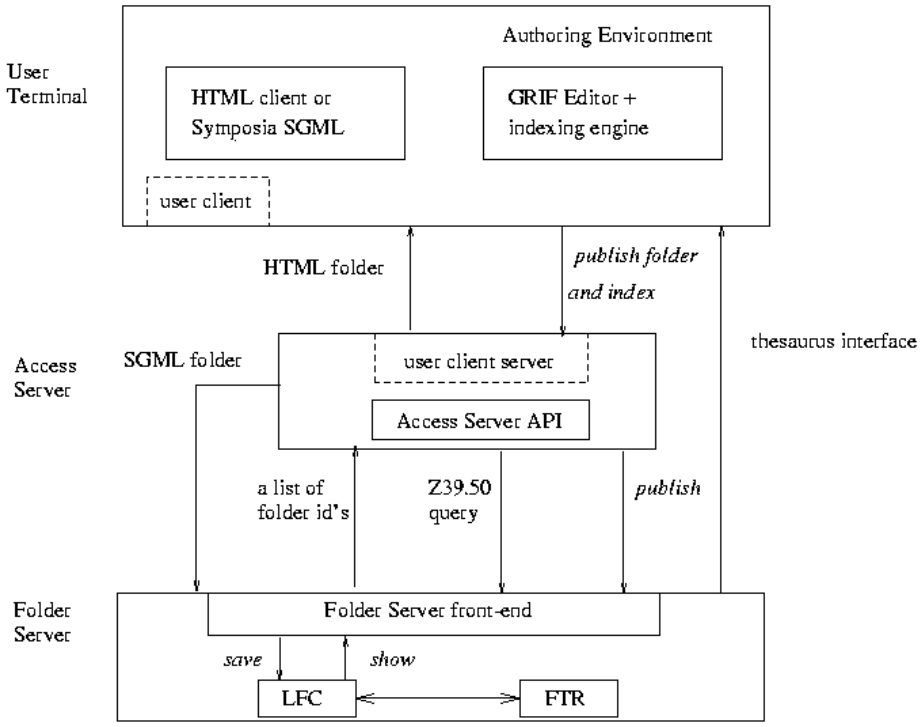
- Οι εξυπηρέτες θεματικής ενότητας (folder servers), στους οποίους η πληροφορία των εξυπηρετών αρχείου οργανώνεται σε θεματικές ενότητες.

- Οι εξυπηρέτες πρόσβασης (access servers), οι οποίοι αποτελούν τις κεντρικές λειτουργικές μονάδες του συστήματος και χρησιμοποιούνται ως μεσολαβητές κατά την επικοινωνία των υπολοίπων τριών λειτουργικών μονάδων.

- Τα τερματικά των χρηστών (user terminals), μέσω των οποίων οι χρήστες συνδέονται στο σύστημα ώστε να τους γίνει διαθέσιμη η υπάρχουσα πληροφορία.

Οι αρχιτεκτονική αυτή φαίνεται και στο Σχήμα 5.1.

Η παρούσα εργασία αποτελεί μέρος της λειτουργικής μονάδας του εξυπηρέτη θεματικής ενότητας. Στο εκάστοτε εξυπηρέτη θεματικής ενότητας υπάρχει, όπως φαίνεται και στο 5.1, ένας τοπικός κατάλογος θεματικής ενότητας (Local Folder Catalogue) καθώς και ένα σύστημα ανάκτησης κειμένων (Full Text Retrieval System). Ο τοπικός κατάλογος θεματικής ενότητας, ο οποίος είναι βασισμένος στο ΣΣΕ, χρησιμοποιείται για τον ευρετηριασμό των κειμένων τα οποία είναι αποθηκευμένα στο σύστημα ανάκτησης κειμένων. Τα κείμενα αυτά παριστούν τα δεδομένα τα οποία βρίσκονται αποθηκευμένα στον εξυπηρέτη αρχείου. Λόγω της ετερογένειας των εξυπηρετών αρχείου, ήταν απαραίτητη η ύπαρξη ενός ομοιόμορφου τρόπου παράστασης των δεδομένων. Η SGML επιλέχθηκε για το σκοπό αυτό και, μέσω της έννοιας του καθορισμού τύπου εγγράφου, επιτεύχθηκε η οργάνωση των δεδομένων σε θεματικές ενότητες. Προκειμένου να αποθηκευτούν έγγραφα SGML στο ΣΣΕ έπρεπε να μετατραπούν στον μοντέλο παράστασης δεδομένων του ΣΣΕ. Για την μετατροπή αυτή χρησιμοποιήθηκε η γεννήτρια μεταφραστών SGML.



Σχήμα 5.1: Γραφική παράσταση της αρχιτεκτονικής του συστήματος AQUARELLE

5.2 Σύστημα Σημασιολογικού Ευρετηριασμού

Το Σύστημα Σημασιολογικού Ευρετηριασμού (Semantic Index System ή SIS) είναι ένα εργαλείο για περιγραφή και τεκμηρίωση μεγάλου πληθυσμού, ιδιόμορφων, εξελισσόμενων και πολλαπλώς συνδεδεμένων δεδομένων. Για το λόγο αυτό είναι κατάλληλο για την παράσταση επιστημονικής, σχεδιαστικής και κατασκευαστικής γνώσης. Το ΣΣΕ αποτελείται από ένα μηχανισμό αποθήκευσης, που βασίζεται σε ένα συνδυασμό οντοκεντρικού μοντέλου και σημασιολογικού δικτύου, και από μια γενική διαλογική επαφή χρήσης για την εισαγωγή και ανάκτηση (retrieval) πληροφοριών. Το μοντέλο των δεδομένων είναι υποσύνολο της γλώσσας παράστασης γνώσης Telos. Το ΣΣΕ και η ενσωματωμένη σε αυτό υλοποίηση του δομικού μέρους της γλώσσας Telos έχουν αναπτυχθεί από τον τομέα Πληροφοριακών Συστημάτων και Τεχνολογίας Λογισμικού του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας (ΙΤΕ).

5.3 Σύντομη Περιγραφή της TELOS

Η TELOS είναι μια γλώσσα παράστασης γνώσης που αρχικά αναπτύχθηκε για την κατασκευή μοντέλων που περιγράφουν πληροφοριακά συστήματα. Οι μηχανισμοί που χρησιμοποιεί είναι ανάλογοι με αυτούς των σημασιολογικών δικτύων. Οι στοιχειώδεις δομές της γλώσσας είναι προτάσεις που παριστάνουν *οντότητες* και *σχέσεις* μεταξύ των οντοτήτων. Οι δομές αυτές μπορούν να συνδυαστούν για να παραστήσουν πολυπλοκότερες. Οι οντότητες της TELOS διακρίνονται σε ατομικές οντότητες και σε κλάσεις. Οι βασικοί μηχανισμοί της γλώσσας είναι οι σχέσεις *ταξινόμησης, απόδοσης γνωρίσματος και γενίκευσης/εξειδίκευσης*, οι οποίοι είναι συνήθεις στα σημασιολογικά δίκτυα. Οι ιεραρχίες ταξινόμησης και γενίκευσης μπορούν να φτάσουν σε απεριόριστο βάθος και δίδεται επίσης και η δυνατότητα πολλαπλής ταξινόμησης. Τόσο οι οντότητες του σημασιολογικού δικτύου όσο και τα γνωρίσματα που αυτές έχουν, αντιμετωπίζονται από την TELOS με ομοιόμορφο τρόπο. Η υλοποίηση της TELOS που χρησιμοποιήθηκε δεν περιλαμβάνει τον μηχανισμό εξαγωγής συμπερασμάτων όπως αυτός περιγράφεται στο [MBJK90]. Παρακάτω δίνεται μια συνοπτική περιγραφή των βασικών μηχανισμών που χρησιμοποιεί η TELOS.

- *Ονοματοδοσία*

  Κάθε οντότητα έχει ένα εσωτερικό, παραγόμενο από το σύστημα, αναγνωριστικό όνομα. Επιπλέον ο χρήστης έχει τη δυνατότητα να ονομάσει ο ίδιος μια οντότητα χρησιμοποιώντας ένα *λογικό όνομα*. Το λογικό όνομα ενός γνωρίσματος είναι της μορφής *x.y*, όπου *x* είναι το λογικό όνομα της οντότητας της οποίας αποτελεί γνώρισμα και *y* είναι το όνομα του ιδίου.

- *Ταξινόμηση*

  Με το μηχανισμό αυτό μια ατομική οντότητα περιγράφεται ως μέλος, περίπτωση, μιας κλάσης της οποίας κληρονομεί τα γνωρίσματα. Μια κλάση είναι και αυτή με τη σειρά της μια οντότητα, άρα μπορεί να είναι περίπτωση μιας άλλης κλάσης. Η TELOS απαιτεί κάθε οντότητα να αποτελεί περίπτωση μιας κλάσης. Έτσι δημιουργείται μια μη φραγμένη ιεραρχία από κλάσεις. Στο κατώτερο επίπεδο τοποθετούνται οι *ατομικές οντότητες* ή *Tokens*, κατόπιν υπάρχουν οι *απλές κλάσεις* περιπτώσεις των οποίων είναι οι ατομικές οντότητες, ακολουθούν οι *μέτα-κλάσεις* περιπτώσεις των οποίων είναι οι απλές κλάσεις, έπονται οι *μετα-μετακλάσεις*, κοκ. Η απαίτηση κάθε οντότητα να ανήκει σε μια κλάση καλύπτεται με την ύπαρξη ειδικών *κλάσεων του συστήματος* ή *ω-κλάσεων*. Οι κλάσεις του συστήματος δεν μπορούν

να αλλαχθούν από τον χρήστη. Αποτελούν τον αρχικό πληθυσμό της βάσης και οποιαδήποτε δεδομένα που θα εισάγει στο σύστημα ο χρήστης πρέπει να συνδεθούν έμμεσα ή άμεσα με αυτές. Ολες οι οντότητες που περιγράφονται με την TELOS ταξινομούνται στην κλάση του συστήματος *Object*. Υποκλάσεις της *Object* είναι οι κλάσεις *Individual* και *Attribute*. Στην κλάση Individual ταξινομούνται οι ατομικές οντότητες, οι κλάσεις, οι μέτα-κλάσεις κοκ. Στην κλάση Attribute ταξινομούνται οι σχέσεις που έχουν οι οντότητες μεταξύ τους, οι κλάσεις σχέσεων, οι κλάσεις από κλάσεις σχέσεων, κοκ. Οι κλάσεις που ορίζει ο χρήστης ταξινομούνται στην κλάση του συστήματος *Class*. Οι ατομικές οντότητες ταξινομούνται στην κλάση *Token*. Οι απλές κλάσεις ταξινομούνται στην κλάση *S_Class*, οι μετακλάσεις στην *M1_Class*, οι μέτα-μέτακλάσεις στην *M2_Class*, κοκ. Επίσης υπάρχουν οι πρωτογενείς τιμές *Integer*, *Real*, *String*. Οι πρωτογενείς τιμές δεν μπορούν να δημιουργηθούν ή να καταστραφούν αλλά μόνο να γίνει αναφορά σε αυτές.

Με την ύπαρξη των κλάσεων του συστήματος η ιεραρχία ταξινόμησης φτάνει μόνο ως εκεί που είναι απαραίτητο. Το πλεονέκτημα που προσφέρεται με την ύπαρξη μη φραγμένης ιεραρχίας είναι ότι το σχήμα της βάσης μπορεί να αντιμετωπιστεί όπως τα δεδομένα και άρα να αλλάζει δυναμικά.

Μία οντότητα μπορεί να ανήκει σε παραπάνω από μία κλάσεις. Με αυτόν τον τρόπο μπορεί η ταξινόμηση να υποκαταστήσει την απόδοση γνωρισμάτων. Με αυτόν τον τρόπο όμως πρέπει να αποδίδονται μόνο εγγενείς ιδιότητες μιας οντότητας ενώ ο μηχανισμός απόδοσης γνωρίσματος είναι πιο κατάλληλος για την περιγραφή των επιφανειακών γνωρισμάτων που πιθανόν να έχει η οντότητα.

- *Απόδοση Γνωρίσματος*

  Με το μηχανισμό αυτό αποδίδονται γνωρίσματα στις οντότητες. Τα γνωρίσματα αυτά μπορούν να θεωρηθούν ως σχέσεις μεταξύ οντοτήτων μιας και έχουν ένα *πεδίο ορισμού*, την οντότητα στην οποία αποδίδονται, και ένα *πεδίο τιμών*, την οντότητα που προσδιορίζει τι τύπου είναι η τιμή του γνωρίσματος. Κάθε γνώρισμα μπορεί να έχει παραπάνω από μια ή και καμία τιμή. Επίσης μπορεί να έχει και αυτό γνωρίσματα, γεγονός που απορρέει από την ισότιμη μεταχείριση οντοτήτων και γνωρισμάτων από την TELOS.

- *Περιορισμός στην ταξινόμηση των γνωρισμάτων*

  Αν ένα γνώρισμα είναι περίπτωση μιας κλάσης γνωρισμάτων τότε το πεδίο ορισμού και το πεδίο τιμών του πρέπει να είναι περιπτώσεις των πεδίων ορισμού και τιμών

της κλάσης γνωρισμάτων στην οποία ανήκει.

- *Γενίκευση (αντίστροφο : εξειδίκευση)*
  Ο μηχανισμός αυτός ισχύει μόνο για κλάσεις, όχι δηλαδή για ατομικές οντότητες, που βρίσκονται στο ίδιο επίπεδο ταξινόμησης. Ορίζει μια σχέση συνόλου - υποσυνόλου μεταξύ των κλάσεων που ονομάζεται *isA* . Αν A *isA* B, με A και B κλάσεις, τότε η A ονομάζεται υποκλάση της B και η B υπερκλάση της A. Η A κληρονομεί όλα τα γνωρίσματα της B και είτε έχει επιπλέον γνωρίσματα είτε περιορίζει το σύνολο τιμών των γνωρισμάτων που κληρονομεί από την B. Μια κλάση μπορεί να έχει παραπάνω από μια υπερκλάσεις. Έτσι η σχέση *isA* υποστηρίζει πολλαπλή και αυστηρή κληρονόμηση.

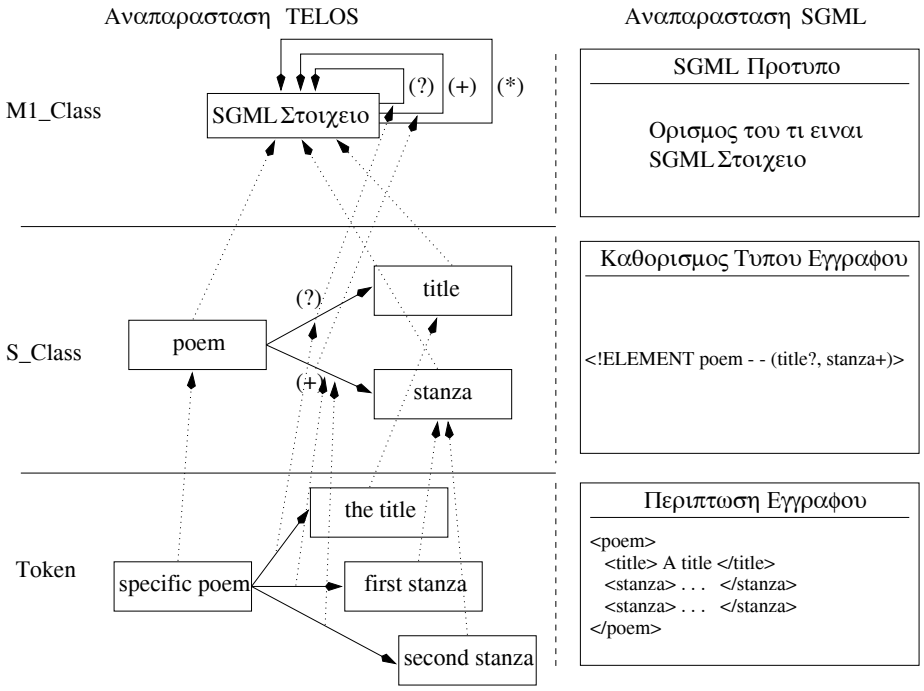
  Ο μηχανισμός γενίκευσης/εξειδίκευσης επιτρέπει την οργάνωση των κλάσεων σε διάφορες ιεραρχίες γενίκευσης οι οποίες προσδίδουν οικονομία και συνέπεια στο μοντέλο, αφού δεν χρειάζεται να επαναληφθεί ο ορισμός ενός γνωρίσματος που έχει ήδη αποδοθεί σε μια γνωστή υπερκλάση.

- *Περιορισμός στις υποκλάσεις γνωρισμάτων*
  Έστω AC1 και AC2 κλάσεις γνωρισμάτων. Αν AC1 *isA* AC2, τότε τα πεδία ορισμού και τιμών της AC1 είναι υποκλάσεις των αντίστοιχων της AC2.

5.4 Περιγραφή διαδικασίας φόρτωσης περιπτώσεων εγγράφων

Το πρώτο βήμα για την επιτυχή φόρτωση των περιπτώσεων εγγράφων στο ΣΣΕ ήταν ο σχεδιασμός ενός μοντέλου, χρησιμοποιώντας την TELOS, το οποίο θα μοντελοποιεί την γλώσσα SGML. Η αρχική σχεδίαση του μοντέλου έγινε από την Αθηνά Τραψιώτη στα πλαίσια της μεταπτυχιακής της εργασίας[??]. Κατόπιν το μοντέλο αναθεωρήθηκε από τους Βασίλη Χριστοφίδη, Martin Doerr, Ειρήνη Φουντουλάκη[CDF97] μέσα στα πλαίσια του προγράμματος AQUARELLE. Το μοντέλο αυτό βρίσκεται στο M1_Class επίπεδο της γλώσσας TELOS. Σε αυτό το επίπεδο μοντελοποιούνται οι βασικές έννοιες της SGML όπως τα στοιχεία, τα γνωρίσματα και οι οντότητες. Τα συγκεκριμένα στοιχεία και γνωρίσματα ενός καθορισμού τύπου εγγράφου μοντελοποιούνται στο S_Class επίπεδο της TELOS και οι οντότητες από τις οποίες αντιπροσωπεύονται αποτελούν περιπτώσεις των οντοτήτων που μοντελοποιούν τις έννοιες αυτές στο M1_Class επίπεδο. Τέλος οι εμφανίσεις στοιχείων μια περίπτωσης εγγράφου μοντελοποιούνται από οντότητες στο Token επίπεδο της TELOS και αποτελούν περίπτωση των οντοτήτων στοιχείων του S_Class

επιπέδου.



Σχήμα 5.2:

Ενα παράδειγμα του μοντέλου φαίνεται στο Σχήμα 5.2. Στο σχήμα παρουσιάζεται ένα μέρος του μοντέλου σε κάθε επίπεδο. Στο M1_Class επίπεδο παρουσιάζεται το μέρος του μοντέλου το οποίο μοντελοποιεί το γεγονός ότι τα περιεχόμενα ενός στοιχείου μπορεί να αποτελείται από άλλα στοιχεία όπου το καθένα μπορεί να έχει τελεστή εμφάνισης είτε το ερωτηματικό είτε το συν είτε τον αστερίσκο. Ο τελεστής εμφάνισης μοντελοποιείται χρησιμοποιώντας διαφορετικό είδος σχέσης για κάθε τελεστή. Για το συγκεκριμένο καθορισμό τύπου εγγράφου poem στο S_Class επίπεδο υπάρχει ένα μοντέλο το οποίο μοντελοποιεί το γεγονός ότι το στοιχείο έχει ένα προαιρετικό τίτλο και μία ή περισσότερες στροφές. Το στοιχείο poem καθώς και τα στοιχεία title και stanza αποτελούν περιπτώσεις της οντότητας SGML Στοιχείο του M1_Class επιπέδου. Τέλος μια συγκεκριμένη περίπτωση εγγράφου εμφανίζεται στο Token επίπεδο, με τις οντότητες της να αποτελούν περιπτώσεις των αντίστοιχων οντοτήτων του S_Class επιπέδου.

Το μοντέλο που υπάρχει τόσο στο S_Class επίπεδο όσο και στο Token επίπεδο παράγεται αυτόματα. Ετσι με δεδομένο ένα καθορισμό τύπου εγγράφου παράγεται το μοντέλο το οποίο παριστάνει την δομή αυτού του τύπου σε TELOS. Κατόπιν και για κάθε περίπτωση εγγράφου παράγεται το μοντέλο σε Token επίπεδο και γίνεται και η

σύνδεση αυτού, μέσω ταξινόμησης (instantiation), με το μοντέλο του αντίστοιχου τύπου στο S_Class επίπεδο.

Για την παραγωγή του μοντέλου του καθορισμού τύπου εγγράφου χρησιμοποιείται η πληροφορία η οποία συγκεντρώνεται από το μέτα-αναλυτή, ενώ για την παραγωγή του μοντέλου της περίπτωσης εγγράφου χρησιμοποιείται η πληροφορία η οποία συγκεντρώνεται από τον αναλυτή. Επίσης χρησιμοποιείται μία λειτουργική μονάδα η οποία παράγει δηλώσεις TELOS, και η οποία κατασκευάστηκε απο τον συγγραφέα μέσα στα πλαίσια της διπλωματικής του εργασίας [Κεμ96]. Η πληροφορία η οποία συλλέγεται κάθε φορά μετατρέπεται σε δηλώσεις TELOS και αποθηκεύεται σε ένα αρχείο. Το αρχείο αυτό χρησιμοποιείται για την φόρτωση του μοντέλου στο ΣΣΕ. Το αρχείο με τις δηλώσεις TELOS που αντιστοιχούν στον καθορισμού τύπου εγγράφου παράγεται μόνο μία φορά, κατά την ανάλυση του τύπου από το μετα-αναλυτή. Από κει και πέρα, για κάθε περίπτωση εγγράφου η οποία αναλύεται από τον παραγόμενο αναλυτή, παράγεται και ένα αρχείο με δηλώσεις TELOS το οποίο χρησιμοποιείται για την φόρτωση της δομής και των περιεχομένων της περίπτωσης στο ΣΣΕ. Θα ήταν επιθυμητό η φόρτωση αυτή να γίνεται απευθείας στο ΣΣΕ, χωρίς την μεσολάβηση αρχείου, μέσω συναρτήσεων ενημέρωσης, συναρτήσεις οι οποίες θα ανήκουν στην προγραμματιστική διεπιφάνεια χρήσης του ΣΣΕ. Δυστυχώς οι συναρτήσεις αυτές δεν ήταν διαθέσιμες κατά την εξέλιξη της παρούσας εφαρμογής. Από την στιγμή που αυτές θα γίνουν διαθέσιμες, και για την βελτιστοποίηση της απόδοσης της εφαρμογής, θα ήταν επιθυμητή η ενσωμάτωση της παραπάνω λειτουργικότητας στο υπάρχον σύστημα.

Κεφάλαιο 6

Επίλογος

Στην παρούσα εργασία παρουσιάστηκε η δημιουργία ενός συντακτικού και λεξικογραφικού αναλυτή εγγράφων, έγγραφα τα οποία ακολουθούν το πρότυπο της SGML. Ο κατασκευασθείς αναλυτής, μέτα-αναλυτής για την ακρίβεια, δέχεται ως είσοδο ένα καθορισμό τύπου εγγράφου και παράγει ως έξοδο ένα αναλυτή για τις περιπτώσεις του δοθέντος τύπου. Τόσο ο μέτα-αναλυτής όσο και ο παραγόμενος αναλυτής είναι αναλυτές πιστοποίησης. Επιπλέον προσφερόμενες υπηρεσίες είναι αυτές της ανίχνευσης αμφισημιών καθώς και οι δυνατότητα απαλοιφής ετικετών. Υπάρχουσες προσεγγίσεις υποστηρίζουν περισσότερες, προς το παρόν, υπηρεσίες αλλά η ανοιχτή αρχιτεκτονική της προτεινόμενης προσέγγισης την κάνει πιο ευέλικτη, πιο ελκυστική, και σαφώς πιο επεκτάσιμη.

Για την υλοποίηση χρησιμοποιήθηκαν τα εργαλεία Bison και Flex και ο κώδικας είναι γραμμένος σε C. Το μέγεθος του κώδικα για το μέτα-αναλυτή είναι περίπου 170KB, γύρω στις 4500 γραμμές, ενώ τα μεγέθη των αρχείων του Bison και Flex είναι 68KB και 34KB αντίστοιχα. Το εκτελέσιμο του μέτα-αναλυτή έχει μέγεθος 360KB. Οσον αφορά τον αναλυτή το μέγεθος του κώδικα είναι περίπου 45KB ενώ το μέγεθος των παραγόμενων αρχείων Bison και Flex εξαρτάται βέβαια από το μέγεθος του καθορισμού τύπου εγγράφου. Ενδεικτικά μεγέθη των αρχείων αυτών για τον καθορισμό τύπου εγγράφου που χρησιμοποιείται στο AQUARELLE είναι 100KB και 45KB αντίστοιχα. Το εκτελέσιμο του αναλυτή είναι, για τον ίδιο τύπο εγγράφου, 275KB. Το σύστημα αναπτύχθηκε στο λειτουργικό σύστημα SUN/UNIX Solaris 2.6 αλλά η μεταφορά του συστήματος σε άλλη πλατφόρμα και σε άλλο λειτουργικό, δεν παρουσιάζει δυσκολίες λόγω των χρησιμοποιηθέντων εργαλείων. Αυτή τη στιγμή η παρούσα υλοποίηση χρησιμοποιείται στο πρόγραμμα AQUARELLE για την συντακτική και λεξικογραφική ανάλυση SGML εγ-

γράφων καθώς και για την φόρτωση της δομής και των περιεχομένων των εγγράφων στο Σύστημα Σημασιολογικού Ευρετηριασμού. Οι μελλοντικές επεκτάσεις του συστήματος συμπεριλαμβάνουν την υποστήριξη του τελεστή ampersand καθώς και την υποστήριξη εξαιρέσεων.

6.1 Μελλοντικές κατευθύνσεις

Η προτεινόμενη προσέγγιση δεν υποστηρίζει εξαιρέσεις και δεν υποστηρίζει τον τελεστή ampersand στο μοντέλο περιεχομένων. Η υλοποίηση των δύο προαναφερθέντων χαρακτηριστικών θεωρήθηκε ότι είναι έξω από τα πλαίσια της παρούσης εργασίας. Η απόφαση αυτή πάρθηκε λαμβάνοντας υπόψη τόσο το φόρτο της εργασίας όσο και το χρονικό πλαίσιο μέσα στο οποίο αυτή έπρεπε να ολοκληρωθεί. Αυτό δεν σημαίνει ότι τα χαρακτηριστικά αυτά είναι δευτερεύουσας σημασίας, και δεν πρέπει να υλοποιηθούν. Αποτελούν τον κύριο άξονα πάνω στο οποίο θα κινηθούν οι μελλοντικές επεκτάσεις. Η υλοποίηση των προαναφερθέντων μπορεί να ενσωματωθεί, στο μέλλον, στην παρούσα υλοποίηση κάνοντας τα εξής. Εστω ένα μοντέλο περιεχομένων που έχει σαν τελεστή σύνδεσης το ampersand όπως φαίνεται και στο ακόλουθο παράδειγμα.

Παράδειγμα 6.1 *:*

$$<!ELEMENT\ a\ -\ -\ (b\ \&\ c\ \&\ d)>$$

Η παραγόμενη γραμματική, σαν μοντέλο περιεχομένων του στοιχείου a θα πρέπει να μπορεί να αναγνωρίζει όλες τις πιθανές μεταθέσεις των στοιχείων b, c, d. Μια απλή υλοποίηση θα ήταν να παράγονται όλες αυτές οι μεταθέσεις και να ενσωματώνονται στην γραμματική όπως φαίνεται ακολούθως

Παραγωγή 6.1 *:*

$$\begin{aligned} a \rightarrow\ & b\ c\ d \\ & b\ d\ c \\ & c\ b\ d \\ & c\ d\ b \\ & d\ b\ c \\ & d\ c\ b \end{aligned}$$

Η προσέγγιση αυτή όμως θα αύξανε το μέγεθος της γραμματικής του παραγόμενου αναλυτή εκθετικά, για το λόγο αυτό προτείνεται η ακόλουθη: προσέγγιση

Παραγωγή 6.2 *:*

a → a_*model* a_*model* a_*model*

Παραγωγή 6.3 *:*

a_*model* → *b*
| *c*
| *d*

Στην προτεινόμενη υλοποίηση παράγεται, κατ'αρχήν, μία παραγωγή στην οποία επαναλαμβάνεται το μη τερματικό στοιχείο της γραμματικής a_model τόσες φορές όσες είναι ο πληθάριθμος των στοιχείων τα οποία αποτελούν το μοντέλο περιεχομένων του στοιχείου a. Το μη τερματικό a_model είναι μία παραγωγή η οποία μπορεί να οδηγήσει σε b ή c ή d. Η γλώσσα την οποία αναγνωρίζει η παραγόμενη γραμματική αποτελεί βέβαια υπερσύνολο της ζητούμενης γλώσσας αφού αναγνωρίζεται και η ύπαρξη τριών b. Με τη χρήση όμως καταλλήλων κανόνων σημασιολογικού περιεχομένου στην παραγόμενη γραμματική, μπορεί να εξασφαλιστεί μοναδική εμφάνιση των b, c, d. Τέλος ένα πρόβλημα το οποίο χρήζει ιδιαίτερης προσοχής και απαιτείται παραπέρα μελέτη είναι η εξιχνίαση και η αντιμετώπιση των αμφισημιών σε ένα μοντέλο περιεχομένων το οποίο περιέχει τον εν λόγω τελεστή.

Οσον αφορά την ενσωμάτωση αναγνώρισης εξαιρέσεων αυτό που προτείνεται είναι η κατασκευή ενός μηχανισμού ο οποίος λειτουργεί ως εξής. Η γραμματική του παραγόμενου αναλυτή θα διατηρηθεί ως έχει. Οταν ένα στοιχείο εμφανίζεται σε μία περίπτωση εγγράφου, και το στοιχείο αυτό δεν ανήκει στο εξεταζόμενο εκείνη τη στιγμή μοντέλο περιεχομένων, τότε παράγεται λάθος από την γραμματική και καλείται η ρουτίνα yyerror του Bison. Στην παρούσα υλοποίηση αυτό που γίνεται είναι κατευθείαν να εκτυπώνεται στην οθόνη ένα μήνυμα συντακτικού λάθους. Αντί αυτού όμως, προτείνεται να ελέγχεται αν το στοιχείο αυτό ανήκει στις εξαιρέσεις του μοντέλου περιεχομένων και, αν αυτό είναι αληθές, ένας μηχανισμός να αναλαμβάνει τη συνέχιση της συντακτικής ανάλυσης λαμβάνοντας υπόψη το στοιχείο, το οποίο συναντήθηκε. Σε αντίθετη περίπτωση το μήνυμα λάθους θα εκτυπώνεται στην οθόνη.

Τέλος, μία σημαντική επέκταση του συστήματος είναι η δημιουργία διεπιφάνειας χρήσης του συστήματος μέσω της γλώσσας Java. Θα δοθεί έτσι η δυνατότητα της ανάλυσης και παρουσίασης της δομής εγγράφων γραμμένων σε SGML, με τη χρήση ευρέως διαδεδομένων εργαλείων αναδύφησης του διαδικτύου, όπως το Netscape Navigator και το Internet Explorer. Ο χρήστης θα μπορεί, χρησιμοποιώντας τα προαναφερθέντα εργαλεία, να καθορίζει το έγγραφο SGML το οποίο θέλει να αναλύσει. Κατόπιν θα

καλείται ο αναλυτής ο οποίος θα επιτελεί την ανάλυση του εγγράφου και, μετά την ολοκλήρωσή της, θα εμφανίζεται στο εργαλείο αναδύφησης ένας γράφος ο οποίο θα παριστά την δομή του αναλυμένου εγγράφου. Οι κόμβοι του γράφου θα παριστούν τα στοιχεία του εγγράφου. Μαρκάροντας ένα κόμβο, ο χρήστης θα μπορεί να ζητά επιπλέον πληροφορία για αυτόν. Η προγραμματιστική διεπιφάνεια χρήσης, η οποία μελετήθηκε στο Κεφάλαιο 4, θα χρησιμοποιείται ώστε να συλλέγονται οι απαιτούμενες πληροφορίες απο τον αναλυτή και κατάλληλος μηχανισμός θα εμφανίζει τις πληροφορίες αυτές στο εργαλείο αναδύφησης.

Η προτεινόμενη επέκταση αυτή είναι δυνατόν να επιτευχθεί λόγω της ύπαρξης, αυτή τη στιγμή, εργαλείων τα οποία δέχονται ως είσοδο μία Bison γραμματική και παράγουν ως έξοδο κώδικα Java. Ενα από τα εργαλεία αυτά είναι το BYACC/Java [Jam98].

Βιβλιογραφία

[Κεμ96] Α. Κεμεντσιετσίδης. Σύνδεση ενός SGML Editor με το Σύστημα Σημασιολογικού Ευρετηριασμού. Διπλωματική Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Αύγουστος 1996.

[AO83] J. Albert and T. Ottmann. *Automaten, Sprachen und Maschinen fuer Anwender*. Bibliographisches Institut, Mannheim, 1983.

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers*. Addison-Wesley, Reading, Massachusetts, 1986.

[B+71] R. Book et al. Ambiguity in graphs and expressions. *IEEE Transactions on Computers*, 20(2):149--153, 1971.

[BK94a] A. Brüggemann-Klein. Regular expressions into finite automata. Technical Report Aug16-4, Technical University of Munich, August 16, 1994.

[BK94b] A. Brüggemann-Klein. Unambiguity of extended regular expressions in SGML document grammars. Technical Report Aug16-1, Technical University of Munich, August 16, 1994.

[BKW93] A. Brüggemann-Klein and Derick Wood. The validation of SGML content models. Technical Report 355, Computer Science Department, University of Western Ontario, London, Ontario, Canada, March 1993.

[BKW94] A. Brüggemann-Klein and D. Wood. Deterministic regular languages. Technical Report Aug16-3, Technical University of Munich, August 16, 1994.

[BS86] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48(1):117--126, 1986.

[CD94] P. Constantopoulos and M. Dörr. The Semantic Index System : A brief presentation. Institute of Computer Science, Foundation for Research and Technology-Hellas, May 1994. http://www.ics.forth.gr/proj/isst/Systems/sis.

[CDF97] V. Christophides, M. Doerr, and I. Fundulaki. A semantic network approach to semi-structured documents repositories. *Lecture Notes in Computer Science*, 1324:305--329, 1997.

[Cla94] J. Clark. SGMLS Parser. Available at http://www.sil.org/sgml/publicsw.html, December 1994.

[Cla97] J. Clark. SGML Parser, (SP). Available at http://www.jclark.com/sp/index.htm, September 1997.

[COM] SGML discussion mailing list. http://CandL.let.ruu.nl/archive/cts/html/hub.htm.

[Con97] D. Connolly, editor. *XML: Principles, Tools, and Techniques*, volume 2(4) of *The World Wide Web Journal*. O'Reilly Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, and 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, Winter 1997.

[DS90] C. Donnelly and R. Stallman. *Bison: the Yacc-compatible parser generator*. Free Software Foundation, Cambridge, MA, USA, Version 1.12 edition, December 1990.

[Eng97] J. English. Inclusion and exclusion exceptions. Available at http://www.sil.org/sgml/inclusionenglish970818.html, 1997.

[GJ88] D. Grune and C. J. H. Jacobs. A programmer-friendly LL(1) parser generator. *Software Practice and Experience*, 18(1):29--38, January 1988.

[Glu61] V. M. Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1--53, 1961.

[Gol92] C. Goldfarb. Arc-SGML: IBM Almaden Research Center SGML Parser. Available at http://www.sil.org/sgml/publicsw.html#arc-sgml, 1992.

[GR90] C. F. Goldfarb and Y. Rubinsky. *The SGML handbook*. Clarendon Press, Oxford, UK, 1990.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introdction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., USA, 1979.

[IBM] IBM. SGML Translator. http://booksrv2.raleigh.ibm.com:80/cgi-bin/BookMgr/bookmgr.cmd/BOOKS/EHMGMA00/CC.

[INR] INRIA. AQUARELLE. http://aquarelle.inria.fr.

[Ins97] Institute Of Computer Science (FORTH) - Hellas. *SIS - Semantic Index System*, version 2.1 edition, May 1997.

[ISO86] ISO. *Information Processing - Text and Office Systems - Standard General MarkUp Language (SGML).* ISO8879, 1986.

[ISO89] ISO. *Information Processing - Text and Office Systems - Office Document Architecture (ODA) and interchange format, Part 1,2,4,5,6,7,8.* ISO8613, 1989.

[ISO95] ISO. *Information technology - Processing languages Standard Page Description Language (SPDL).* ISO/IEC 10180:1995, 1995.

[ISO96] ISO. *Information technology - Processing languages - Document Style Semantics and Specification Language (DSSSL).* ISO/IEC 10179:1996, 1996.

[Jam98] B. Jamison. BYACC/Java, an extension of YACC to produce Java parsers. Available at http://www.lincom-asg.com/ rjamison/byacc. LinCom Innovations, 1998.

[Joh75] S. C. Johnson. YACC: Yet another compiler compiler. *Computing Science TR*, 32, 1975.

[KR78] B. W. Kernighan and D. M. Ritchie. *The C programming language.* Prentice-Hall software series. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1978.

[Lam86] L. Lamport. LaTeX*: a document preparation system.* Addison-Wesley, Reading, MA, USA, 1986.

[Loe94] A. Loeffen. SGML: Simplification and disambiguation revisited. Available at http://www.sil.org/sgml/loeftabl.html. University of Utrecht, November 1994.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing Knowledge about Information Systems. *ACM Transactions on Information Systems*, 8(4):325--362, 1990.

[NBL93] B. Nordin, D. T. Barnard, and I. A. Mac Leod. A review of the Standard Generalized Markup Language (SGML). *Computer Standards and Interfaces*, 15(1):5--19, May 1993.

[Pap92] H. R. Lewis και Χ. Παπαδημητρίου (H. H. Papadimitriou). *Στοιχεία Θεωρίας Υπολογισμού (Elements of the Theory of Computation)*. Τεχνικό Επιμελητίριο Ελλάδος, 1992.

[Pax88] V. Paxson. Flex - fast lexical analyzer generator. Free Software Foundation, Cambridge, MA, USA, 1988. Available via anonymous ftp to *lbl-csam.arpa*, *lbl-rtsg.arpa*, or *prep.ai.mit.edu*.

[Ric97] P. Richard. Yorktown Advanced SGML Parser (YASP). Available at http://www.sil.org/sgml/publicsw.html#yasp. Electricitι de France, Direction des Etudes et Recherches, April 1997.

[Sof] SoftQuad. Panorama. http://www.softquad.com/products/pc-ppub.htm.

[SSS88] S. Sippu and E. Soisalon-Soininen. *Parsing Theory: Languages and Parsing*, volume 1. Springer Verlag, Berlin, Heidelberg, New York, 1988.

[WE89] J. Warmer and S. Van Egmond. The implementation of the Amsterdam SGML Parser. *Electronic PublishingOrigination, Dissemination, and Design*, 2(2):65--90, July 1989.

[XER] XEROX. Astoria. http://www.chrystal.com/products/index.html.