ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Αμφίδρομη Μετάφραση της Σύνταξης Παρουσίασης και Σειριοποίησης της PSOA RuleML

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Φωλίνας Ευάγγελος

Επιβλέπων: Στεφανέας Πέτρος
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Αμφίδρομη Μετάφραση της Σύνταξης Παρουσίασης και Σειριοποίησης της PSOA RuleML

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Φωλίνας Ευάγγελος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

.....	.....	.....
Στεφανέας Πέτρος Επίκουρος	Θεοδώνης Ιωάννης	Συμβώνης Αντώνιος
Καθηγητής Ε.Μ.Π. (Επιβλέπων)	Ε.ΔΙ.Π. Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2022

Φωλίνας Ευάγγελος
Διπλωματούχος Μαθηματικός Εφαρμογών Ε.Μ.Π.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου, Πέτρο Στεφανέα για την εμπιστοσύνη που μου έδειξε, καθώς επίσης και τα μέλη της εξεταστικής επιτροπής. Τον υποψήφιο διδάκτορα Θοδωρή Μήτσικα, η βοήθεια του οποίου ήταν καθοριστική για την πραγματοποίηση της εργασίας. Την οικογένειά μου που με στηρίζει πάντα. Την Σίσσυ, τον Βασίλη και όλους όσους απαντούν στο ”φίλε, αδερφέ, μα πάνω απ’ όλα σύντροφε” που είναι πάντα δίπλα μου.

Περίληψη

Η παρούσα διπλωματική εργασία αποσκοπεί στην πραγματοποίηση αμφίδρομης μετατροπής των μορφών Σύνταξης Παρουσίασης (Presentation Syntax) και Σύνταξης Σειριοποίησης (Serialization Syntax) της PSOA RuleML μέσω αφηρημένων συντακτικών δέντρων, με σκοπό την αποτελεσματικότερη λειτουργία της εν λόγω γλώσσας. Στον βαθμό που αποτελούν προϋπόθεση της εν λόγω μετατροπής, εκτίθενται τα βασικά στοιχεία αυτών των μορφών και εξηγούνται οι κανόνες που τις διέπουν, ανατρέχοντας στα αντίστοιχα παραδείγματα. Η υλοποίηση του μεταφραστή πραγματοποιήθηκε επεκτείνοντας το ήδη υπάρχων σύστημα PSOATransRun στη γλώ- σσα προγραμματισμού Java, χρησιμοποιώντας το εργαλείο ANTLR για τη δημιουργία του λεκτικού και συντακτικού αναλυτή.

Λέξεις Κλειδιά: RuleML, XML, Αφηρημένα Συντακτικά Δέντρα, Java, ANTLR

Abstract

The present Diploma's research thesis deals with the bidirectional translation of PSOA RuleML in Presentation Syntax form as well as Serialization Syntax by virtue of abstract syntax trees, for more efficient operation of the programming language. By referring to the respective examples, the basic elements as well as the rules governing that forms are presented, insofar their clarification builds the precondition for the issue at stake. The realization of the translation program is performed by expanding the already existed system PSOATransRun by using the programming language Java through the tool ANTRL for creating the lexer and parser rule.

Keywords: RuleML, XML, Abstract Syntax Tree, Java, ANTLR

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Γενικά

Οι Γλώσσες Κανόνων (Rule Languages) για τον Σημασιολογικό Ιστό, όπως είναι η RuleML και η RIF, έχουν σχεδιαστεί με σκοπό την ανταλλαγή κανόνων και ερωτημάτων μεταξύ συστημάτων αναπαράστασης γνώσης. Αυτό πραγματοποιείται με την ανταλλαγή δεδομένων σε μηχαναγνώσιμη μορφή, όπως για παράδειγμα σε μορφή XML. Καθώς όμως δεδομένα σε XML είναι συνήθως δύσκολα αναγνώσιμα από τους ανθρώπους, οι παραπάνω γλώσσες ορίζουν και εναλλακτική αναπαράσταση, στοχεύοντας στην εύκολη ανάγνωση από ανθρώπους. Συνεπώς, η μία μορφή αναπαράστασης είναι η Σύνταξη Σειριοποίησης (Serialization Syntax), η οποία είναι για τη βέλτιστη ανάγνωση από τους υπολογιστές. Η άλλη είναι η μορφή της Σύνταξης Παρουσίασης (Presentation Syntax), που είναι πιο εύκολα αναγνώσιμη από τους ανθρώπους.

Ως εκ τούτου, μία αμφίδρομη μετάφραση μεταξύ αυτών των δύο μορφών είναι σημαντική για να διευκολύνει και τον άνθρωπο αλλά και τον υπολογιστή στην ανάγνωση και ανταλλαγή πληροφοριών. Ο μεταφραστής θα μπορεί αυτόματα να αλλάξει την πληροφορία στην μορφή της επιλογής του ανθρώπου, ανάλογα με τη χρήση που θέλει να κάνει, δηλαδή να παίξει το ρόλο της δικλείδας επικοινωνίας των μορφών αναπαράστασης γνώσης.

1.2 Αντικείμενο Διπλωματικής

Οι γλώσσες κανόνων αποτελούν σημαντικό συστατικό του Σημασιολογικού Ιστού και έχουν σκοπό τον ορισμό και την ανταλλαγή κανόνων, καθώς και την πραγματοποίηση συλλογισμών στα πλαίσιά του. Μια τέτοια γλώσσα κανόνων είναι και η PSOA RuleML, και έχει δυο μορφές σύνταξης: Την Σύνταξη Σειριοποίησης (Serialization Syntax), που στοχεύει στην ανταλλαγή κανόνων μεταξύ συστημάτων και είναι σε μορφή XML (PSOA-XML), καθώς και την Σύνταξη Παρουσίασης (Presentation Syntax, PSOA-PS) η οποία στοχεύει στο να είναι πιο ευανάγνωστη στον άνθρωπο.

Η παρούσα διπλωματική εργασία έχει ως κύριο αντικείμενο την υλοποίηση της αμφίδρομης μετατροπής μεταξύ των δυο μορφών αναπαράστασης της γλώσσας PSOA RuleML, επεκτείνοντας το ήδη υπάρχον σύστημα PSOATransRun.

Στόχος είναι δηλαδή να υπάρξει μία σύνδεση μεταξύ της μορφής της Σύνταξης Σειριοποίησης και της Σύνταξη Παρουσίασης, ώστε να επιτρέπει στον χρήση απρόσκοπτη ανάγνωση μιας βάσης κανόνων που είναι σε Σύνταξη Σειριοποίησης με την αυτόματη μετάφραση της μορφής αυτής, ή αντίστοιχα, την μετατροπή βάσεων κανόνων που είναι στη μορφή της Σύνταξη Παρουσίασης και χρειάζεται να μετατραπεί σε XML για οποιαδήποτε επεξεργασία από υπολογιστή.

Σημαντικό ρόλο για την εν λόγω μετάφραση αποτελεί η έννοια των Αφηρημένων Συντακτικών Δέντρων (Abstract Syntax Trees, AST), καθώς και το εργαλείο ANTLR για την συντακτική ανάλυση και μετατροπή μιας βάσης κανόνων στο ανάλογο Αφηρημένο Συντακτικό Δέντρο, όπως και την περαιτέρω μετατροπή του στην επιθυμητή μορφή (PSOA-PS ή PSOA-XML). Το υπάρχον σύστημα PSOATransRun χρησιμοποιεί ήδη το συγκεκριμένο εργαλείο για την ανάγνωση τον βάσεων κανόνων (σε PSOA-PS) και την μετατροπή τους σε AST και στη συνέχεια σε εκτελέσιμο κώδικα (γλώσσας Prolog ή TPTP). Η επέκτασή του έτσι ώστε να διαβάζει PSOA-XML και να παράγει το συντακτικό του δέντρο, καθώς και να μετατρέπει συντακτικά δέντρα σε PSOA-XML ή PSOA-PS είναι μια σημαντική προσθήκη που θα διευκολύνει την χρήση της PSOA RuleML και από τον άνθρωπο και τον υπολογιστή.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Παρουσίαση των μορφών της Σύνταξης Παρουσίασης, της Σύνταξης Σειριοποίησης καθώς και των Αφηρημένων Συντακτικών Δέντρων, μέσα από τα κατάλληλα παραδείγματα

2. Καταγραφή των κανόνων της γλώσσας PSOA RuleML.

3. Δημιουργία κώδικα σε ANTLR για τη δημιουργία των κλάσεων που θα πραγματοποιούν την αμφίδρομη μετατροπή των μορφών σε αφηρημένα συντακτικά δέντρα.

4. Ενσωμάτωση των κατάλληλων μεθόδων στον ήδη υπάρχων κώδικα του PSOATransRun που είναι σε Java για την πραγματοποίηση και εμφάνιση των μεταφράσεων.

5. Παρουσίαση των συμπερασμάτων και κάποιων μελλοντικών επεκτάσεων.

1.3 Οργάνωση Κειμένου

Η διπλωματική εργασία στο κεφάλαιο 2 παρουσιάζει τη μορφή Σύνταξης Παρουσίασης. Στο κεφάλαιο 3 παρουσιάζεται η μορφή της Σύνταξης Σειριοποίησης, αφού εξηγηθούν οι βασικοί κανόνες της XML. Το κεφάλαιο 4 αφορά τα Συντακτικά Δέντρα, το ρόλο τους για τη μετάφραση και την απεικόνισή τους. Στα κεφάλαια 2, 3 και 4 αξιοποιούνται ορισμένα παραδείγματα για την ευκολότερη κατανόησή τους. Στο κεφάλαιο 5 εξηγείται η λειτουργία του προγράμματος που επεκτάθηκε, παρουσιάζονται οι μορφές όπως είναι κατά την εισαγωγή στο πρόγραμμα, κατά τη διαδικασία εκτέλεσης και μετά το πέρας αυτής. Τέλος, το κεφάλαιο 6 περιλλαμβάνει τα συμπεράσματα της εργασίας καθώς και πιθανές μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ 2

ΣΥΝΤΑΞΗ ΠΑΡΟΥΣΙΑΣΗΣ ΤΗΣ PSOA RULEML

Η μορφή της Σύνταξης Παρουσίασης (Presentation Syntax) της PSOA RuleML είναι ένα συντακτικό που παρουσιάζει με πιο κατανοητό τρόπο για τον άνθρωπο την αναπαράσταση γνώσης. Η βάση της είναι ο ατομικός τύπος (atom).

Ένας ατομικός τύπος είναι ένα γεγονός (fact). Περιέχει κάποιο κατηγόρημα, το οποίο πρακτικά δηλώνει το θέμα του, και τα ανάλογα ορίσματά του (descriptors). Τα ορίσματα χωρίζονται σε δύο κατηγορίες:

- πλειάδες (tuples), τα οποία αποτελούνται από διατεταγμένα στοιχεία
- slots, τα οποία είναι ένα ζεύγος από την ιδιότητα και το όνομα του ορίσματος.

Ένας ατομικός τύπος μπορεί να αποτελείται μόνο από πλειάδες, μόνο από slots, αλλά και από ένα συνδυασμό αυτών. Τα ορίσματα, ανεξάρτητα από το είδος τους, μπορεί είναι εξαρτημένα (dependent) του κατηγορήματος ή ανεξάρτητα (independent). Η εξάρτηση αυτή είναι ένα από τα δύο κριτήρια για τον χαρακτηρισμό του ατομικού τύπου. Ακόμα, τα ορίσματα μπορεί να είναι σταθερά ή μεταβλητά εκτός από το κατηγόρημα και την ιδιότητα στα slots.

Σε έναν ατομικό τύπο μπορεί να υπάρχει αναγνωριστικό αντικειμένου, Object IDentifier (OID). Το αναγνωριστικό αντικειμένου με την ύπαρξή του προσδιορίζει τη μοναδικότητα του ατομικού τύπου. Υπάρχει και η δυνατότητα απουσίας του. Αν υπάρχει αναγνωριστικό αντικειμένου (oidful) στον ατομικό τύπο ή όχι (oidless) είναι το δεύτερο κριτήριο για τον χαρακτηρισμό του. [2]

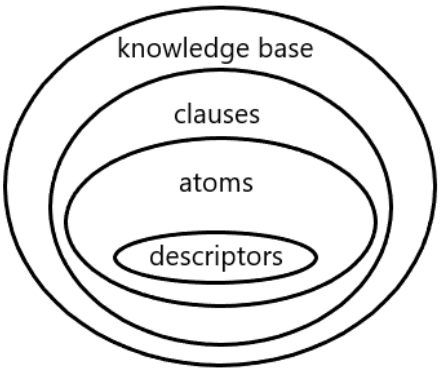
Η εξάρτηση, ή όχι, των ορισμάτων και η ύπαρξη, ή όχι, ενός αναγνωριστικού αντικειμένου που δομούν έναν ατομικό τύπο είναι τα χαρακτηριστικά που κρίνουν το γεγονός. Στον ακόλουθο πίνακα φαίνονται τα πιθανά είδη ενός γεγονότος.

dependent	oidless (”...ships”)	oidful (”...points”)
tupled (”relation...”)	relationships	relationpoints
slotted (”pair...”)	pairships	pairpoints

independent	oidless (”...ships”)	oidful (”...points”)
tupled (”shelf...”)	shelfships	shelfpoints
slotted (”frame...”)	frameships	framepoints

[3]

Οι ατομικοί τύποι αναφέρονται συλλογικά ως προτάσεις (clause) και οι προτάσεις είναι τα βασικά στοιχεία που περιέχει μία βάση γνώσης (knowledge base). Αν ένας ατομικός τύπος είναι κενός σημαίνει ότι δεν περιέχει ορίσματα. Αποτέλεσμα αυτού θα είναι η πρόταση να μην περιέχει κανέναν ατομικό τύπο και η βάση γνώσης καμία πρόταση. Στο σχήμα που ακολουθεί φαίνονται οι σχέσεις που έχουν μεταξύ τους τα ορίσματα, οι ατομικοί τύποι, οι προτάσεις και οι βάσεις γνώσης.



Σχήμα 2.1: Σύνολα

2.1 Παρουσίαση της Σύνταξης Παρουσίασης

Για να γίνει πιο εύκολα κατανοητή η μορφή της Σύνταξης Παρουσίασης θα χρησιμοποιηθούν μια σειρά από παραδείγματα, με διάφορα είδη από γεγονότα. Για αρχή θα παρουσιαστεί ο τρόπος με τον οποίο συντάσσεται κάθε ατομικός τύπος και μετά οι κανόνες που υποστηρίζει η PSOA RuleML. Στο τέλος θα γίνει μία αναφορά στον τρόπο σύνταξης της ισότητας και της υποκλάσης. [4]

Ατομικός Τύπος (Atom)

Στη Σύνταξη Παρουσίαση της PSOA RuleML μία αγορά, ’purchase’, σαν κατηγόρημα με πλειάδα που αποτελείται από τρία ορίσματα, αγοραστή ’Vag’, πωλητή ’Teo’ και αντικείμενο ’Hornet’, μπορεί να αναπαρασταθεί στην πιο απλή του μορφή ως relationship γεγονός ως εξής:

```
purchase (Vag Teo Hornet)
```

Σε αυτή την πιο απλή μορφή, τα ορίσματα του κατηγορήματος, 'purchase', περιβάλλονται από παρενθέσεις, μετά το 'purchase(...)', και μεταξύ των ορισμάτων '(Vag Teo Hornet)' υπάρχει το κενό διάστημα. Από τη στιγμή που δεν έχει επισημανθεί με κάποιο τρόπο, η σχέση των ορισμάτων με το κατηγόρημα θεωρούνται εξαρτώμενα από αυτό.Το παραπάνω παράδειγμα είναι μια συνοπτική έκδοση με όλα τα σύμβολα να θεωρούνται τοπικά (local), δηλαδή αφορούν σύμβολα της τρέχουσας βάσης γνώσης. Το γεγονός ότι τα σύμβολα είναι τοπικά μπορεί να γίνει σαφές με τη χρήση κάτω παύλας για το κατηγόρημα, '_purchase', καθώς και για τα τρία ορίσματά του, '_Vag', '_Teo' και '_Hornet' (το πρόθεμα "_" κάτω παύλα υποδηλώνει τοπικά σύμβολα, σε αντίθεση με το πρόθεμα "άνω και κάτω τελεία (:)", που δηλώνει ότι μπορούν να χρησιμοποιηθούν και σε άλλες βάσεις γνώσης):

```
_purchase(_Vag _Teo _Hornet)
```

Εάν ο ατομικός τύπος αποτελείται από πλειάδα που τα ορίσματά του είναι ανεξάρτητα από το κατηγόρημα, πρόκειται δηλαδή για shelfship γεγονός, πρέπει να επισημανθεί με το σύμβολο "-" και τα ορίσματα να περικλείονται από "τετράγωνες αγκύλες ([])". Η μορφή του κειμένου θα είναι κάπως έτσι:

```
_purchase(-[_Vag _Teo _Hornet])
```

Δεδομένου ότι η σειρά των ορισμάτων μίας πλειάδας είναι σημαντική, οι δύο ακόλουθες παραλλαγές γεγονότων δεν είναι ισοδύναμες:

```
purchase(Vag Teo Hornet)
```

```
purchase(Teo Vag Hornet)
```

Οι διάφοροι «κρυμμένοι ρόλοι» σε μια σχέση, όπως σε αυτό το παράδειγμα του αγοραστή, του πωλητή και του αντικειμένου, θα βοηθούσε να δηλώνονται με κάποιο ξεκάθαρο τρόπο, κάτι που κρίνεται απαραίτητο όταν πρόκειται για πολλά ορίσματα. Για αυτό ο ατομικός τύπος μπορεί να αναπαρασταθεί σαν pairship γεγονός, με τη χρήση slots. Στο παραπάνω παράδειγμα δηλαδή θα μπορούσαν να χρησιμοποιηθούν οι ιδιότητες ενός slot 'buyer', 'seller', 'item' (το επίθεμα "+>" αντιστοιχίζει την ιδιότητα με το γνώρισμα, το "+" δηλώνει την εξάρτηση του ορίσματος από το κατηγόρημα):

```
purchase(buyer+>Vag seller+>Teo item+>Hornet)
```

Εδώ τα τρία σύμβολα "+>" στους συνδυασμούς ιδιότητα με γνώρισμα κάνουν σαφή ότι υπάρχουν πολλαπλά εξαρτώμενα από το κατηγόρημα ορίσματα. Επειδή κάθε όρισμα σε ένα slot είναι ένα διατεταγμένο ζεύγος ιδιότητας με γνώρισμα, η σειρά τους δεν έχει σημασία, σε αντίθεση με τις πλειάδες, για αυτό το παραπάνω slot είναι ισοδύναμο με οποιαδήποτε σειρά και αν είναι διατεταγμένα. Για παράδειγμα το προηγούμενο pairship γεγονός βγάζει το ίδιο νόημα με το ακόλουθο:

```
purchase(seller+>Teo buyer+>Vag item+>Hornet)
```

Για τη μετατροπή του γεγονότος από pairship σε frameship, δηλαδή τα ορίσματα του slot να είναι ανεξάρτητα από το κατηγόρημα, αρκεί η αντικατάσταση του συμβόλου ”συν (+)” σε ”πλην (-)”. Θα γίνει δηλαδή:

```
purchase(buyer->Vag seller->Teo item->Hornet)
```

Ένας ατομικός τύπος που περιέχει slot διαφέρει από κάποιον άλλον αν έχουν διαφορετικό αριθμό ορισμάτων. Το παρακάτω pairship γεγονός διαφέρει με το προηγούμενο για αυτό το λόγο:

```
purchase(buyer+>Vag seller+>Teo item+>Hornet price+>2200))
```

Σε κάποιον ατομικό τύπο μπορεί να προκύψει η ανάγκη επισήμανσης της μοναδικότητάς του για να την αποφυγή οποιονδήποτε συγχύσεων. Λύση σε αυτό, όπως έχει ήδη αναφερθεί, έρχεται να δώσει το αναγνωριστικό αντικειμένου. Η εισαγωγή του αναγνωριστικού αντικειμένου σε κάποιον ατομικό τύπο καθορίζει το είδος του γεγονότος που δηλώνει. Στη σύνταξη η θέση του είναι πριν από το κατηγόρημα και χωρίζεται από αυτό με μία ”δίεση (#) (hash)”. Ενδεικτικά αν ο ρόλος του αναγνωριστικού αντικειμένου παίζεται από το ’transaction’ η μορφή που θα έχει ένα framepoint είναι:

```
transaction#purchase(buyer->Vag seller->Teo item->Hornet)
```

Ενδέχεται να υπάρξει ο ίδιος ατομικός τύπος αλλά για διαφορετική χρήση. Αν για παράδειγμα υπάρχει ένα pairpoint γεγονός, με αναγνωριστικό αντικειμένου τη μεταβλητή ’ο’ (η μεταβλητή δηλώνεται με το ”αγγλικό ερωτηματικό (?)” πριν από αυτή):

```
?o#purchase(buyer+>Vag seller+>Teo item+>Hornet)
```

και υπάρχει και ένα ίδιο pairpoint γεγονός με τα ίδια χαρακτηριστικά, δηλαδή ίδιο κατηγόρημα και ίδια ορίσματα, αλλά δεν έχει κάποιο συγκεκριμένο αναγνωριστικό αντικειμένου, τότε ορίζεται αυτόματα από τη μορφή της Σύνταξης Παρουσίασης μία τοπική σταθερά, συνήθως ένα νούμερο, και παίρνει την ακόλουθη μορφή:

```
_1#purchase(buyer+>Vag seller+>Teo item+>Hornet)
```

Στην περίπτωση όπου ένας ατομικός τύπος αποτελείται από δύο γεγονότα με το ίδιο αναγνωριστικό αντικειμένου και τα ορίσματά τους να είναι slot με τις ίδιες ιδιότητες αλλά διαφορετικό γνώρισμα προκαλείται η πολλαπλή ερμηνεία της ιδιότητα του ορίσματος του slot. Αυτό συμβαίνει αν τα ορίσματα είναι ανεξάρτητα του κατηγορήματος. Δηλαδή τα παρακάτω framepoint γεγονότα προκαλούν σύγχυση όσον αφορά την ιδιότητα ’item’:

```
transaction#purchase(buyer->Vag seller->Teo item->Hornet)
transaction#liability(responsible->Vag item->moto)
```

Λύση σε αυτό το ζήτημα έρχεται να δώσει η έννοια της προοπτική (perspectivity), η οποία μετατρέπει τα ορίσματα του γεγονότος σε εξαρτώμενα από το κατηγόρημα, ξεκαθαρίζοντας με αυτόν τον τρόπο σε ποιο κατηγόρημα αναφέρεται κάθε φορά. Είναι προφανές ότι ένα framepoint γεγονός δεν έχει προοπτική, μιας και τα ορίσματά του είναι ανεξάρτητα από το κατηγόρημα. Σε περίπτωση όμως που χρειάζεται, πρέπει να μετατραπεί σε pairpoint γεγονός. Δηλαδή το παραπάνω παράδειγμα θα γινόταν:

```
transaction#purchase(buyer+>Vag seller+>Teo item+>Hornet)
transaction#liability(responsible+>Vag item+>moto
```

Ένα γεγονός που τα ορίσματά του είναι τύπου slots μπορεί να συνταχθεί με τη χρήση του συνδετικού 'And()' και να παρέχει τις ίδιες πληροφορίες. Ένα τέτοιο γεγονός θα είχε την ακόλουθη μορφή:

```
And( ?o#purchase(buyer+>Vag)
     ?o#purchase(seller+>Teo)
     ?o#purchase(item+>Hornet) )
```

Παρόλο που υπάρχει αυτή η δυνατότητα δε συνίσταται όταν ο ατομικός τύπος αποτελείται από ένα γεγονός. Εάν όμως ο ατομικός τύπος αποτελείται από τουλάχιστον δύο γεγονότα η χρήση του συνδετικού καθίσταται αναγκαία. Για παράδειγμα, αν ένας ατομικός τύπος περιέχει ένα framepoint γεγονός που παρουσιάζει μία αγορά (purchase), όπως ένα από τα παραπάνω παραδείγματα, και ένα relationpoint γεγονός που έχει ως κατηγόρημα την ευθύνη (liability) και ορίσματα το 'Vag' και 'Hornet' η συνύπαρξη τους σε έναν ατομικό τύπο πραγματοποιείται με τη χρήση του συνδετικού 'And()'. Ένας τέτοιος ατομικός τύπος θα είναι ως εξής:

```
And( transaction#purchase(buyer->Vag seller->Teo item->Hornet)
     transaction#lease(Vag Hornet)
    )
```

Γενικά, ένας ατομικός τύπος μπορεί να έχει ενσωματωμένους και άλλους ατομικούς τύπους. Έστω το γεγονός τύπου frameship:

```
store(owner->Teo location->Meteora)
```

Αυτό το frameship μπορεί να περιέχεται σε έναν ατομικό τύπο που πέρα από το frameship να είναι τύπου relationship. Αυτός ο ατομικός τύπος θα έχει την εξής μορφή:

```
purchase(
    Vag
    store(owner->Teo
          location->Meteora)
    Hornet)
```

Εννοείται ότι οι ατομικοί τύποι θα μπορούσαν να είναι οποιοδήποτε είδος γεγονότος.

Κανόνες (Rules)

Οι κανόνες της PSOA RuleML αποτελούνται από δύο μέρη. Το πρώτο μέρος είναι η συνθήκη (condition) που είναι ένας ατομικός τύπος ή μία σύνδεση ατομικών τύπων. Το δεύτερος μέρος είναι το συμπέρασμα (conclusion) που είναι ένας απλός ατομικός τύπος.

Η γενική εικόνα της σύνταξης ενός κανόνα είναι συμπέρασμα:-συνθήκη, όπου με το σύμβολο ":-" δηλώνεται το "Αν (If)". Ένα παράδειγμα κανόνα το οποίο σε φυσική γλώσσα μεταφράζεται σαν "Ο Vag έχει την ευθύνη για την Hornet, αν ο Vag αγοράσει από τον Teo την Hornet" έχει ως συνθήκη ένα relationship γεγονός με κατηγόρημα μία αγορά και ως συμπέρασμα πάλι ένα relationship γεγονός αλλά με κατηγόρημα μία ευθύνη. Αυτό το παράδειγμα σε PS έχει τη μορφή:

```
liability(Vag Hornet) :-
  purchase(Vag Teo Hornet)
```

Προφανώς οι συνθήκες και τα συμπεράσματα σε έναν κανόνα μπορεί να είναι κάθε είδους γεγονός. Ενδεικτικά, αν το παράδειγμα έχει σαν συνθήκη ένα γεγονός τύπου pairpoint και σαν συμπέρασμα τύπο framepoint θα ήταν έτσι:

```
r#liability(responsible->Vag item->Hornet) :-
  t#purchase(buyer+>Vag seller+>Teo item+>Hornet)
```

Ένας κανόνας μπορεί να μην περιέχει καμία μεταβλητή, όπως τα προηγούμενα παραδείγματα. Ένας τέτοιος κανόνας ονομάζεται βασικός κανόνας (ground rule). Υπάρχει όμως και κανόνας που ο κώδικας του περιλαμβάνει μεταβλητές.

Ένας κανόνας με μεταβλητές χρησιμοποιεί την εντολή 'Forall()' για να εισαγάγει στον κώδικα τις μεταβλητές. Στα προηγούμενα παραδείγματα μπορούν να αντικατασταθούν οι σταθερές `'Vag'`, `'Teo'` και `'Hornet'` από τις αντίστοιχες μεταβλητές `'?V'`, `'?T'` και `'?H'` και να προκύψει το ακόλουθο παράδειγμα:

```
Forall ?V ?T ?H (
    liability(?V ?H) :-
        purchase(?V ?T ?H)
)
```

Και σε αυτό το είδος κανόνα, αυτού δηλαδή που περιέχονται μεταβλητές, θα μπορούσε να

αποτελείται από οποιοδήποτε γεγονός, αρκεί βέβαια να πληρείται η προϋπόθεση της σωστής δόμησής του. Ένα τέτοιο παράδειγμα είναι το παρακάτω:

```
Forall ?r ?t ?V ?T ?H (
    ?r#liability(responsible->?V item->?H) :-
        ?t#purchase(buyer+>?V seller+>?T item+>?H)
```

Στις συνθήκες των κανόνων υπάρχει η έννοια της άρνησης, δηλαδή το "αν δεν". Στη Σύνταξη Παρουσίασης συντάσσεται με την εντολή 'Naf' και η γενική μορφή είναι συμπέρασμα :-Naf(συνθήκη). Ένα παράδειγμα αντίστοιχο με τα προηγούμενα, που σε φυσική γλώσσα μεταφράζεται σαν "Ο Teo είναι υπεύθυνος για τη Hornet αν δεν αγοράσει ο Vag την Hornet από τον Teo", στην PSOA RuleML σε μορφή PS γράφεται ως:

```
liability(Teo Hornet) :-
  Naf(purchase(Vag Teo Hornet))
```

Εννοείται ότι όπως και στους υπόλοιπους κανόνες, βασικούς ή μη, έτσι και όταν υπάρχει η άρνηση στον κώδικα ο ατομικός τύπος μπορεί να περιέχει κάθε είδους γεγονός.

Ισότητα (Equal) και Υποκλάση (Subclass)

Η ισότητα (equal) και η υποκλάση (subclass) είναι δύο σχέσεις που μπορεί να περιγράφονται μέσα σε έναν ατομικό τύπο.

Η ισότητα παρουσιάζει την ισοδυναμία μεταξύ δύο ορισμάτων και συντάσσεται με τα δύο ορίσματα να χωρίζονται με τον σύμβολο του ίσον, "=", δηλαδή η γενική εικόνα είναι "όρισμα=όρισμα".

Η υποκλάση δηλώνει αν ένα όρισμα είναι υποκλάση ενός άλλου ορίσματος και συντάσσεται με δύο σύμβολα της δίεσης, "##", ενωμένα ανάμεσα στα δύο ορίσματα. Το όρισμα στα δεξιά είναι η υποκλάση της κλάσης στα αριστερά, δηλαδή "όρισμα(υποκλάση)##όρισμα(κλάση)".

2.2 Η γραμματική της Σύνταξης Παρουσίασης της Psoa RuleML

Η γραμματική της Σύνταξης Παρουσίασης της Psoa RuleML είναι σε μορφή EBNF (Extended Backus–Naur Form).

Η μορφή EBNF είναι η εκτεταμένη μορφή της BNF (Backus–Naur Form) για την πραγματοποίηση της απλούστευσής της, μέσω κάποιων πρόσθετων συμβολισμών.

Μία γραμματική σε μορφή BNF αποτελείται από:

- ένα σύνολο από λεκτικές μονάδες (tokens), δηλαδή συμβολοσειρές που αποτελούν τα μικρότερα αδιαίρετα κομμάτια της σύνταξης του προγράμματος
- ένα σύνολο από μη τερματικά σύμβολα (non-terminals)

- Συμβολοσειρές που εγκλείονται σε αγκύλες, π.χ. <RuleML>, και αντιπροσωπεύουν κομμάτια του συντακτικού της γλώσσας

- το αρχικό σύμβολο (start symbol) της γραμματικής, ένα συγκεκριμένο μη τερματικό σύμβολο που αποτελεί τη ρίζα του συντακτικού δένδρου

- ένα σύνολο από κανόνες παραγωγής (production rules).

Οι κανόνες παραγωγής χρησιμοποιούνται για την κατασκευή του συντακτικού δένδρου. Κάθε κανόνας έχει τη μορφή A::=B. Το αριστερό μέρος A αποτελείται από ένα μη τερματικό σύμβολο ενώ το δεξί μέρος B είναι μια ακολουθία από τερματικά (λεκτικές μονάδες) και μη τερματικά σύμβολα. Κάθε κανόνας προσδιορίζει έναν πιθανό τρόπο κατασκευής του συντακτικού δένδρου που [5]:

- έχει ως ρίζα του το μη τερματικό σύμβολο στο αριστερό μέρος A του κανόνα

- ως παιδιά αυτής της ρίζας (με την ίδια σειρά εμφάνισης) έχει τα σύμβολα στο δεξί μέρος B του κανόνα

Η EBNF της PSOA RuleML

Η μορφή EBNF της Σύνταξης Παρουσίασης έχει χωριστεί σε δύο μέρη, Rule Language και Condition Language.

- Η Rule Language περιλαμβάνονται τα μη τερματικά σύμβολα.

- Η Condition Language αποτελείται από τα τερματικά σύμβολα και τη σχέση που μπορεί να έχουν μεταξύ τους.

Rule Language [6]:

```
RuleML ::= 'RuleML' '(' Base? Prefix* Import* (Assert | Query)* ')'
Base ::= 'Base' '(' ANGLEBRACKIRI ')'
Prefix ::= 'Prefix' '(' Name ANGLEBRACKIRI ')'
Import ::= 'Import' '(' ANGLEBRACKIRI PROFILE? ')'
Assert ::= 'Assert' '(' (RULE | Assert)* ')'
Query ::= 'Query' '(' FORMULA ')'
RULE ::= ('Forall' Var+ '(' CLAUSE ')') | CLAUSE
CLAUSE ::= Implies | HEAD
Implies ::= HEAD ':-' FORMULA
HEAD ::= ATOMIC | 'Exists' Var+ '(' HEAD ')' | 'And' '(' HEAD* ')'
PROFILE ::= ANGLEBRACKIRI
```

Condition Language [6]:

```
FORMULA ::= 'And' '(' FORMULA* ')' |
            'Or' '(' FORMULA* ')' |
            'Exists' Var+ '(' FORMULA ')' |
            ATOMIC |
            'External' '(' Atom ')' |
            'Naf' '(' FORMULA ')'
ATOMIC ::= Atom | Equal | Subclass
Atom ::= ATOMOIDLESS | ATOMOIDFUL
Equal ::= TERM '=' TERM
Subclass ::= TERM '##' TERM
PSOA ::= Atom | Expr
ATOMOIDLESS ::= ATOMOIDLESSSHORT | ATOMOIDLESSLONG
ATOMOIDLESSSHORT ::= TERM '(' (TERM+ | TUPLEDI*) SLOTDI* ')'
ATOMOIDLESSLONG ::= '#' ATOMOIDLESSSHORT
ATOMOIDFUL ::= TERM '#' ATOMOIDLESSSHORT
Expr ::= EXPRSHORT | EXPRLONG
EXPRSHORT ::= TERM '(' (TERM+ | TUPLED*) SLOTD* ')'
EXPRLONG ::= '^' EXPRSHORT
TUPLEDI ::= ('+' | '-') '[' TERM* ']'
SLOTDI ::= TERM ('+>' | '->') TERM
TUPLED ::= '+' '[' TERM* ']'
SLOTD ::= TERM '+>' TERM
TERM ::= Const | Var | Atom | Expr | 'External' '(' Expr ')'
Const ::= '"' UNICODESTRING '"^^' SYMSPACE | CONSTSHORT
Var ::= '?' PN_LOCAL?
SYMSPACE ::= ANGLEBRACKIRI | CURIE
CONSTSHORT ::= ANGLEBRACKIRI | CURIE | '"' UNICODESTRING '"'
                             | NumericLiteral | '_' PN_LOCAL?
```

2.3 Παραδείγματα της Σύνταξης Παρουσίασης

Η ενότητα αυτή περιλαμβάνει κάποια ολοκληρωμένα παραδείγματα και τις ερμηνείες τους, με σκοπό την καλύτερη κατανόηση της Σύνταξης Παρουσίασης.

Παράδειγμα 1

Έστω το γνωστό παράδειγμα από την παρουσίαση της Σύνταξης Παρουσίασης που περιέχει έναν βασικό κανόνα που αφορά μία ευθύνη (liability) και προκύπτει από μία αγορά (purchase). Σε φυσική γλώσσα το παράδειγμα μεταφράζεται ως εξής "Την ευθύνη για τη Hornet την έχει ο Vag, αν ο αγοραστής Vag αγοράσει από τον πωλητή Teo το αντικείμενο Hornet". Η συνθήκη (condition) είναι ένα framepoint γεγονός και το συμπέρασμα (conclusion) ένα relationship γεγονός.

```
RuleML (
   Assert (
      transaction#purchase(buyer->Vag seller->Teo item->Hornet) :-
         liability(Vag Hornet)
      )
   )
)
```

Στην PSOA RuleML το ριζικό στοιχείο είναι το 'RuleML()', όπως φαίνεται στην γραμμή 1, και το σύνολο του κώδικα σε Σύνταξη Παρουσίαση περιλαμβάνεται στις παρενθέσεις του. Η γραμμή 2 δηλώνει την εισαγωγή των στοιχείων και πρακτικά μετά από την εντολή 'Assert()' ξεκινάει το βασικό μέρος του κώδικα, όπου περιέχονται τα γεγονότα. Από τη γραμμή 3 έως τη γραμμή 5 είναι ο βασικός κανόνας που αφορά ο συγκεκριμένος κώδικας. Αναλυτικά η σύνταξη του είναι στο υποκεφάλαιο 2.1. Το κλείσιμο του κώδικα δηλώνεται με τις κλειστές παρενθέσεις που εκκρεμούν.

Παράδειγμα 2

Το παράδειγμα 2 αναπαριστά τη γνώση ότι η μεταβλητή 'semfe' είναι υποκλάση (subclass) της σταθεράς 'ntua' και ότι η μεταβλητή 'Hudson' ισούται (equal) με '51'.

```
RuleML (
  Assert (
   ?semfe##ntua
   ?Hudson = 51
  )
)
```

Όπως όλα τα αρχεία PSOA RuleML στη μορφή της Σύνταξης Παρουσίασης έτσι και αυτό έχει ως ριζικό στοιχείο το 'RuleML()', και ξεκινάει ο κώδικας με αυτό στη γραμμή 1. Στη γραμμή 2 με το μη τερματικό σύμβολο 'Assert()', επισημαίνεται η εισαγωγή στοιχείων στον κώδικα. Στις γραμμές 3 και 4 βρίσκονται οι σχέσεις των ορισμάτων που αφορούν αυτό το αρχείο της PSOA RuleML. Είναι μία υποκλάση και μία ισότητα με τα ορίσματα που έχουν ήδη αναφερθεί. Τέλος, οι παρενθέσεις δείχνουν το πέρας του κώδικα.

ΚΕΦΑΛΑΙΟ 3

ΣΥΝΤΑΞΗ ΣΕΙΡΙΟΠΟΙΗΣΗΣ ΤΗΣ PSOA RULEML

Σε αυτό το κεφάλαιο γίνεται παρουσίαση της γλώσσας XML και της μορφής Σύνταξης Σειριοποίησης της PSOA RuleML, που είναι σε μορφή XML.

3.1 XML

Η γλώσσα XML (eXtensible Markup Language) είναι μία επεκτάσιμη γλώσσα σήμανσης. Είναι ένα εργαλείο ανεξάρτητο από λογισμικό (software) και υλικό (hardware) για την αποθήκευση και τη μεταφορά δεδομένων. Σχεδιάστηκε για ανάγνωση τόσο από τον άνθρωπο όσο και από τους υπολογιστές. Παίζει σημαντικό ρόλο σε πολλά και διαφορετικά συστήματα πληροφορικής (IT systems) και χρησιμοποιείται συχνά για τη διανομή δεδομένων μέσω του διαδικτύου.

Έχει σχεδιαστεί με τρόπο τέτοιο ώστε να είναι αυτόνομη. Πιο συγκεκριμένοι αποτελείται από:

- πληροφορίες αποστολέα (sender information)
- πληροφορίες παραλήπτη (receiver information)
- μια επικεφαλίδα (a heading)
- ένα σώμα μηνυμάτων (a message body)

Παρόλο που οι πληροφορίες στην XML περιβάλλονται μέσα σε ετικέτες, οι ετικέτες δεν είναι προκαθορισμένες. Με την XML, ο χρήστης πρέπει να ορίσει τις ετικέτες, τις πληροφορίες που θα περιέχουν οι ετικέτες και τη δομή του εγγράφου. Δηλαδή πρέπει να γράψει έναν κώδικα για αποστολή, λήψη, αποθήκευση ή προβολή. Για παράδειγμα, ο παρακάτω κώδικας:

```
<note>
    <to>Vag</to>
    <from>Teo</from>
    <heading>Reminder</heading>
    <body>Learn Java</body>
</note>
```

έχει ως ετικέτες τα '<note>...</note>', '<to>...</to>', '<from>...</from>', '<heading>...</heading>', '<body>...</body>' και για πληροφορίες τα 'Vag', 'Teo', 'Reminder', 'Learn Java'.

Το γεγονός ότι η XML είναι επεκτάσιμη γλώσσα, σημαίνει ότι οι περισσότεροι κώδικές της θα λειτουργούν όπως αναμενόταν, ακόμη και αν προστεθούν ή καταργηθούν νέα δεδομένα.

Για παράδειγμα, μια εφαρμογή έχει σχεδιαστεί για να εμφανίζει την αρχική έκδοση του 'note.xml' ('<to>', '<from>', '<heading>', '<body>'). Στη συνέχεια, δημιουργείται μια νεότερη έκδοση του 'note.xml' με το στοιχείο '<date>' να αντικαθιστά την ετικέτα '<heading>'. Παρά την ανανέωση της εφαρμογής, η παλαιότερη έκδοση κατασκευής της XML θα εξακολουθεί να λειτουργεί και το αποτέλεσμα θα είναι ως εξής:

Old Version	New Version
Note	Note
To: Vag	To: Vag
From: Teo	From: Teo
Reminder	Date: 25/12/2021
Learn Java	Learn Java

Γενικά η XML απλοποιεί τα πράγματα. Πιο συγκεκριμένα δίνει τη δυνατότητα να γίνει:

- κοινή χρήση δεδομένων (data sharing)
- μεταφορά δεδομένων (data transport)
- αλλαγές πλατφόρμας (platform changes)
- διαθεσιμότητα δεδομένων (data availability)

[7]

3.1.1 Κανόνες Σύνταξης

Οι κανόνες σύνταξης της XML (XML Syntax Rules) είναι απλοί, εύκολοι για εκμάθηση και εύχρηστοι. Τα έγγραφα XML πρέπει να περιέχουν ένα ριζικό στοιχείο (root), που είναι το αρχικό στοιχείο (parent) όλων των άλλων στοιχείων. Για παράδειγμα στον κώδικα:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Vag</to>
  <from>Teo</from>
  <heading>Reminder</heading>
  <body>Lern Java</body>
</note>
```

το '<note>' είναι το ριζικό (root) στοιχείο.

Η γραμμή <?xml version="1.0" encoding="UTF-8"?> λέγεται πρόλογος στην XML (XML prolog) και δηλώνει την έκδοση και την κωδικοποίηση των χαρακτήρων του κώδικα. Τα έγγραφα XML μπορούν να περιέχουν διεθνείς χαρακτήρες, όπως Νορβηγικά øæå ή Γαλλικά êèé κτλ. Για την αποφυγή σφαλμάτων, πρέπει να έχει καθοριστεί η κωδικοποίηση που χρησιμοποιείται ή να αποθηκευτούν τα αρχεία XML ως UTF-8 (το UTF-8 είναι η προεπιλεγμένη κωδικοποίηση χαρακτήρων για έγγραφα XML). Ο πρόλογος στην XML είναι προαιρετικός. Εάν υπάρχει, πρέπει να είναι πάνω πάνω στον κώδικα.

Σε έναν XML κώδικα, δεν επιτρέπεται να παραλειφθεί η ετικέτα κλεισίματος. Όλα τα στοιχεία πρέπει να έχουν ετικέτα κλεισίματος, εκτός από τον πρόλογο (prolog XML) γιατί δεν αποτελεί μέρος του κώδικα XML. Για παράδειγμα:
This is a paragraph.

Οι ετικέτες στην XML είναι αρκετά "ευαίσθητες". Για παράδειγμα, η ετικέτα <Letter> διαφέρει από την ετικέτα <letter>. Για αυτό οι ετικέτες ανοίγματος και κλεισίματος πρέπει να γράφονται με τον ίδιο ακριβώς τρόπο. Οι ετικέτες ανοίγματος και κλεισίματος αναφέρονται συχνά ως "ετικέτες έναρξης και λήξης", που είναι ακριβώς το ίδιο, για αυτό μπορεί να γίνει χρήση οποιουδήποτε από τους δύο όρους.

Στην XML όλα τα στοιχεία πρέπει να τοποθετούνται σωστά μεταξύ τους. Αν για παράδειγμα ξεκινήσει μία ετικέτα μέσα σε μία άλλη, τότε θα πρέπει και να κλείνει μέσα σε αυτή. Για παράδειγμα:
.....
αφού το στοιχείο '' ανοίγει μέσα στο στοιχείο '', πρέπει και να κλείνει μέσα στο στοιχείο '', πριν δηλαδή από την ετικέτα κλεισίματος ''.

Οι ετικέτες στην XML μπορούν να περιέχουν και κάποια χαρακτηριστικά πέρα από το όνομα. Δηλώνονται στην ετικέτα έναρξης και δεν αναφέρονται στην ετικέτα λήξης. Π.χ.:

```
<note date="25/12/2021">
  <to>Santa</to>
  <from>Vag</from>
</note>
```

Ορισμένοι χαρακτήρες έχουν ιδιαίτερη σημασία για την XML. Εάν για παράδειγμα τοποθετήσουμε έναν χαρακτήρα όπως το μικρότερο "<" μέσα σε ένα στοιχείο XML, θα δημιουργηθεί σφάλμα επειδή το ερμηνεύει ως την αρχή μιας νέας ετικέτας. Για να αποφύγουμε αυτό το σφάλμα, αντικαθιστούμε τον χαρακτήρα μικρότερο "<" με μια αναφορά οντότητας (entity reference). Υπάρχουν ακόμα τέσσερις χαρακτήρες που θα μπορούσαν να προκαλέ-

σουν ανάλογα σφάλματα. Στον παρακάτω πίνακα φαίνονται και οι πέντε χαρακτήρες και με τι αντικαθιστώνται ώστε να αποφευχθούν σφάλματα:

ολογράφως	**σύμβολο**	**αναφορά οντότητας**
μικρότερο (less than)	<	<
μεγαλύτερο (greater than)	>	>
και (ampersand)	&	&
απόστροφος (apostrophe)	’	'
εισαγωγικά (quotation mark)	”	"

Γενικά, μόνο το σύμβολο του μικρότερου ”<” και του και ”&” απαγορεύονται αυστηρά στη XML, αλλά για τη αποφυγή τυχών σφαλμάτων καλό είναι να γίνεται αντικατάσταση σε όλα τα σύμβολα του πίνακα.

Η σύνταξη σχολίων στη XML είναι ως εξής:
<!– Αυτό είναι ένα σχόλιο –>.
Δεν επιτρέπονται δύο παύλες στη μέση ενός σχολίου.

Το κενό διατηρείται στην XML και δεν περικόπτει τα πολλαπλά κενά. [8]

3.1.2 Στοιχεία

Ένα έγγραφο XML περιέχει στοιχεία (XML Elements). Τα στοιχεία στη XML είναι τα πάντα, από την αρχική ετικέτα του στοιχείου (element’s start tag) έως της τελικής ετικέτας του στοιχείου (element’s end tag). Ένα στοιχείο μπορεί να περιέχει:

- κείμενο (text)
- γνωρίσματα (attributes)
- άλλα στοιχεία (other elements)
- ένα μείγμα των παραπάνω (a mix of the above)

Στο παράδειγμα:

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

οι ετικέτες '<title>', '<author>', '<year>' και '<price>' έχουν περιεχόμενο κειμένου (text content) επειδή περιέχουν κείμενο (όπως '29,99'), ενώ οι ετικέτες '<bookstore>' και '<book>' έχουν περιεχόμενο στοιχείων (element contents), επειδή περιέχουν άλλα στοιχεία. Το '<book>' περιέχει χαρακτηριστικά (όπως 'category ="children"').

Κενά στοιχεία

Ένα στοιχείο χωρίς περιεχόμενο λέγεται κενό (Empty XML Elements). Τα κενά στοιχεία μπορούν να έχουν χαρακτηριστικά στην ετικέτα τους. Στην XML, η υπόδειξη για ένα κενό στοιχείο γίνεται με αυτό τον τρόπο:
<element></element>,
ή με μια λεγόμενη αυτο-κλειστή ετικέτα (self-closing tag):
<element />.
Και οι δύο τρόποι παράγουν το ίδιο αποτέλεσμα.

Κανόνες ονομάτων

Τα στοιχεία XML πρέπει να ακολουθούν τους παρακάτω κανόνες ονομασίας (XML Naming Rules). Τα ονόματα των στοιχείων πρέπει :

- να είναι πεζά
- να ξεκινούν με ένα γράμμα ή κάτω παύλα
- να μην ξεκινούν με τα γράμματα xml (ή XML ή Xml)
- μπορούν να περιέχουν γράμματα, ψηφία, ενωτικά, κάτω παύλες και τελείες
- να μην περιέχουν κενά, εκτός αν υπάρχει χαρακτηριστικό που διαχωρίζεται από το όνομα με κενό

Μπορεί να χρησιμοποιηθεί οποιοδήποτε όνομα, χωρίς να δεσμεύονται λέξεις, εκτός από ”xml”.

Δεν έχει οριστεί κάποιος συγκεκριμένος τρόπος παρουσίασης για τα ονόματα των στοιχείων στη XML. Εδώ είναι μερικά παραδείγματα που χρησιμοποιούνται συνήθως:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

Καλό είναι ένας κώδικας XML να έχει τον ίδιο τρόπο παρουσίασης των ονομάτων από την αρχή μέχρι το τέλος του.

Τα στοιχεία XML είναι επεκτάσιμα. Αυτό σημαίνει ότι ο χρήσης μπορεί να τα τροποποιήσει όποτε το επιθυμεί. [9]

3.1.3 Χαρακτηριστικά

Τα στοιχεία της XML μπορούν να περιέχουν χαρακτηριστικά (attributes). Τα χαρακτηριστικά έχουν σχεδιαστεί για να περιέχουν δεδομένα (data) που σχετίζονται με ένα συγκεκριμένο στοιχείο. Οι τιμές των χαρακτηριστικών πρέπει πάντα να αναφέρονται. Μπορούν να χρησιμοποιηθούν είτε με απλά είτε με διπλά εισαγωγικά.

Έστω το παράδειγμα που αφορά το φύλο ενός ατόμου. Το στοιχείο ’<person>’ μπορεί να γραφεί ως εξής:
<person gender=”female”> ή <person gender=’female’>
Εάν η ίδια η τιμή του χαρακτηριστικού περιέχει διπλά εισαγωγικά, μπορεί να χρησιμοποιήσει απλά εισαγωγικά, όπως σε αυτό το παράδειγμα:
<gangster name=’Vito ”Godfather” Corleone’> ή όπως θα συντασσόταν πιο ασφαλές σε XML
<gangster name=”Vito "Godfather" Corleone”>.

Ορισμένα πράγματα που πρέπει να ληφθούν υπόψιν κατά τη χρήση χαρακτηριστικών είναι τα εξής:

- τα χαρακτηριστικά δεν μπορούν να περιέχουν πολλαπλές τιμές (τα στοιχεία μπορούν)
- τα χαρακτηριστικά δεν μπορούν να περιέχουν δομές δέντρων (τα στοιχεία μπορούν)
- τα χαρακτηριστικά δεν είναι εύκολα επεκτάσιμα (για μελλοντικές αλλαγές)

3.1.4 Στοιχεία και Χαρακτηριστικά

Δεν υπάρχουν κανόνες σχετικά με το πότε να χρησιμοποιούνται τα χαρακτηριστικά ή πότε να χρησιμοποιούνται τα στοιχεία σε XML.

Στο παράδειγμα

```
<person gender="female">
  <firstname>Sissy</firstname>
  <lastname>Folina</lastname>
</person>
```

το φύλο (gender) είναι χαρακτηριστικό, ενώ στο παράδειγμα

```
<person>
  <gender>female</gender>
  <firstname>Sissy</firstname>
  <lastname>Folina</lastname>
</person>
```

είναι στοιχείο (element). Και τα δύο παραδείγματα παρέχουν τις ίδιες πληροφορίες.

Τα ακόλουθα τρία έγγραφα XML παρέχουν ακριβώς τις ίδιες πληροφορίες:
Ένα χαρακτηριστικό που δηλώνει μία ημερομηνία χρησιμοποιείται στο πρώτο παράδειγμα:

```
<note date="2020-12-25">
  <to>Santa</to>
  <from>Vag</from>
</note>
```

Στο δεύτερο παράδειγμα χρησιμοποιείται ένα στοιχείο για να δηλώσει μία ημερομηνία:

```
<note>
  <date>2020-12-25</date>
  <to>Santa</to>
  <from>Vag</from>
</note>
```

Στο τρίτο παράδειγμα η ημερομηνία δηλώνεται με ένα αναπτυγμένο στοιχείο <date>:

```
<note>
  <date>
    <year>2020</year>
    <month>12</month>
    <day>25</day>
  </date>
  <to>Santa</to>
  <from>Vag</from>
</note>
```
[10]

3.1.5 Χώροι ονομάτων

Οι χώροι ονομάτων (XML Namespaces) παρέχουν μια μέθοδο για την αποφυγή διενέξεων στα ονόματα των στοιχείων. Επειδή στην XML τα ονόματα των στοιχείων καθορίζονται από τον χρήστη συχνά δημιουργούνται διενέξεις όταν γίνεται προσπάθεια να συνδυαστούν έγγραφα XML από διαφορετικές εφαρμογές.
Για παράδειγμα ο κώδικας:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

μεταφέρει πληροφορίες για ένα <table>. Όμως και ο κώδικας:

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

μεταφέρει πληροφορίες για ένα '<table>' αλλά αναφέρεται σε κάτι διαφορετικό. Εάν αυτά τα τμήματα XML χρησιμοποιηθούν μαζί, θα υπάρξει σύγχυση μεταξύ των ονομάτων. Και τα δύο περιέχουν ένα στοιχείο '<table>', αλλά το κάθε στοιχείο έχει διαφορετικό περιεχόμενο και νόημα.

Οι διενέξεις ονομάτων σε XML μπορούν να αποφευχθούν χρησιμοποιώντας ένα πρόθεμα ονόματος (name prefix).

Δηλαδή τα δύο προηγούμενα παραδείγματα να πάψουν να ταυτίζονται αν μετατραπούν σε:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Στα παραπάνω παραδείγματα, δεν θα υπάρξει σύγχυση επειδή τα δύο στοιχεία '<table>' έχουν διαφορετικά προθέματα ονόματος.

Όταν γίνεται χρήση προθέματος σε XML, πρέπει να οριστεί ένας χώρος ονομάτων για το πρόθεμα. Ο χώρος αυτός μπορεί να είναι ένα χαρακτηριστικό xmlns (xmlns attribute) στην ετικέτα έναρξης ενός στοιχείου. Η δήλωση χώρου ονομάτων έχει την ακόλουθη σύνταξη: xmlns: πρόθεμα = "URI" ή όπως φαίνεται στο παράδειγμα:

```
<root>
    <h:table xmlns:h="http://www.w3.org/TR/html4/">
      <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
      </h:tr>
    </h:table>

    <f:table xmlns:f="https://www.w3schools.com/furniture">
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
    </f:table>
</root>
```

Στο παραπάνω παράδειγμα το χαρακτηριστικό xmlns στο πρώτο στοιχείο <table> δίνει στο πρόθεμα h: ένα κατάλληλο χώρο ονομάτων, ενώ το χαρακτηριστικό xmlns στο δεύτερο στοιχείο <table> ο κατάλληλος χώρος ονομάτων δίνεται από το πρόθεμα f.

Όταν ορίζεται ένας χώρος ονομάτων για ένα στοιχείο, όλα τα θυγατρικά στοιχεία με το ίδιο πρόθεμα συσχετίζονται.

Τα ονόματα χώρων μπορούν επίσης να δηλωθούν στο ριζικό στοιχείο σαν χαρακτηριστικά:

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="https://www.w3schools.com/furniture">
    <h:table>
      <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
      </h:tr>
    </h:table>

    <f:table>
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
    </f:table>
</root>
```

[11]

3.1.6 Απεικόνιση - Εμφάνιση

Τα αρχεία XML μπορούν να προβληθούν σε όλα τα προγράμματα περιήγησης (browsers). Για παράδειγμα το αρχείο XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Santa</to>
    <from>Vag</from>
    <heading>Reminder</heading>
    <body>Don't forget me this Christmas!</body>
</note>
```

σε κάποιο browser εμφανίζεται ως:

```
<note>
    <to>Santa</to>
    <from>Vag</from>
    <heading>Reminder</heading>
    <body>Don't forget me this Christmas!</body>
</note>
```

Τα περισσότερα προγράμματα περιήγησης μπορεί να εμφανίζουν ένα έγγραφο XML με χρωματικά στοιχεία. Συχνά το σύμβολο συν (+) ή μείον (-) στα αριστερά των στοιχείων αναπτύσσουν ή συμπτύσσουν τη δομή του στοιχείου. Για να προβληθεί η ακατέργαστη-αρχική

πηγή XML υπάρχει η επιλογή ”Προβολή πηγής σελίδας” (”View Page Source”) ή ”Προβολή προέλευσης” (”View Source”) από το μενού του προγράμματος περιήγησης.

Εάν ανοιχθεί ένα εσφαλμένο αρχείο XML, ορισμένα προγράμματα περιήγησης θα αναφέρουν το σφάλμα και κάποια θα το εμφανίσουν ή κάποια θα το εμφανίσουν εσφαλμένα.

Τα έγγραφα XML δεν περιέχουν πληροφορίες σχετικά με τον τρόπο εμφάνισης των δεδομένων. Δεδομένου ότι οι ετικέτες XML ”εφευρέθηκαν” από τον χρήστη του εγγράφου, τα προγράμματα περιήγησης δεν γνωρίζουν σε τι αναφέρεται μια ετικέτα. Χωρίς πληροφορίες σχετικά με τον τρόπο εμφάνισης των δεδομένων, τα προγράμματα περιήγησης μπορούν να εμφανίσουν το έγγραφο XML όπως είναι.
[12]

3.2 Η Μορφή της Σύνταξης Σειριοποίησης

Για την αναπαράσταση γνώσης στην PSOA RuleML χρησιμοποιείται ως επί το πλείστον η μορφή της Σύνταξης Παρουσίασης γιατί είναι πιο εύκολη στην ανάγνωση συγκριτικά με τη Σύνταξη Σειριοποίησης. Όμως, για την επικύρωση, τη μετατροπή, την ανταλλαγή και την επαναχρησιμοποίηση δεδομένων και γνώσεων, γενικά για την τροποποίηση κάποιου αρχείου PSOA RuleML, αξιοποιείται η μορφή της Σύνταξης Σειριοποίησης, μιας και στην XML αυτές οι δυνατότητες είναι πιο εύκολα προσαρμόσιμες από τον χρήστη.

Στη μορφή της Σύνταξης Σειριοποίησης μπορεί να αναπαρασταθεί οποιαδήποτε γνώση μπορεί να αναπαρασταθεί και στη Σύνταξη Παρουσίασης, χρησιμοποιώντας τα κατάλληλα στοιχεία. Τα στοιχεία έχουν για όνομα το αριστερό μέρος όπως παρουσιάζεται στη μορφή EBNF.

3.2.1 Παρουσίαση της Σύνταξης Σειριοποίησης

Για την ευκολότερη κατανόηση της σύνταξης ενός αρχείου PSOA RuleML στη μορφή Σύνταξης Σειριοποίησης θα γίνει μία ανάλογη χρήση παραδειγμάτων, όπως και στη Σύνταξη Παρουσίασης.

Ατομικός Τύπος

Κάποια γενικά χαρακτηριστικά της σύνταξης σε XML ενός ατομικού τύπου είναι ότι:

- ξεκινάει με την ετικέτα ”<Atom>” και τελειώνει με την ετικέτα κλεισίματος ”</Atom>”. Ότι υπάρχει εκεί ανάμεσα αφορά τον συγκεκριμένο ατομικό τύπο
- το κατηγόρημα βρίσκεται ανάμεσα από τις ετικέτες ”<Rel>” και ”</Rel>”
- κάθε σταθερά είναι ανάμεσα από τις ετικέτες <Ind> και </Ind>
- κάθε μεταβλητή είναι ανάμεσα από τις ετικέτες <Var> και </Var>

Η μορφή της υλοποίησης ενός απλού ατομικού τύπου με relationship γεγονός, με κατηγόρημα μία αγορά, 'purchase', και για ορίσματά του μία πλειάδα με έναν αγοραστή, 'Vag, έναν πωλητή, 'Teo', και ένα αντικείμενο, 'Hornet' θα είναι:

```
<Atom>
    <Rel>purchase</Rel>
    <tupdep>
        <Tuple>
            <Ind>Vag</Ind>
            <Ind>Teo</Ind>
            <Ind>Hornet</Ind>
        </Tuple>
    </tupdep>
</Atom>
```

Στον παραπάνω ατομικό τύπο μπορεί να μετατραπεί το γεγονός σε shelfship, δηλαδή τα ορίσματα της πλειάδας να είναι ανεξάρτητα από το κατηγόρημα, και να έχει την ακόλουθη μορφή:

```
<Atom>
    <Rel>purchase</Rel>
    <tup>
        <Tuple>
            <Ind>Vag</Ind>
            <Ind>Teo</Ind>
            <Ind>Hornet</Ind>
        </Tuple>
    </tup>
</Atom>
```

η διαφορά τους είναι στην ετικέτα '<tupdep>' και '<tup>' αντίστοιχα. Η σειρά των ορισμάτων παραμένει σημαντική όταν πρόκειται για πλειάδες.

Σε περίπτωση που χρειαστεί να γίνει ξεκάθαρη η ιδιότητα του κάθε ορίσματος μπορεί να γίνει η χρήση slots αντί των πλειάδων. Για το συγκεκριμένο παράδειγμα θα χρησιμοποιηθούν οι ιδιότητες του αγοραστής, 'buyer', του πωλητή, 'seller' και του αντικειμένου, 'item' με τα ίδια εξαρτώμενα ορίσματα που χρησιμοποιήθηκαν στα παραπάνω παραδείγματα και το γεγονός θα γίνει pairship:

```
<Atom>
    <Rel>purchase</Rel>
    <slotdep>
        <Ind>buyer</Ind>
        <Ind>Vag</Ind>
    </slotdep>
    <slotdep>
        <Ind>seller</Ind>
        <Ind>Teo</Ind>
    </slotdep>
    <slotdep>
        <Ind>item</Ind>
        <Ind>Hornet</Ind>
    </slotdep>
</Atom>
```

Η μετατροπή του σε frameship γεγονός πραγματοποιείται με την αντικατάσταση των ετικέτών '<slotdep>' με '<slot>' και βέβαια με την ανάλογη ετικέτα κλεισίματος, '</slotdep>' με '</slot>', δηλώνοντας έτσι ότι την ανεξαρτησία των ορισμάτων από το κατηγόρημα.

Η εισαγωγή ενός αναγνωριστικού αντικειμένου θα επηρέαζε τους παραπάνω ατομικού τύπους αλλάζοντας το είδος των γεγονότων. Για παράδειγμα αν προστεθεί στο pairship γεγονός ένα αναγνωριστικό αντικειμένου, 'transaction', αυτόματα θα γίνει pairpoint και σε μορφή XML θα είναι:

```
<Atom>
    <oid>
        <Ind>transaction</Ind>
    </oid>
        <Rel>purchase</Rel>
    <slotdep>
        <Ind>buyer</Ind>
        <Ind>Vag</Ind>
    </slotdep>
    <slotdep>
        <Ind>seller</Ind>
        <Ind>Teo</Ind>
    </slotdep>
    <slotdep>
        <Ind>item</Ind>
        <Ind>Hornet</Ind>
    </slotdep>
</Atom>
```

Αντίστοιχα, αν προστεθεί αναγνωριστικό αντικειμένου σε ένα γεγονός θα επέλθουν οι

αναμενόμενες αλλαγές στο είδος του. Στη μορφή της XML η θέση του αναγνωριστικού αντικειμένου είναι, όπως και στη Σύνταξη Παρουσίασης, πριν από το κατηγόρημα με τις ετικέτες όπως φαίνεται στο παραπάνω παράδειγμα. Αντίστοιχη θέση έχει σε κάθε γεγονός, αν φυσικά περιέχει ένα.

Όσον αφορά τη σύζευξη ατομικών τύπων, με το συνδετικό ”And” στην XML πραγματοποιείται με την εξής σύνταξη:

```
<And>
    <Atom>
      <oid>
        <Ind>transaction</Ind>
      </oid>
      <Rel>purchase</Rel>
      <slot>
        <Ind>buyer</Ind>
        <Ind>Vag</Ind>
      </slot>
      <slot>
        <Ind>seller</Ind>
        <Var>Teo</Var>
      </slot>
      <slot>
        <Ind>item</Ind>
        <Ind>Hornet</Ind>
      </slot>
    </Atom>
    <Atom>
      <Rel>purchase</Rel>
      <tupdep>
        <Tuple>
          <Ind>Vag</Ind>
          <Ind>Teo</Ind>
          <Ind>Hornet</Ind>
        </Tuple>
      </tupdep>
    </Atom>
</And>
```

Στο παραπάνω παράδειγμα χρησιμοποιήθηκαν τυχαία ένας ατομικός τύπος με framepoint γεγονός και ένας με relationship. Η σύζευξη προφανώς μπορεί να αποτελείται απο κάθε είδους γεγονότος.

Κανόνες

Η λογική των κανόνων είναι η ίδια με αυτή που παρουσιάστηκε στη μορφή Σύνταξης Παρουσίασης. Πρόκειται δηλαδή για μία συνθήκη και το συμπέρασμά της. Χωρίζονται σε βασικούς κανόνες, αυτούς δηλαδή που δεν περιέχουν μεταβλητές και σε μη βασικούς κανόνες όταν περιέχουν.

Η εισαγωγή μεταβλητών γίνεται με την εντολή ”Forall” και έχει την ακόλουθη ετικέτα ’<Forall>’. Μετά από αυτήν την ετικέτα δηλώνονται όλες οι μεταβλητές που θα χρησιμοποιηθούν στον κώδικα. Αφού τελειώσει αυτό το κομμάτι ξεκινάει η σύνταξη του κανόνα.

Η σύνταξη ενός κανόνα σε γενική μορφή είναι η ακόλουθη:

```
<Implies>
    <If>
        <Atom>
            ...
        </Atom>
    </If>
    <then>
        <Atom>
            ...
        </Atom>
    </then>
</Implies>
```

Αν υπάρχουν μεταβλητές όλο αυτό το κομμάτι κώδικα βρίσκεται ανάμεσα από τις ετικέτες ’<Forall>’ και ’</Forall>’. Εννοείται ότι ανάμεσα από το ’<Atom>’...
’</Atom>’ είναι το είδος του ατομικού τύπου ή μια σύζευξη.

Η άρνηση της συνθήκης στην XML υλοποιείται αν συμπεριληφθεί το κομμάτι της μέσα στις ετικέτες ’<Naf>’ και ’</Naf>’.

Ένα παράδειγμα μη βασικού κανόνα με άρνηση της συνθήκης είναι ο ακόλουθος:

```
<Forall>
    <Var>b</Var>
    <Var>s</Var>
    <Var>i</Var>
    <Var>t</Var>
    <Implies>
        <Naf>
            <if>
                <Atom>
                    <oid><Var>t</Var></oid>
                    <op><Rel>purchase</Rel></op>
                    <slotdep>
                        <Ind>buyer</Ind><Var>b</Var>
                    </slotdep>
                    <slotdep>
                        <Ind>seller</Ind><Var>s</Var>
                    </slotdep>
                    <slotdep>
                        <Ind>item</Ind><Var>i</Var>
                    </slotdep>
                </Atom>
            </if>
        </Naf>
        <then>
            <Atom>
                <oid>
                    <Var>t</Var>
                </oid>
                <Rel>liability</Rel>
                <slotdep>
                    <Ind>bearer</Ind><Var>b</Var>
                </slotdep>
                <slotdep>
                    <Ind>item</Ind><Var>i</Var>
                </slotdep>
            </Atom>
        </then>
    </Implies>
</Forall>
```

Ισότητα (Equal) και Υποκλάση (Subclass)

Στην XML η ισότητα μεταξύ δύο ορισμάτων δηλώνεται όταν αυτά τα ορίσματα είναι ανάμεσα από τις ετικέτες της ισότητας, δηλαδή '<Equal>'...'</Equal>'.

Η υποκλάση δηλώνεται με παρόμοιο τρόπο όπως η ισότητα, αλλά αντί για τις ετικέτες της ισότητας έχουν τις ετικέτες της υποκλάσης οι οποίες είναι '<Subclass>' και '</Subclass>'. Το πρώτο όρισμα είναι η υποκλάση της κλάσης που είναι το δεύτερο όρισμα. [13]

3.2.2 Παραδείγματα της Σύνταξης Σειριοποίησης

Ακολουθούν δύο ολοκληρωμένα παραδείγματα για να γίνει πιο εύκολα αντιληπτό πως συντάσσεται η μορφή της Σύνταξης Σειριοποίησης της PSOA RuleML. Να φανεί δηλαδή η ετικέτα που χρησιμοποιείται για την έναρξη και το κλείσιμο του κώδικα, αλλά και οι ενδιάμεσες ετικέτες. Προφανώς οι συνδυασμοί είναι ατελείωτοι αλλά η γενική ιδέα μπορεί να γίνει αντιληπτή από τα παρακάνω παραδείγματα.

Παράδειγμα 1

Σε φυσική γλώσσα το πρώτο παράδειγμα μεταφράζεται ως "Την ευθύνη για τη Hornet την έχει ο Vag, αν ο Vag αγοράσει από τον Teo την Hornet". Παρουσιάζει δηλαδή το συμπέρασμα που είναι η ευθύνη (liability), ως ένα relationship γεγονός, και τη συνθήκη που είναι η αγορά (purchase) ως ένα pairpoint γεγονός.

```
1  <RuleML>
2      <Assert>
3          <Implies>
4              <If>
5                  <Atom>
6                      <oid><Ind>transaction</Ind></oid>
7                      <Rel>purchase</Rel>
8                      <slot>
9                          <Ind>buyer</Ind>
10                         <Ind>Vag</Ind>
11                     </slot>
12                     <slot>
13                         <Ind>seller</Ind>
14                         <Var>Teo</Var>
15                     </slot>
16                     <slot>
17                         <Ind>item</Ind>
18                         <Ind>Hornet</Ind>
19                     </slot>
20                 </Atom>
21             </If>
22             <then>
23                 <Atom>
24                     <Rel>liability</Rel>
25                     <tupdep>
26                         <Tuple>
27                             <Ind>Vag</Ind>
28                             <Ind>Hornet</Ind>
29                         </Tuple>
30                     </tupdep>
31                 </Atom>
32             </then>
33         </Implies>
34     </Assert>
35 </RuleML>
```

Ο κώδικας ξεκινάει με το ριζικό στοιχείο που είναι το '<RuleML>, όπως φαίνεται στη γραμμή 1, και στη γραμμή 2 είναι το στοιχείο '<Assert>' που δηλώνει την εισαγωγή των ατομικών τύπων. Από τη γραμμή 3 γίνεται αντιληπτό ότι ο κώδικας αποτελεί ένα βασικό κανόνα, αφού πριν από το '<Implies>' δεν υπάρχει κάποια δήλωση μεταβλητών. Ακολουθεί η σύνταξη του κανόνα από τη γραμμή 4 εώς τη γραμμή 33, όπου και τελειώνει ο κανόνας με το στοιχείο '</Implies>'. Στις δύο τελευταίες γραμμές κλείνουν οι ετικέτες που εκκρεμούν από το "Assert" και το "RuleML".

Παράδειγμα 2

Στο δεύτερο παράδειγμα φαίνεται πως συντάσσεται ο κώδικας XML όταν περιλαμβάνει υποκλάσεις και δηλώνει μία ισότητα. Εδώ το παράδειγμα σε φυσική γλώσσα λέει ότι "η semfe είναι υποκλάσση του ntua και ότι η μεταβλητή Hudson ισούται με 51".

```
1  <RuleML>
2     <Assert>
3        <Subclass>
4           <Ind>semfe</Ind>
5           <Ind>ntua</Ind>
6        </Subclass>
7        <Equal>
8            <Var>Hudson</Var>
9            <Ind>51</Ind>
10       </Equal>
11    </Assert>
12 </RuleML>
```

Ο κώδικας ξεκιάει με τον γνωστό τρόπο, οπως φαίνεται στη γραμμή 1 η εντολή '<RuleML>', και στη συνέχεια δηλώνεται η εισαγωγή ορισμάτων, στη γραμμή 2 '<Assert>'. Η υποκλάση στην XML αρχίζει και τελειώνει με τις ετικέτες '<Subclass>' γραμμή 3 και '</Subclass>' γραμμή 6 αντίστοιχα, και μέσα σε αυτές είναι τα ορίσματα, στην προκειμένη οι σταθερές 'semfe' και 'ntua', γραμμές 4 και 5. Η ισότητα αρχίζει με την ετικέτα '<Equal>', γραμμή 7, και κλείνει με '</Equal>', γραμμή 10. Ανάμεσα από αυτές τις ετικέτες είναι τα ορίσματα που ισούνται, δηλαδή η μεταβλητή 'Hudson' και η σταθερά 51, γραμμές 8 και 9 αντίστοιχα. Στις γραμμές 11 και 12 είναι οι ετικέτες που πρέπει να χρησιμοποιηθούν για να κλείσει ο κώδικας.

ΚΕΦΑΛΑΙΟ 4

ΑΦΗΡΗΜΕΝΑ ΣΥΝΤΑΚΤΙΚΑ ΔΕΝΤΡΑ

Η σύνταξη των γλωσσών προγραμματισμού χρησιμοποιεί γραμματικές. Οι γραμματικές παράγουν τα συντακτικά δέντρα, τα οποία αναπαριστούν το σύνολο των συμβολοσειρών. Άρα τα συντακτικά δέντρα (Syntax Trees) είναι η αναπαράσταση του συνόλου των συμβολοσειρών.

Η γραμματική έχει ένα σύνολο κανόνων που ορίζουν τον τρόπο με τον οποίο θα κατασκευαστεί ένα συντακτικό δέντρο. Ο τρόπος που ορίζει πως προκύπτουν τα συντακτικά δένδρα από μία ακολουθία λεκτικών μονάδων (tokens) δημιουργείται από τον συντακτικό αναλυτή (parser). Δηλαδή στο συντακτικό δέντρο αποτυπώνει το αποτέλεσμα του συντακτικού αναλυτή (parser) και εμφανίζει τα κομμάτια της γλώσσας που δεν μπορούν να θεωρηθούν κατασκευαζόμενα από μικρότερα κομμάτια.

Η κατασκευή ενός συντακτικού δέντρου έχει ως ”ρίζα του δέντρου” το αρχικό σύμβολο της γραμματικής. Εκεί προσθέτονται όλα τα ”φύλλα του δέντρου” που είναι μη τερματικά σύμβολα (non-terminal) και αποτελούν τους εσωτερικούς κόμβους του δέντρου, χρησιμοποιώντας τους κανόνες παραγωγής που ορίζει η κάθε γλώσσα. Η διαδικασία τερματίζει όταν όλα τα ”φύλλα του δένδρου” αποτελούνται από τερματικά σύμβολα (terminal) ή αλλιώς λεκτικές μονάδες (tokens). Η συμβολοσειρά που αντιστοιχεί στο δένδρο βρίσκεται διαβάζοντας τα φύλλα του δένδρου από αριστερά προς τα δεξιά.

Το αφηρημένο συντακτικό δέντρο (Abstract Syntax Tree) είναι μία συμπτυγμένη μορφή του συντακτικού δέντρου από τις υλοποιήσεις των γλωσσών. Κάθε ρεαλιστική γλώσσα περιέχει πολλά μη τερματικά σύμβολα (non-terminal), ειδικά εάν από τη γραμματική έχει εξαλειφθεί η ασάφεια. Τα επιπλέον μη τερματικά σύμβολα καθορίζουν τον τρόπο με τον οποίο προκύπτει ένα μοναδικό συντακτικό δένδρο. Όταν κατασκευαστεί το συντακτικό δένδρο, τα επιπλέον μη τερματικά δεν έχουν κάποια χρησιμότητα πλέον. Για τον λόγο αυτό προέκυψε η συμπτυγμένη μορφή του συντακτικού δέντρου, η αφηρημένη (abstract). [14]

Παρακάτω αποτυπώνεται η εικόνα ενός αφηρημένου συντακτικού δέντρου (Abstract Syntax Tree) στη γλώσσα XML για την καλύτερη κατανόηση της λειτουργίας του. Θα ακολουθήσει το αποτέλεσμα του συντακτικού αναλυτή αν εφαρμοστεί στη PSOA RuleML είτε σε μορφή Σύνταξης Παρουσίασης είτε σε Σύνταξη Σειριοποίησης.

4.1 Συντακτικά Δέντρα της XML

Τα έγγραφα που είναι γραμμένα σε XML σχηματίζουν μια δομή δέντρου που ξεκινά από τη ”ρίζα” (root) και διακλαδίζονται σε ”φύλλα” (leaves).

Ένας άλλο τρόπος για να περιγραφεί η σχέση μεταξύ των στοιχείων σε ένα συντακτικό δέντρο είναι οι όροι ”γονέας (parent)”, ”παιδί (child)” και ”αδελφός (sibling)”. Οι γονείς έχουν παιδιά. Τα παιδιά έχουν γονείς. Τα αδέλφια είναι παιδιά στο ίδιο επίπεδο.

Όλα τα στοιχεία έχουν ή περιεχόμενο κειμένου (text content) ή χαρακτηριστικά (attributes). Περιεχόμενο κειμένου έχουν συνήθως τα μη τερματικά σύμβολα ενώ χαρακτηριστικά τα τερματικά.

Μία γενική εικόνα για τον τρόπο που συντάσσεται ένα δέντρο σε κώδικας XML είναι η εξής:

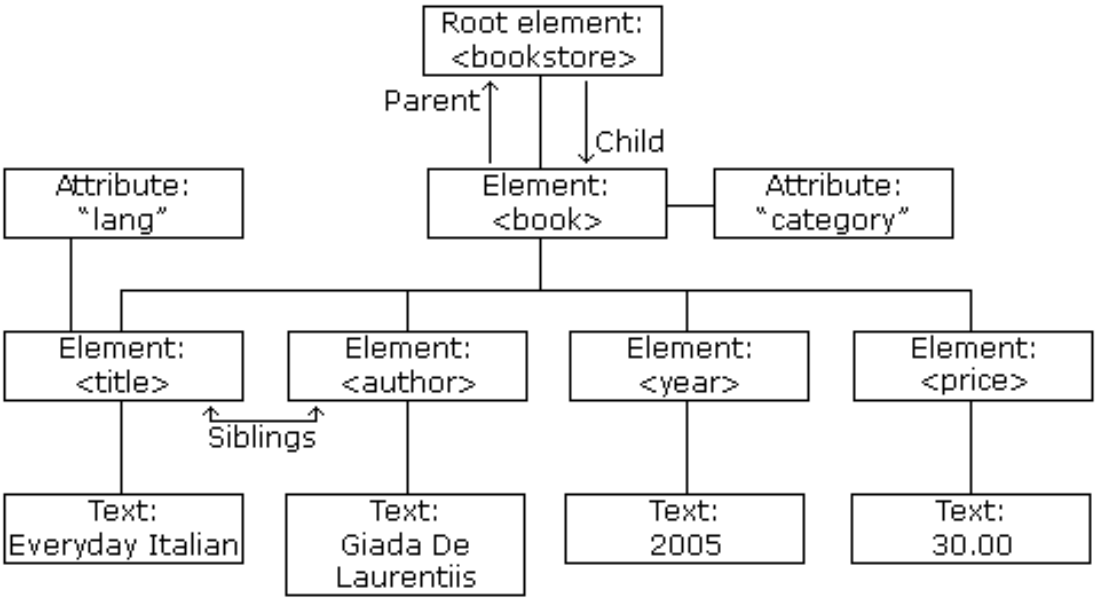
```
<root>
    <child>
        <subchild>.....</subchild>
    </child>
</root>
```

Παράδειγμα

Έστω το έγγραφο XML που περιλαμβάνει πληροφορίες σχετικά με βιβλία.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Το συντακτικό δέντρο που θα παραχθεί αφού δράσει ο συντακτικός αναλυτής θα είναι:



Σχήμα 4.1: Παράδειγμα δομής ενός συντακτικού δέντρου XML [1]

4.2 Συντακτικά Δέντρα της PSOA RuleML

Όπως όλες οι γλώσσες, έτσι και η PSOA RuleML σχηματίζει συντακτικά δέντρα που ξεκινά από τη "ρίζα" (root) και διακλαδίζεται σε "φύλλα" (leaves).

Το αφηρημένο συντακτικό δέντρο (Abstract Syntax Tree) που προκύπτει από τον συντακτικό αναλυτή όταν γίνεται εισαγωγή κειμένου σε γλώσσα PSOA RuleML είναι το ίδιο είτε είναι σε μορφή Σύνταξης Παρουσίασης είτε σε Σύνταξη Σειριοποίησης (για αυτό άλλωστε είναι εφικτή η αμφίδρομη μετατροπή μέσω αφηρημένων συντακτικών δέντρων). Τα στοιχεία που δημιουργούν τη ρίζα και οι κόμβοι των στοιχείων είναι όλα τα μη τερματικά στοιχεία όπως παρουσιάζονται στη μορφή EBNF. Την ανάλογη μορφή EBNF έχουν και τα φύλλα που αποτελούν τα τερματικά στοιχεία.

4.2.1 Παραδείγματα

Ακολουθούν τα συντακτικά δέντρα που προκύπτουν όταν δρα ο συντακτικός αναλυτής στα δύο παραδείγματα που παρουσιάστηκαν στη Σύνταξη Παρουσίασης και στη Σύνταξη Σειριοποίησης.

Παράδειγμα 1

το πρώτο παράδειγμα σε φυσική γλώσσα είναι: "Την ευθύνη για τη Hornet την έχει ο Vag, αν ο Vag αγοράσει από τον Teo την Hornet".

Σε μορφή της Σύνταξης Παρουσίασης υπενθυμίζεται ότι είναι η ακόλουθη:

```
RuleML (
   Assert (
      transaction#purchase(buyer->Vag seller->Teo item->Hornet) :-
         liability(Vag Hornet)
      )
   )
)
```

και σε μορφή της Σύνταξης Σειριοποίησης είναι:

```
<RuleML>
    <Assert>
        <Implies>
            I<f>
                <Atom>
                    <oid><Ind>transaction</Ind></oid>
                    <Rel>purchase</Rel>
                    <slot>
                        <Ind>buyer</Ind>
                        <Ind>Vag</Ind>
                    </slot>
                    <slot>
                        <Ind>seller</Ind>
                        <Var>Teo</Var>
                    </slot>
                    <slot>
                        <Ind>item</Ind>
                        <Ind>Hornet</Ind>
                    </slot>
                </Atom>
            I</f>
            <then>
                <Atom>
                    <Rel>liability</Rel>
                    <tupdep>
                        <Tuple>
                            <Ind>Vag</Ind>
                            <Ind>Hornet</Ind>
                        </Tuple>
                    </tupdep>
                </Atom>
            </then>
        </Implies>
    </Assert>
</RuleML>
```

Το αφηρημένο συντακτικό δέντρο που δημιουργεί αυτό το παράδειγμα είναι:

```
(RuleML
    (Assert
        (:-
            (PSOA (SHORTCONST transaction)
            (# (SHORTCONST purchase))
                (SLOT - (SHORTCONST buyer) (SHORTCONST Vag))
                (SLOT - (SHORTCONST seller) (SHORTCONST Teo))
                (SLOT - (SHORTCONST item) (SHORTCONST Hornet)))
            (PSOA (INSTANCE (SHORTCONST liability))
                (TUPLE + (SHORTCONST Vag) (SHORTCONST Hornet)))
        )
    )
)
```

Παράδειγμα 2

Ο κώδικας του δεύτερου παραδείγματος παρουσιάζει τις υποκλάσεις (Subclass) και την ισότητα (Equal).

Σε Σύνταξη Παρουσίασης είναι:

```
RuleML (
  Assert (
   ?semfe##ntua
   ?hudson = 51
  )
)
```

και σε Σύνταξη Σειριοποίησης είναι:

```
<RuleML>
   <Assert>
      <Subclass>
         <Ind>semfe</Ind>
         <Rel>ntua</Rel>
      </Subclass>
      <Equal>
         <Var>Hudson</Var>
         <Ind>51</Ind>
      </Equal>
   </Assert>
</RuleML>
```

Το αφηρημένο συντακτικό δέντρο του παραπάνω αρχείου PSOA RuleML είναι:

```
(RuleML
    (Assert
        (## (SHORTCONST semfe) (SHORTCONST ntua))
        (= Hudson (SHORTCONST 51))
    )
)
```

ΚΕΦΑΛΑΙΟ 5

ΥΛΟΠΟΙΗΣΗ

Για την υλοποίηση των κλάσεων που χρειάζονται για τις μεταφράσεις αξιοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Eclipse. Έγινε χρήση της προγραμματιστικής γλώσσας Java, αξιοποιώντας το γλωσσικό εργαλείο A-
NTLR.

Ο ANTLR (ANother Tool for Language Recognition) είναι ένα εργαλείο ανάλυσης για ανάγνωση, επεξεργασία, εκτέλεση ή μετάφραση δομημένου κειμένου ή δυαδικών αρχείων. Χρησιμοποιείται ευρέως για τη δημιουργία γλωσσών, εργαλείων και πλαισίων. Ένα αρχείο ANTLR της έκδοσης 3.x έχει κατάληξη ".g". Αυτό το αρχείο περιέχει τους γραμματικούς και τους συντακτικούς κανόνες της γλώσσας που θα μεταγλωττίσει. [15] Η γενική δομή ενός αρχείου ANTLR είναι:

```
grammar first;
/*
 * Parser Rules
    options
    headers
    members
    rule-condition language%,
 */
/*
 * Lexer Rules
    tokens%,         tokens,
 */
```

Πιο αναλυτικά οι κανόνες του ANTLR χωρίζονται σε:

- συντακτικό αναλυτή (Parser), ο οποίος καθορίζει το συντακτικό δέντρο και μπορεί να είναι:

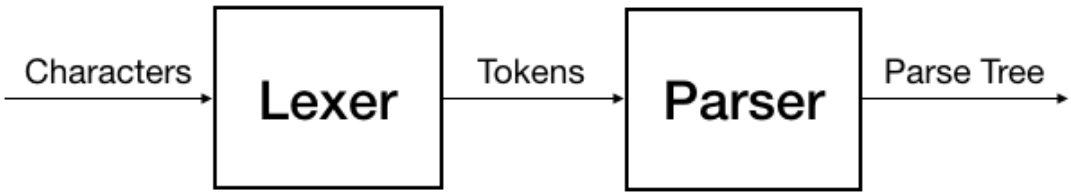
- επιλογές (options), οι οποίες καθορίζονται από τις ρυθμίσεις του ANTLR
- επικεφαλίδες (headers), όπου γίνεται εισαγωγή από τα πακέτα της Java
- μέλη (members), δηλώνονται τα πεδία Java και γράφονται οι μέθοδοί της
- κανόνες που προυποθέτει η γλώσσα (rule-condition language), περιλαμβάνει τους κανόνες που μετατρέπονται

• λεκτικός αναλυτής (Lexer Rules), ο οποίος καθορίζει τις λεκτικές μονάδες (tokens), και είναι:

- λεκτικές μονάδες (tokens), όπου γίνεται εισαγωγή των συμβολοσειρών της PSOA RuleML

.

Η λειτουργία του ANTLR φαίνεται στο ακόλουθο σχήμα:



Σχήμα 5.1: Λειτουργία ANTLR

Μετά την εισαγωγή του κώδικα δρα ο λεκτικός αναλυτής και ξεχωρίζει τους χαρακτήρες σε λεκτικές μονάδες. Στη συνέχεια ο συντακτικός αναλυτής παράγει το συντακτικό δέντρο με τερματικά σύμβολα τις λεκτικές μονάδες.

Μόλις γίνει επαλήθευση (compile) του αρχείου ANTLR δημιουργούνται αυτόματα οι κλάσεις σε γλώσσα Java (generated sources).

Για την πραγματοποίηση των μεταφράσεων της PSOA RuleML έγινε εισαγωγή, στο προγραμματιστικό περιβάλλον Eclipse, του υλικού (materials) που πρέπει να προσαρμοστεί προκειμένου να δημιουργηθούν οι αλγόριθμοι για τη μετάφραση. Το υλικό αυτό προήλθε από την ιστοσελίδα ”github.com”, και αποτελείται από τα εξής μέρη:

• PSOA2X, περιέχει τα αρχεία γραμματικής που χρησιμοποιούνται για τη μετατροπή ενός αφηρημένου συντακτικού δέντρου σε PSOA RuleML, καθώς και τις κλάσεις που χρειάζονται για τη μετάφραση αυτή.

• PSOACore, περιέχει τη γραμματική που αναλύει ο συντακτικός αναλυτής για τη μετάφραση από PSOA RuleML σε αφηρημένο συντακτικό δέντρο. Σε αυτόν τον φάκελο επίσης είναι και οι κλάσεις που αφορούν τη λειτουργία των αρχείων PSOA RuleML που εισάγονται.

• PSOATransRun, εδώ βρίσκονται οι κλάσεις που εκτελούνται προκειμένου να εμφανιστεί το αποτέλεσμα.

- PSOATransRunComponents, περιέχει τα υπόλοιπα τέσσερα μέρη συγκεντρωτικά.

- PSOATransRunWebService, αποτελεί μία διεπαφή για τον ιστό

Ένα από τα αρχεία που περιλαμβάνονται στο υλικό από το github.com είναι το `TreeWalkerTemplate.g`. Αυτό το αρχείο γραμματικής είναι το πρότυπο για όλους τους μετασχηματιστές. Οι κανόνες της γραμματικής που περιέχονται σε αυτό το αρχείο αντιστοιχούν στη γραμματική της μορφής EBNF της PSOA RuleML. Δηλαδή, εμπεριέχει όλα τα σύνολα συμβολοσειρών της PSOA RuleML. Άρα, τροποποιώντας κατάλληλα αυτό το αρχείο παράγεται ή το αφηρημένο συντακτικό δέντρο ή η μορφή Σύνταξης Παρουσίασης ή η μορφή Σύνταξης Σειριοποίησης.

Αναλυτικότερα, το `TreeWalkerTemplate.g` δεν έχει συμπληρωμένα τα κομμάτια που αφορούν τις επικεφαλίδες και τα μέλη του συντακτικού αναλυτή καθώς και τις λεκτικές μονάδες που περιλαμβάνει ο λεκτικός αναλυτής. Δηλαδή συμπληρωμένο είναι το κομμάτι των επιλογών και οι κανόνες της PSOA RuleML.

Οι επιλογές περιλαμβάνουν το αποτέλεσμα που θα εμφανίζει το αρχείο, την υπερκλάση στην οποία θα ανήκει το παραγόμενο αρχείο σε Java και τέλος δηλώνει το όριο των βημάτων του συντακτικού αναλυτή προκειμένου να επιτευχθεί η βέλτιστη ανάλυση.

Οι κανόνες της PSOA RuleML όπως τους έχει αποθηκευμένους το πρότυπο αρχείο για τους μετασχηματιστές δηλώνουν το αφηρημένο συντακτικό δέντρο της. Στον παρακάτω πίνακα παρουσιάζονται οι κανόνες της PSOA RuleML και οι εντολές του ANTLR για τη δημιουργία ενός αφηρημένου συντακτικού δέντρου. Πριν από αυτό προηγείται επεξήγηση κάποιων συμβόλων που περιέχονται στην απεικόνιση του αφηρημένου συντακτικού δέντρου και η ερμηνεία τους.

- κάθετη παύλα "|", συμβολίζει το διαζευκτικό ή

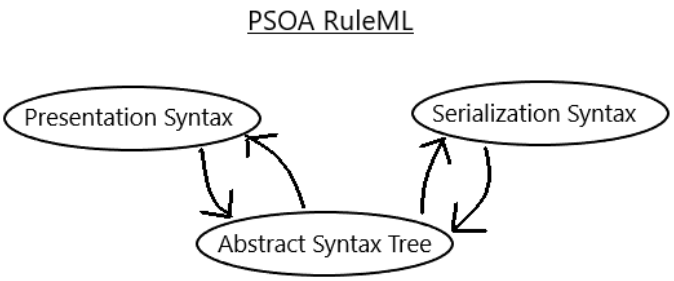
- αγγλικό ερωτηματικό "?", δηλώνει ότι μπορεί να υπάρχει ένα ή κανένα από τα σύμβολα που αναφέρεται

- συν "+", πρέπει να υπάρχει τουλάχιστον ένα τέτοιο στοιχείο

- αστερίσκος "*", μπορεί να μην περιλαμβάνει κανένα τέτοιο στοιχείο ή μπορεί να περιέχει άπειρα

Κανόνες PSOA RuleML	**AST**
document	(DOCUMENT base? prefix* importDecl* group?)
group	(GROUP group_element*)
group_element	rule group
query	formula
rule	(FORALL VAR_ID+ clause) clause
clause	(IMPLICATION head formula) head
head	atomic (AND head+) (EXISTS VAR_ID+ head)
formula	(AND formula+) (EXISTS VAR_ID+ formula) FALSITY naf_formula atomic external
naf_formula	(NAF formula)
atomic	atom equal subclass
atom	psoa
equal	(EQUAL term term)
subclass	(SUBCLASS term term)
term	constant VAR_ID psoa external
external	(EXTERNAL psoa)
psoa	(PSOA term? (INSTANCE term) tuple* slot*) (OIDLESSEMBATOM (INSTANCE term) tuple* slot*)
tuple	(TUPLE DEPSIGN term*)
slot	(SLOT DEPSIGN term term)
constant	(LITERAL IRI) (SHORTCONST constshort) TOP
constshort	IRI | LITERAL | NUMBER | LOCAL

Όπως έχει ξεκαθαριστεί, οι μεταφράσεις της γλώσσας PSOA RuleML από μορφή Σύνταξης Παρουσίασης σε Σύνταξη Σειριοποίησης και αντίστροφα επιτυγχάνονται μέσω των αφηρημένων συντακτικών δέντρων (Abstact Syntax Tree) που παράγονται αφού δράση ο συντακτικός αναλυτής (parser). Στην παρακάτω εικόνα φαίνονται καθαρά οι διαδρομές που ακολουθούνται για να επιτευχθεί το επιθυμητό αποτέλεσμα.



Σχήμα 5.2: Διαδρομές για τη μετάφραση

Η ανάλυση του κώδικα θα γίνει με βάση αυτές τις διαδρομές.

5.1 Από Σύνταξη Παρουσίασης σε Σύνταξη Σειριοποίησης

5.1.1 Το Αφηρημένο Συντακτικό Δέντρο της Σύνταξης Παρουσίασης

Το τελικό αρχείο που παράγει το αφηρημένο συντακτικό δέντρο της Σύνταξης Παρουσίασης είναι το ”PSOAPS.g”. Ακολουθεί παρουσίαση του αρχείου με βάση τον τύπο της ανάλυσης, δηλαδή συντακτική και λεκτική ανάλυση.

Συντακτικός Αναλυτής

Οι επιλογές σε αυτό το αρχείο είναι οι ίδιες με αυτές στο πρότυπο. Δηλαδή είναι το αποτέλεσμα που θα εμφανίζει το αρχείο, η υπερκλάση στην οποία θα ανήκει το παραγόμενο αρχείο σε Java και τέλος ο αριθμός των βημάτων του συντακτικού αναλυτή προκειμένου να επιτευχθεί η βέλτιστη ανάλυση, όπως έχει αναφερθεί.

Στις επικεφαλίδες δηλώνονται οι κλάσεις Java που θα χρησιμοποιηθούν και στα μέλη δηλώνονται τα πεδία και οι μέθοδοι της Java.

Στο τελευταίο μέρος του συντακτικού αναλυτή δηλώνεται η μετατροπή των κανόνων της PSOA RuleML από Σύνταξη Παρουσίασης σε αφηρημένο συντακτικό δέντρο. Το αποτέλεσμα θα είναι αυτό που φαίνεται στο πρότυπο αρχείο, ”TreeWalkerTemplate-
.g”, αλλά η εισαγωγή θα είναι σε Σύνταξη Παρουσίασης.

Στον ακόλουθο πίνακα είναι τα τερματικά και μη τερματικά σύμβολα της PSOA RuleML και πως αυτά εμφανίζονται σε Σύνταξη Παρουσίασης και αφηρημένο συντακτικό δέντρο:

(non) terminal	**PS**	**AST**
document	RuleML()	(RuleML)
group	Assert()	(Assert)
rule	Forall()	(FORALL)
clause	:-	:-
AND	And()	(AND)
naf_formula	Naf()	(NAF)
Atom		PSOA
oid		SHORTCONST
rel	#	(#)
Ind		SHORTCONST
VAR	?	
tuple	+[] -[]	TUPLE + TUPLE -
slots	+> ->	SLOT + SLOT -
equal	=	(EQUAL)
subclass	##	(SUBCLASS)

Πίνακας 5.1: Σύμβολα PS και AST

Λεκτικός Αναλυτής

Ο λεκτικός αναλυτής του αρχείου περιλαμβάνει:

- τις λέξεις κλειδιά (Keywords) και πως εμφανίζονται στον κώδικα:

DOCUMENT	RuleML
BASE	Base
IMPORT	Import
PREFIX	Prefix
GROUP	Group ή Assert
FORALL	Forall
EXISTS	Exists
AND	And
OR	Or
EXTERNAL	External
TOP	Top

- τα σύμβολα (Operators) που περιέχουν οι γλώσσες στη σύνταξή τους:

IMPLICATION	:-
EQUAL	=
SUBCLASS	##
INSTANCE	#
SLOT_ARROW	-> ή +>
SYMSPACE_OPER	^^

- τα σημεία στίξης (Punctuation):

LPAR	(
RPAR	)
LESS	<
GREATER	>
DEPSIGN	+ ή -
LSQBR	[
RSQBR	]

- τη δήλωση σταθερών (Constants), αλφαριθμητικών

- και τα fragment που είναι ένας ειδικός τύπος λεκτικού κανόνα (lexer rule) ο οποίος δεν έχει ως αποτέλεσμα τη δημιουργία λεκτικών μονάδων (tokens) αλλά υπάρχει μόνο για την εισαγωγή λογικής έκφρασης που απλοποιεί τη γραμματική. Για παράδειγμα fragment θεωρούνται τα αλφαριθμητικά σύμβολα.

5.1.2 Από το Αφηρημένο Συντακτικό Δέντρο σε Σύνταξη Σειριοποίησης

Το αρχείο που δημιουργήθηκε για να μεταφράσει το αρχείο σε XML είναι το `XMLConverter.g`. Ο λεκτικός αναλυτής είναι ίδιος με αυτόν στο `PSOAPS.g`. Για να μην επαναλαμβάνεται και στο `XMLConverter.g` εκχωρείται στις επιλογές του συντακτικού αναλυτή. Οι υπόλοιπες επιλογές είναι οι ίδιες. Επίσης ίδιες είναι και οι επικεφαλίδες καθώς και τα μέλη. Η μόνη διαφορά είναι ότι στα μέλη έχει προστεθεί και μία μέθοδος για να μορφοποιεί το αποτέλεσμα XML.

Το υπόλοιπο αρχείο είναι οι κανόνες μου μεταφράζονται από το αφηρημένο συντακτικό δέντρο που δημιουργήθηκε απο το `PSOAPS.g` σε Σύνταξη Σειριοποίησης, δηλαδή στη μορφή XML.

(non) terminal	AST	XML

document	(RuleML)	`<RuleML>`
		`...`
		`</RuleML>`

group	(Assert)	`<Assert>`
		`...`
		`</Assert>`

rule	(FORALL)	`<Forall>`
		`...`
		`</Forall>`

clause	:-	`<Implies>`
		`<If>`
		`...`
		`</If>`
		`<then>`
		`...`
		`</then>`
		`</Implies>`

AND	(AND)	`<And>`
		`...`
		`</And>`

naf_formula	(NAF)	`<Naf>`
		`...`
		`</Naf>`

Atom	PSOA	`<Atom>`
		`...`
		`</Atom>`

(non) terminal	AST	XML

oid	SHORTCONST	`<oid>`
		`<Ind>`
		...
		`</Ind>`
		`</oid>`

rel	(#)	`<Rel>`
		...
		`</Rel>`

Ind	SHORTCONST	`<Ind>`
		...
		`</Ind>`

VAR		`<Var>`
		...
		`</Var>`

equal	(EQUAL)	`<Equal>`
		...
		`</Equal>`

subclass	(SUBCLASS)	`<Subclass>`
		...
		`</Subclass>`

(non) terminal	AST	XML

tuple	TUPLE +
	TUPLE -
		`<tupdep>`
		`  <Tuple>`
		`  ...`
		`  </Tuple>`
		`</tupdep>`
		
		`<tup>`
		`  <Tuple>`
		`  ...`
		`  </Tuple>`
		`</tup>`

slots	SLOT +
	SLOT -
		`<slotdep>`
		`  ...`
		`</slotdep>`
		
		`<slot>`
		`  ...`
		`</slot>`

5.1.3 Εκτέλεση

Το πρώτο βήμα για τη μετάφραση γίνεται στην κλάση `PSOATransRunCmdLine`. Στην κλάση αυτή καλείται αρχικά το εισαγόμενο αρχείο, όπου γίνεται η εμφάνισή της βάσης γνώσης αν ζητηθεί από τον χρήστη, και στη συνέχεια δίνονται οι κατάλληλες εντολές για την εκτέλεση της μετάφρασης. Πρόκειται δηλαδή για τη διεπαφή του χρήστη με τη μετάφραση. Ακολουθεί η μετατροπή του αρχείου από Σύνταξη Παρουσίασης σε αφηρημένο συντακτικό δέντρο.

Οι μέθοδοι για την μετατροπή αυτή βρίσκονται στην κλάση `PSOATransRun`, όπως επίσης και οι μέθοδοι για τη μετάφραση στη γλώσσα επιλογής. Αρχικά, στην κλάση αυτή υλοποιείται η μέθοδος που καλεί την κλάση `PSOAKB`.

Στην `PSOAKB` γίνεται η αποθήκευση της βάσης γνώσης της Σύνταξης Παρουσίασης και καλούνται τα παραγόμενα (από το αρχείο `PSOAPS.g` του ANTLR) αρχεία Java, `PSOAPSLexer` και `PSOAPSParser`. Πραγματοποιείται δηλαδή η μετατροπή της Σύνταξης Παρουσίασης σε αφηρημένο συντακτικό δέντρο. Η `PSOAKB` είναι υποκλάση της `PSOAInput`.

Στην κλάση `PSOAInput` βρίσκεται η μέθοδος που χρησιμοποιεί την κλάση που περιέχει τον μεταφραστή `XMLTranslator` και είναι υποκλάση της `ANTLRBasedTranslator`.

Η μέθοδος που κάνει τη μετάφραση από αφηρημένο συντακτικό δέντρο σε XML χρησιμοποιεί, την παραγόμενη από το αρχείο `XMLConverter.g`, κλάση Java.

Το αποτέλεσμα καλείται από την κλάση `ANTLRBasedTranslator`, η οποία είναι υποκλάση της `Translator`.

Στην `Translator` υπάρχουν οι μέθοδοι που συνδέουν το αποτέλεσμα με την κλάση `PSOATransRun`, όπου υπάρχει η μέθοδος για την εμφάνιση της μετάφρασης.

5.2 Από Σύνταξη Σειριοποίησης σε Σύνταξη Παρουσίασης

Σε γενικές γραμμές η μετάφραση σε Σύνταξη Παρουσίασης από Σύνταξη Σειριοποίησης είναι παρόμοια με την αντίστροφη. Δηλαδή υπάρχει το αρχείο ANTLR `PSOAXML.g` που παράγει το αφηρημένο συντακτικό δέντρο από Σύνταξη Σειριοποίησης, το αρχείο `PSOAConverter.g` που μετατρέπει σε Σύνταξη Παρουσίασης και οι κατάλληλες μέθοδοι για την πραγματοποίηση και την εμφάνιση της μετάφρασης.

5.2.1 Το Αφηρημένο Συντακτικό Δέντρο της Σύνταξης Σειριοποίησης

Το αρχείο ANTLR `PSOAXML.g` που παράγει το αφηρημένο συντακτικό δέντρο από τη μορφή Σύνταξης Σειριοποίησης είναι μία τροποποίηση του αρχείου `PSOAPS.g` που παράγει το ίδιο ακριβώς αποτέλεσμα αλλά από τη μορφή Σύνταξης Παρουσίασης.

Ο λεκτικός αναλυτής δε χρειάστηκε κάποια αλλαγή. Περιλαμβάνει δηλαδή τις ίδιες ενότητες που αφορούν

- τις λέξεις κλειδιά (Keywords) και πως εμφανίζονται στον κώδικα
- τα σύμβολα (Operators) που περιέχουν οι γλώσσες στη σύνταξή τους
- τα σημεία στίξης (Punctuation)
- η δήλωση σταθερών (Constants)
- και τα fragment.

Ο συντακτικός αναλυτής παρέμεινε ο ίδιος εκτός από το κομμάτι των κανόνων, οι οποίοι προσαρμόστηκαν ώστε να αναγνωρίζεται η γλώσσα XML. Στον πίνακα που ακολουθεί φαίνονται τα μη τερματικά σύμβολα της PSOA RuleML στη μορφή της Σύνταξης Σειριοποίησης και ως αφηρημένο συντακτικό δέντρο:

(non) terminal	XML	AST

document	`<RuleML>`
	`...`
	`</RuleML>`
		(RuleML)

group	`<Assert>`
	`...`
	`</Assert>`
		(Assert)

rule	`<Forall>`
	`...`
	`</Forall>`
		(FORALL)

clause	`<Implies>`
	`<If>`
	`...`
	`</If>`
	`<then>`
	`...`
	`</then>`
	`</Implies>`
		:-

AND	`<And>`
	`...`
	`</And>`
		(AND)

naf_formula	`<Naf>`
	`...`
	`</Naf>`
		(NAF)

Atom	`<Atom>`
	`...`
	`</Atom>`
		PSOA

(non) terminal	XML	AST

oid	`<oid>`
	`  <Ind>`
	`  ...`
	`  </Ind>`
	`</oid>`
		SHORTCONST

rel	`<Rel>`
	`  ...`
	`</Rel>`
		(#)

Ind	`<Ind>`
	`  ...`
	`</Ind>`
		SHORTCONST

VAR	`<Var>`
	`  ...`
	`</Var>`	

equal	`<Equal>`
	`  ...`
	`</Equal>`
		(EQUAL)

subclass	`<Subclass>`
	`  ...`
	`</Subclass>`
		(SUBCLASS)

(non) terminal	XML	AST

tuple	`<tupdep>`
	`  <Tuple>`
	`  ...`
	`  </Tuple>`
	`</tupdep>`
	
	`<tup>`
	`  <Tuple>`
	`  ...`
	`  </Tuple>`
	`</tup>`
		TUPLE +
		TUPLE -

slots	`<slotdep>`
	`  ...`
	`</slotdep>`
	
	`<slot>`
	`  ...`
	`</slot>`
		SLOT +
		SLOT -

5.2.2 Από το Αφηρημένο Συντακτικό Δέντρο σε Σύνταξη Παρουσίασης

Ο κώδικας που πραγματοποιεί αυτή τη μετάφραση μπορεί να βρεθεί στο αρχείο ANTLR `PSOAConverter.g`. Είναι όμοιο με το `XMLConverter.g` με τις διαφορές ότι δεν υπάρχουν κλάσεις για τη μορφοποίηση του αποτελέσματος και οι κανόνες είναι αυτοί που αντιστοιχούν στη Σύνταξη Σειριοποίησης.

Ο λεκτικός αναλυτής εκχωρείται στον συντακτικό αναλυτή από το πρωτότυπο αρχείο `PSOAPS.g`. Αυτούσια μεταφέρονται στο συντακτικό αναλυτή οι επιλογές, οι επικεφαλίδες και τα μέλη. Οι κανόνες χρειάστηκαν την κατάλληλη προσαρμογή. Ο παρακάτω πίνακας δείχνει αυτήν την προσαρμογή.

(non) terminal	AST	PS
document	(RuleML)	RuleML()
group	(Assert)	Assert()
rule	(FORALL)	Forall()
clause	:-	:-
AND	(AND)	And()
naf_formula	(NAF)	Naf()
Atom	PSOA	
oid	SHORTCONST	
rel	(#)	#
Ind	SHORTCONST	
VAR		?
tuple	TUPLE + TUPLE -	+[] -[]
slots	SLOT + SLOT -	+> ->
equal	(EQUAL)	=
subclass	(SUBCLASS)	##

5.2.3 Εκτέλεση

Η εκτέλεση είναι η αντίστοιχη με την αντίστροφή της, με τη μόνη διαφορά ότι καλεί τις αντίστοιχες κλάσεις.

Επιγραμματικά, το εισαγόμενο αρχείο περνάει πρώτα από τη κλάση `PSOATransRunCmdLine` και μετά από τη `PSOATransRun` για να ξεκινήσει η μετατροπή. Καλούνται οι μέθοδοι από τη κλάση `PSOAKB` για να δημιουργηθεί το αφηρημένο συντακτικό δέντρο, χρησιμοποιώντας τους μετατροπείς και από την κλάση `PSOAInput` γίνεται η σύνδεση με την `PSOATranslator`. Εκεί χρησιμοποιείται ο `PSConverter.g` για την τελική μετατροπή. Η εμφάνιση γίνεται μέσω της κλάσης `PSOATransRun`.

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΛΟΓΟΣ

6.1 Συμπεράσματα

Συνοψίζοντας, στην εργασία αυτή έγινε μία πρώτη προσπάθεια μετάφρασης δύο μορφών αναπαράσταση γνώσης της PSOA RuleML, της Σύνταξης Παρουσίασης και της Σύνταξης Σειριοποίησης, με τη χρήση των αφηρημένων συντακτικών δέντρων.

Αρχικά έγινε παρουσίαση των κανόνων στην κάθε μορφή της PSOA RuleML με τη χρήση των ανάλογων παραδειγμάτων για την ευκολότερη κατανόησή τους. Ξεκινώντας από ένα πιο απλό παράδειγμα, προσθέτονταν συνεχώς κι άλλοι κανόνες με σκοπό να καλυφθούν, αν όχι όλοι, η συντριπτική πλειοψηφία τους. Ανάλογη παρουσίαση επίσης έγινε και για τα αφηρημένα συντακτικά δέντρα, καθώς μέσω αυτών πραγματοποιούνται οι εν λόγω μεταφράσεις.

Στο πέμπτο κεφάλαιο, το οποίο αποτελεί το κεφαλαίο της υλοποίησης, εξηγούνται οι τρόποι με τους οποίους πραγματοποιούνται οι μεταφράσεις. Εξηγείται αρχικά το εργαλείο του ANTLR που χρησιμοποιήθηκε. Με αυτό το εργαλείο παρήχθησαν οι κώδικες που αφορούν τις μετατροπές, τόσο του συντακτικού αναλυτή όσο και του λεκτικού. Αξιοποιήθηκε δηλαδή για τη δημιουργία των αναγκαίων κλάσεων σε προγραμματιστική γλώσσα Java. Πιο συγκεκριμένα εισήχθησαν οι κανόνες της PSOA RuleML στην κάθε μορφή και εμφανιζόταν στη μορφή που προγραμματίστηκε να παράγει, είτε σε Σύνταξη Παρουσίασης, είτε σε Σύνταξη Σειριοποίησης, είτε σε αφηρημένο συντακτικό δέντρο. Στη συνέχεια αυτού, χρειάστηκε η ανάλογη προσαρμογή του ήδη υπάρχοντος κώδικα PSOATransRun, ο οποίος είναι υπεύθυνος για τη μετάφραση κάποιων άλλων μορφών αναπαράστασης γνώσης, ώστε να παράγει το επιθυμητό αποτέλεσμα.

Καθώς υπήρχε η μετάφραση από τη μορφή της Σύνταξης Παρουσίασης σε κάποια άλλη, η προσαρμογή για να εμφανίζει τη μορφή της Σύνταξης Σειριοποίησης πραγματοποιήθηκε με κάποιες αλλαγές στον αντίστοιχο κώδικα. Η μετατροπή δηλαδή της Σύνταξης Παρουσίασης σε αφηρημένο συντακτικό δέντρο αξιοποιήθηκε αυτούσια από την ήδη υπάρχουσα μετάφραση. Χρειάστηκε όμως η μετατροπή που είναι υπεύθυνη για τη μετάφραση από το

αφηρημένο συντακτικό δέντρο σε Σύνταξη Σειριοποίησης δηλαδή σε XML. Μετά το πέρας αυτής της διαδικασίας δηλώθηκε στις παραμέτρους η αντίστοιχη επιλογή συτής της μετάφρασης ώστε να εμφανίζει το αποτέλεσμα.

Ο κώδικας που αφορά τη μετάφραση από Σύνταξη Σειριοποίησης σε Σύνταξης Παρουσίασης έπρεπε να δημιουργηθεί από την αρχή, μιας και δεν έχει πραγματοποιηθεί ανάλογη μετάφραση. Οι δηλώσεις των κανόνων και των αποτελεσμάτων υλοποιήθηκαν εύκολα με τη βοήθεια του ANTLR. Δύσκολο αποδείχθηκε το κομμάτι της δήλωσης που αφορά την αναγνώριση σε πρώτο επίπεδο. Πως δηλαδή θα αναγνωρίζει το πρόγραμμα ότι ο εισαγόμενος κώδικας είναι σε μορφή XML, και στη συνέχεια να πράττει την μετάφραση. Για την επίλυση αυτής της δυσκολίας που προέκυψε χρειάστηκε προσεκτική προσέγγιση στις παραμέτρους που θα διάβαζε το πρόγραμμα.

6.2 Μελλοντικές Επεκτάσεις

Οι μελλοντικές επεκτάσεις του αντικειμένου της παρούσας διπλωματικής μπορούν να χωριστούν σε δύο κύρια ζητήματα. Πρώτον, η βελτίωση των μεταφράσεων που ασχολήθηκε η εργασία αυτή και δεύτερον το έναυσμα που μπορεί να αποτελέσει για ανάλογες μεταφράσεις.

Το πρώτο ζήτημα αφορά κυρίως το αποτέλεσμα της εμφάνισης της μετάφρασης. Πιο συγκεκριμένα, η παραχθείσα μορφή της Σύνταξης Παρουσίασης θα μπορούσε να παρουσιάζεται στον χρήστη με τις γνωστές εναλλαγές των γραμμών καθώς και το κενό στην αρχή αυτών για τη βέλτιστη ευκρίνεια στην ανάγνωση, όπως συμβαίνει στη μορφή της Σύνταξης Σειριοποίησης. Κάποιες όμοιες προσαρμογές θα μπορούσαν να εφαρμοστούν και για τα αφηρημένα συντακτικά δέντρα, αν και δεν είναι τόσο μεγάλης σημασίας μιας και ο χρήστης δε θα τα συναντήσει σε μια απλή χρήση του προγράμματος.

Όσον αφορά στο δεύτερο ζήτημα. Είναι γεγονός ότι υπάρχουν ήδη ορισμένες μεταφράσεις μεταξύ των μορφών αναπαράστασης γνώσης. Όμως με την αύξηση του αριθμού των μορφών αυτών μεγαλώνει και το πρόβλημα "επικοινωνίας" μεταξύ των χρηστών. Καθίσταται λοιπόν υπαρκτή η ανάγκη να αυξηθούν και οι μεταφραστές για τη σύνδεση των μορφών έτσι ώστε να επιτυγχάνεται η βέλτιστη αξιοποίησή των πληροφοριών που περιέχονται.

Μία τέτοια μετάφραση θα μπορούσε να είναι μεταξύ των πρότυπων γλωσσών της RuleML και της RIF. Η κάθε μία από αυτές τις μορφές αποτελεί από μόνη της ένα σύνολο κανόνων και ερωτημάτων για αναπαράσταση γνώσης. Άρα έχουν και οι δύο τις μορφές της Σύνταξης Σειριοποίησης και της Σύνταξης Παρουσίασης. Ένας μεταφραστής θα μπορούσε να αφορά τις μορφές αυτές της μία πρότυπης γλώσσας με την άλλη. Δηλαδή, να πραγματοποιείται αμφίδρομη μετατροπή της Σύνταξης Σειριοποίησης της RuleML με αυτή της RIF και αντίστοιχα της Σύνταξης Παρουσίασης. Από τη στιγμή που υπάρχουν οι μεταφραστές για την αμφίδρομη μετατροπή της κάθε μορφής των γλωσσών, θα μπορούσε να υπάρξει σύνδεση όλων των μορφών και από τις δύο γλώσσες, για παράδειγμα από τη Σύνταξη Σειριοποίησης της RIF να προκύπτει η Σύνταξη Παρουσίασης της RuleML. Μία υπόδειξη για την πιθανή μελλοντική απόπειρα δημιουργίας των εν λόγω μεταφραστών είναι ότι ορισμένες έννοιες που υπάρχουν στην RuleML δεν υπάρχουν στη RIF. Για παράδειγμα, δεν υπάρχει η έννοια της προοπτικής, δηλαδή δεν υπάρχουν εξαρτημένα ορίσματα, ούτε slots ούτε πλειά-

δες. Παρά τις διαφορές που υπάρχουν όμως, μπορεί να δημιουργηθεί ο μεταφραστής, με τις κατάλληλες προσαρμογές στους κανόνες των γλωσσών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] W3C. XML Tree node. https://www.w3schools.com/xml/nodetree.gif.

[2] PSOA RuleML. Psoa ruleml. http://wiki.ruleml.org/index.php/PSOA_RuleML.

[3] PSOA RuleML. Introduction. https://wiki.ruleml.org/index.php/PSOA_RuleML#Introduction.

[4] PSOA RuleML. Presentation preview. http://wiki.ruleml.org/index.php/PSOA_RuleML#Presentation_Preview.

[5] Κωστής Σαγώνας. Σύνταξη και Συντακτική Ανάλυση. https://courses.softlab.ntua.gr/pl1/2008a/Slides/lecture-03-bw.pdf.

[6] PSOA RuleML. Monolithic ebnf for psoa ruleml presentation syntax. http://wiki.ruleml.org/index.php/PSOA_RuleML#Monolithic_EBNF_for_PSOA_RuleML_Presentation_Syntax.

[7] W3C. XML Home Introduction HowToUse. https://www.w3schools.com/xml/default.asp https://www.w3schools.com/xml/xml_whatis.asp https://www.w3schools.com/xml/xml_usedfor.asp.

[8] W3C. Xml syntax rules. https://www.w3schools.com/xml/xml_syntax.asp.

[9] W3C. Xml elements. https://www.w3schools.com/xml/xml_elements.asp.

[10] W3C. Xml attributes. https://www.w3schools.com/xml/xml_attributes.asp.

[11] W3C. Xml namespaces. https://www.w3schools.com/xml/xml_namespaces.asp.

[12] W3C. Displaying xml. https://www.w3schools.com/xml/xml_display.asp.

[13] Rima Chaudhari Tara Athan, Harold Boley. Specifying psoa ruleml/xml 1.03: Myng-modularized schemas for the rnc xsd validation of xslt-normalized data and knowledge. http://www.cs.unb.ca/~boley/papers/SpecPSOARuleMLXML103.pdf.

[14] Κωστής Σαγώνας. Σύνταξη και Συντακτική Ανάλυση. https://courses.softlab.ntua.gr/pl1/2008a/Slides/lecture-03-bw.pdf.

[15] DataCadamia. Antlr - lexer rule (token names|lexical rule). https://datacadamia.com/antlr/lexer_rule.