ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

Αυτόματη μετατροπή οντολογίας σε άλλες απλούστερες μορφές XML με τη χρήση XSLT και άλλων εργαλείων Web

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΚΟΝΤΑΚΗ ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων: Αθανάσιος Γ. Μαλάμος
Προϊστάμενος Τμήματος ΕΠΠ

Ηράκλειο, Μάρτιος 2011

Η σελίδα αυτή είναι σκόπιμα κενή

Περίληψη

Στην εργασία αυτή θα γίνει μελέτη του μετασχηματισμού οντολογιών που εκφράζονται σε ανώτερου επιπέδου τεχνικές αναπαράστασης (RDF, OWL) σε απλούστερες δομές XML, με στόχο την αποδοτικότερη και ταχύτερη εφαρμογή αλγορίθμων σύγκρισης. Η αποτύπωση οντολογιών σε γλώσσες όπως οι RDF και OWL οδηγεί στην δημιουργία οντολογικών γράφων, των οποίων η δομή, είναι δύσκολο να καθοριστεί και να ενταχθεί σε μία τυπική κατηγορία που θα βοηθούσε να εφαρμοστούν αλγόριθμοι σύγκρισης όπως είναι για παράδειγμα τα δέντρα. Στόχος της εργασίας αυτής είναι να εξευρεθούν τεχνικές για την αυτόματη μετατροπή οντολογιών σε ιεραρχικά XML ώστε να είναι δυνατή η γρήγορη αναζήτηση και εκμετάλλευση της σημαντικής πληροφορίας που περιέχει μία οντολογική αποτύπωση.

Λέξεις Κλειδιά: OWL, RDF, Protégé-OWL, SWRL, SQWRL, XSLT

Η σελίδα αυτή είναι σκόπιμα κενή

Abstract

In this thesis becomes study of transformation of ontologies that are expressed in higher level techniques of representation (such as RDF and OWL), into simpler XML structures, aiming at the more efficient and more rapid application of comparison algorithms. The imprinting of ontologies in languages as the RDF and OWL leads to the creation of ontological graphs, which the structure of, it is difficult to be determined and to include itself in a formal category that would help into the appliance of comparison algorithms, as are as an example the trees. Objective of this work is found techniques for the automatic transformation of ontologies in hierarchical XML so that is possible the fast search and exploitation of important information contained in an ontological imprinting.

Keywords: OWL, RDF, Protégé-OWL, SWRL, SQWRL, XSLT

Η σελίδα αυτή είναι σκόπιμα κενή

Πίνακας περιεχομένων

Η σελίδα αυτή είναι σκόπιμα κενή

Πίνακας Σχημάτων

Η σελίδα αυτή είναι σκόπιμα κενή

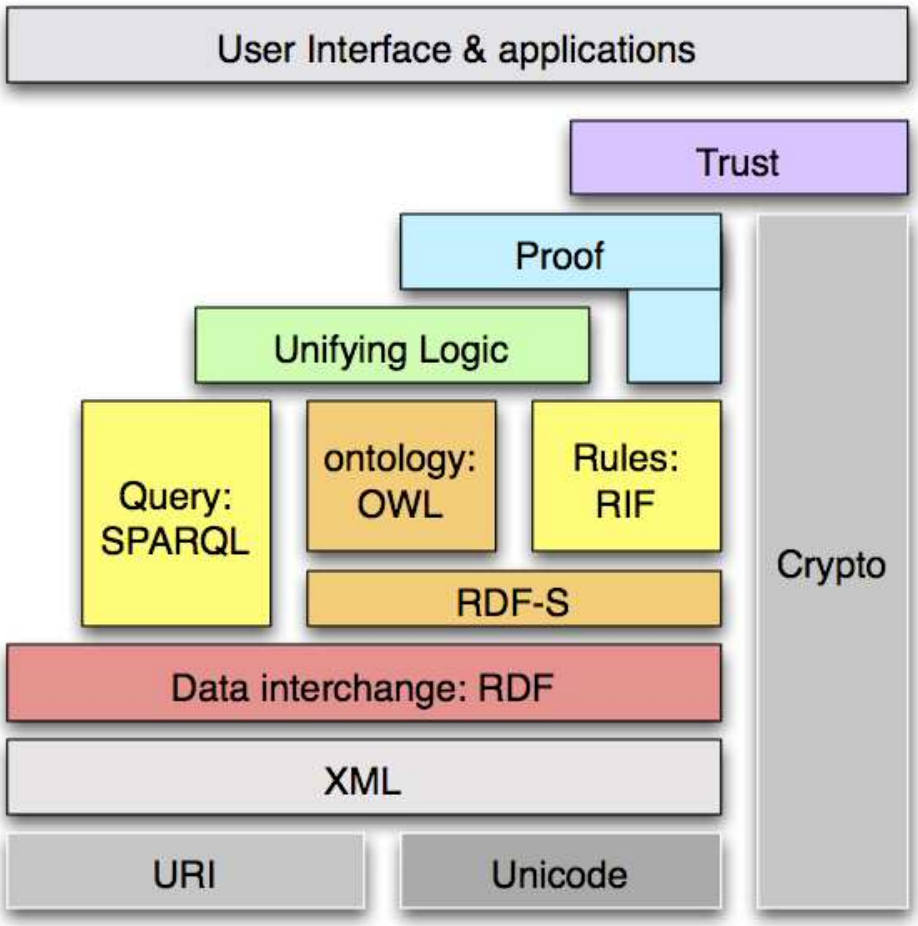
Πίνακας Εικόνων

Η σελίδα αυτή είναι σκόπιμα κενή

Λίστα Πινάκων

Η σελίδα αυτή είναι σκόπιμα κενή

1. ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ



1.1 OWL και RDF Schema

Η Web Ontology Language σχεδιάστηκε για χρήση από εφαρμογές που πρέπει να υποβάλλουν σε επεξεργασία τα περιεχόμενα των πληροφοριών, αντί απλώς να παρουσιάζουν αυτήν την πληροφορία στον άνθρωπο. Η OWL διευκολύνει τις μηχανές στην ερμηνεία του Web περιεχομένου σε σχέση με αυτήν που υποστηρίζεται από τα πρότυπα XML, RDF και RDF Schema, παρέχοντας μεγαλύτερο λεξιλόγιο μαζί με την τυποκρατική σημασιολογία (formal semantics). Αυτό οφείλεται σε μεγάλο μέρος, στην υιοθέτηση πολλών βασικών στοιχείων του RDF Schema. Το RDFS πρόκειται για μια γλώσσα αναπαράστασης γνώσης, που παρέχει βασικά στοιχεία για την περιγραφή οντολογιών, γνωστά ως Resource Description Framework (RDF) λεξιλόγια, με σκοπό να δομήσουν RDF resources. Όμως, η OWL υιοθετώντας το RDFS, κληρονομεί και τους αντίστοιχους περιορισμούς που έχει στην εκφραστική του δύναμη. Περιορισμοί τέτοιου τύπου συναντώνται στις περιπτώσεις των:

- Ιδιοτήτων, καθορίζουν ένα range μιας ιδιότητας για όλες τις κλάσεις που σχετίζονται με αυτή την ιδιότητα. Είναι αδύνατη η δήλωση περιορισμών στο range που αναφέρονται σε κάποιες συγκεκριμένες κλάσεις. Παραδείγματος χάριν, είναι αδύνατο να πούμε ότι ένα πρόβατο τρώει μόνο φυτά, ενώ υπάρχουν και άλλα ζώα που μπορούν να τρώνε και κρέας

- `Disjointness of classes`, μερικές φορές είναι επιθυμητό κάποιες κλάσεις να είναι disjoint μεταξύ τους, δηλαδή να μην έχουν κανένα απολύτως στοιχείο κοινό. Παραδείγματος χάριν, δύο κλάσεις που αντιπροσωπεύουν από ένα φύλο η καθεμιά τους, άντρας και γυναίκα αντίστοιχα
- `Αλγεβρικών Συνδυασμών κλάσεων`, μερικές φορές θέλουμε να συνδυάσουμε κλάσεις μέσω της ένωσής τους, της τομής τους και του συμπληρώματος. Παραδείγματος χάριν, η κλάση άνθρωπος αποτελεί την ένωση των δύο disjoint κλάσεων άντρας και γυναίκα
- `Cardinality περιορισμών`, παραδείγματος χάριν, ένας άνθρωπος έχει ακριβώς δύο γονείς ή ένα μάθημα διδάσκεται από τουλάχιστον ένα διδάσκοντα
- `Χαρακτηριστικά ιδιοτήτων`, αδυναμία υποστήριξης μεταβατικών ιδιοτήτων (μεγαλύτερο από), αδυναμία υποστήριξης αποκλειστικής ιδιότητας (unique), μη ύπαρξη αντίστροφης ιδιότητας μιας άλλης ιδιότητας (“τρώει” και “τρώγεται από”)

Παρόλο που η OWL θα μπορούσε να επεκτείνει το RDF Schema σύμφωνα με τα επίπεδα που καθορίζει η αρχιτεκτονική του Semantic Web, θα προκαλούσε προβλήματα όσων αφορά την απόκτηση εκφραστικής δύναμης και αποδοτικού συλλογισμού. Αυτό συμβαίνει επειδή ο συνδυασμός του RDF Schema με περιγραφές λογικής οδηγεί σε ανεξέλεγκτες υπολογιστικές καταστάσεις. Η λύση που δόθηκε αποσκοπεί στην χρήση τριών διαφορετικών υπογλωσσών ανάλογα με τις απαιτήσεις του χρήστη, με κάθε υπογλώσσα να έχει αυξημένη εκφραστικότητα σε σχέση με την προηγούμενη στην κατάταξη:

- Η `OWL Lite` αποτελεί μια περιορισμένη έκδοση της OWL DL με βασικό πλεονέκτημα την ευκολία υλοποίησης και εφαρμογής οντολογιών για απλούς χρήστες και χρήστες ανάπτυξης εργαλείων. Η αδυναμίας της είναι η υπερβολικά περιορισμένη εκφραστικότητα αφού αφαιρεί τους περισσότερους constructors της γλώσσας σε σχέση με τα άλλα δύο είδη OWL
- Η `OWL DL` αποτελεί μια περιορισμένη έκδοση της OWL Full, ανταποκρίνεται σε μια καλά μελετημένη περιγραφή λογικής (description logic) και επιτρέπει επαρκή υποστήριξη συλλογισμού. Ωστόσο, υστερεί στην πλήρη συμβατότητα με το RDF που σημαίνει ότι δεν είναι όλα τα RDF έγγραφα έγκυρα OWL DL έγγραφα ενώ την ίδια στιγμή κάθε OWL DL έγγραφο είναι ένα έγκυρο RDF έγγραφο
- Η `OWL Full` είναι πλήρως συμβατή με το RDF συντακτικά και σημασιολογικά αλλά αυτό έχει ως αποτέλεσμα την ατελή ή ανεπαρκή υποστήριξη συλλογισμού. Αυτό είναι γνωστό ως undecidable problem, κατά το οποίο κανένας αλγόριθμος δε μπορεί να δώσει λύση σε ένα πρόβλημα λήψης απόφασης

1.2 OWL και Protégé-OWL

Αδυναμίες του SWRL Editor

Ο SWRL Editor είναι μια επέκταση του SWRLTab του Protégé-OWL που επιμελείται την συγγραφή των SWRL κανόνων. Όμως, ο SWRL Editor εκτελεί μόνο δύο βασικές λειτουργίες κατά τη συγγραφή των κανόνων, ελέγχει το συντακτικό και τη σημασιολογία των κανόνων. Με απλά λόγια, εγγυάται ότι ένας κανόνας είναι συντακτικά σωστός πριν αποθηκευτεί και επίσης εγγυάται ότι οποιαδήποτε αναφορά σε OWL οντότητες είναι έγκυρη. Παρόλα αυτά, ο editor δε μπορεί να πραγματοποιήσει κανένα απολύτως έλεγχο ακεραιότητας ανάμεσα στους κανόνες σε μια οντολογία, δε μπορεί να εγγυηθεί την αποτροπή της διένεξης των κανόνων με τα OWL axioms, και δε μπορεί να αποτρέψει τον χρήστη από το να γράψει ορθά συντακτικούς και σημασιολογικούς κανόνες, αλλά χωρίς κανένα απολύτως νόημα.

Περιορισμοί γέφυρας

Η SWRL Jess Bridge είναι η μοναδική γέφυρα που υλοποιείται μέχρι στιγμής στο Protégé-OWL, και διαθέτει κάποιους περιορισμούς όσον αφορά την διάδρασή της με την OWL. Ο σημαντικότερος ίσως περιορισμός της, είναι ότι δεν μπορεί να αναπαραστήσει όλα τα OWL Axioms κατά την ανταλλαγή γνώσης από την OWL οντολογία στη γέφυρα. Αυτή η αδυναμία της Jess, οδηγεί τους

συμπερασματικούς μηχανισμούς να μην γνωρίζουν για τα εναπομείναντα OWL axioms, με αποτέλεσμα να μην εξάγονται όλα τα πιθανά αποτελέσματα. Επίσης, υπάρχει η πιθανότητα διενέξεων στην OWL οντολογία, ανάμεσα στην πληροφορία που εισήχθη από μια μηχανή κανόνων και στα OWL axioms που δεν λήφθηκαν υπόψιν. Προς το παρών, τέτοιες διενέξεις δεν εντοπίζονται αυτόματα, και η επίλυση μιας τέτοιας διένεξης – που στην πραγματικότητα αφορά ένα SWRL κανόνα και ένα ή περισσότερα OWL axioms, και δεν έχει να κάνει με μια συγκεκριμένη rule engine – αφήνεται στον χρήστη. Τυχόν διενέξεις μπορούν να εντοπιστούν εκτελώντας έναν OWL reasoner στην οντολογία που έχει εμπλουτιστεί από συμπερασματική γνώση (inferred knowledge).

Ένα επιπλέον ζήτημα είναι ότι μια χωρίς διενέξεις εκτέλεση ενός classifier (description logic reasoner που βοηθά στον έλεγχο και στην οργάνωση μιας οντολογίας) μπορεί επίσης να συμπεράνει νέα γνώση, που μπορεί με τη σειρά της να παράγει πληροφορία που μπορεί να επωφεληθεί από περαιτέρω διάδραση με την SWRL, μια διαδικασία που μπορεί να χρειαστεί πολλές επαναλήψεις πριν τη δημιουργία μη επιθυμητής νέας γνώσης. Ο Pellet reasoner λαμβάνει υπόψιν του όλα τα σχετικά OWL axioms όταν πραγματοποιεί συμπερασματική γνώση με SWRL κανόνες, αλλά δεν μπορεί να δουλέψει με τις SWRL built-ins βιβλιοθήκες.

Συμπερασματικοί κανόνες

Καθώς η πιο σημαντική εφαρμογή των οντολογιών είναι ο σημασιολογικός ιστός, οι υπολογιστές θα πρέπει να έχουν πρόσβαση σε μία συλλογή πληροφοριών και ένα σύνολο από συμπερασματικούς κανόνες, με σκοπό την αυτόματη παραγωγή λογικής. Για τον λόγο αυτό, μια οντολογία αποτελείται από την ταξινομία (taxonomy), που καθορίζει τις κλάσεις των αντικειμένων και τις σχέσεις μεταξύ τους, και τους συμπερασματικούς κανόνες (inference rules) που παράγουν την λογική που διέπει το σύστημα. Στην πραγματικότητα, οι συμπερασματικοί κανόνες εξάγουν γνώση από την υπάρχουσα γνώση. Τέτοιες μηχανές συμπερασματολογίας είναι βασισμένες σε αλγόριθμους γενικευμένης λογικής (General logic based inference engines). Υπάρχουν δύο είδη λογικής:

- `Υψηλότερης τάξης λογική`, έχει τη μεγαλύτερη εκφραστική δυνατότητα αλλά δεν έχει καλή υπολογιστική ικανότητα. Απαρτίζεται από δύο πλευρές, μια υψηλότερης τάξης σύνταξη και μια υψηλότερης τάξης σημασιολογία
- `Λογική πρώτης τάξης`, λογική από την οποία δεν μπορούν να εξαχθούν ολοκληρωμένα συμπεράσματα (semi decidable), αλλά διαθέτει καλή υπολογιστική ικανότητα. Ωστόσο και αυτή, αντιμετωπίζει προβλήματα υπολογιστικής ικανότητας σε μεγάλες ποσότητες δεδομένων και αξιωμάτων

Η μηχανή συμπερασματικών κανόνων της OWL κάνει χρήση της περιγραφικής λογικής (Description Logics) ως γλώσσα αναπαράστασης γνώσης, η οποία χρησιμοποιεί ένα περιορισμένο σύνολο της λογικής πρώτης τάξης για την ιεραρχία της ορολογίας παρέχοντας ικανοποιητική υπολογιστική δυνατότητα. Όμως, η ικανότητα παραγωγής συμπερασμάτων είναι περιορισμένη μόνο στον έλεγχο της συνέπειας μιας οντολογίας.

Υποστήριξη SQWRL - SWRL

Τα SQWRL ερωτήματα, βασισμένα στην υπάρχουσα γνώση βάσης που περιέχεται στην OWL οντολογία, θα επιστρέψουν κάποιο αποτέλεσμα ή την ένδειξη ότι δεν δημιουργήθηκε κάποιο αποτέλεσμα αν δεν ικανοποιούνται οι συνθήκες του antecedent μέρους του κανόνα. Τα αποτελέσματα που παράγονται από ένα SQWRL ερώτημα δεν μπορούν να χρησιμοποιηθούν από τον χρήστη, άρα ούτε και να εισαχθούν ως νέα γνώση στην OWL οντολογία. Κάτι τέτοιο θα παραβίαζε την open world assumption και θα οδηγούσε σε non-monotonicity, μέθοδοι λογικής που δεν υποστηρίζονται από την OWL.

Σε αντίθεση με τα SQWRL ερωτήματα, οι SWRL κανόνες προσθέτουν την γνώση που προκύπτει από το consequent μέρος του κανόνα (εφόσον ικανοποιούνται πλήρως οι συνθήκες του antecedent μέρους), πίσω στην οντολογία. Ωστόσο, εξακολουθούν να δεσμεύονται από τα SWRL semantics, με αποτέλεσμα να μην υποστηρίζουν non-monotonicity, μορφές άρνησης ως διάψευση (negation as failure) και disjunction (λογικός τελεστής που επιστρέφει true εάν ένας από τους τελεστές του είναι αληθής). Επιπλέον, οι SWRL κανόνες είναι αδύνατο να ανακαλέσουν ή να αφαιρέσουν γνώση που υπάρχει στην βάση ή συμπεράθηκε από κάποιους SWRL κανόνες. Αυτό συνεπάγεται πως στην

περίπτωση που πρέπει να εισάγουν τιμή σε κάποιο πεδίο που φέρει ήδη μία ή περισσότερες τιμές, η νέα τιμή θα προστεθεί ως επιπλέον πληροφορία. Εάν πρόκειται για functional ιδιότητα, ένας OWL reasoner θα εντοπίσει την αντίφαση όταν πρόκειται να προστεθεί η τιμή πίσω στην ιδιότητα. Καθώς η SWRL είναι βασισμένη στην OWL DL, δεν υποστηρίζει συλλογιστική πάνω στην OWL Full και ούτε σε στιγμιότυπα ή σχέσεις που αφορούν RDF ή RDFS. Η αποσφαλμάτωση των SWRL κανόνων είναι πολύ δύσκολη αφού οι σχέσεις εξάρτησης των κανόνων μεταξύ τους και με την OWL οντολογία είναι ογκώδης και άβολη στον έλεγχο. Ο πιο προσιτός τρόπος είναι μέσω της διενέργειας SQWRL ερωτημάτων, κρατώντας σταθερό το antecedent μέρος και μεταβάλλοντας το consequent μέρος στο αντίστοιχο sqwrl:select build-in. Γενικά, η συγγραφή των SWRL κανόνων προσφέρει μεγαλύτερη εκφραστική δύναμη απ' ότι μπορεί να προσφέρει η OWL DL μόνη της, αλλά εις βάρος του decidability. Ωστόσο, υπάρχουν εφαρμογές που η λογική του decidability έχει περισσότερο θεωρητικό υπόβαθρο αντί πρακτικό. Εν τέλει, είναι προτιμότερη η χρησιμοποίηση των δυνατοτήτων της OWL, καθώς μόνο αν κρίνεται απολύτως αναγκαία η αξιοποίηση της περαιτέρω εκφραστικής δύναμης των κανόνων, θα πρέπει αυτοί να συγγραφούν.

1.3 Περαιτέρω Παρατηρήσεις

No Unique Names Assumption

Η έννοια του unique name assumption (UNA) προέρχεται από τις γλώσσες οντολογιών και τις γλώσσες αναπαράστασης γνώσης (description logics) και χρησιμοποιείται σε συστήματα βάσεων δεδομένων. Η υιοθέτηση αυτής της έννοιας υποθέτει ότι διαφορετικά ονόματα αναφέρονται πάντα σε διαφορετικές οντότητες σε ένα κόσμο. Η OWL δεν ακολουθεί αυτή την υπόθεση, αλλά παρέχει κατηγορηματικά δομικά στοιχεία που υποδηλώνουν αν δύο ονόματα αναφέρονται στην ίδια ή διαφορετική οντότητα. Παραδείγματος χάριν, αν δηλώσουμε ότι κάθε μάθημα διδάσκεται από το πολύ έναν διδάσκοντα, και έχουμε κάποιο μάθημα που διδάσκεται από δύο διδάσκοντες, ο OWL reasoner δε πρόκειται να εντοπίσει κάποιο λάθος. Αντιθέτως, θα συμπεράνει ότι οι δύο αυτοί διδάσκοντες αντιπροσωπεύουν την ίδια οντότητα (τα δύο resources είναι ίσα).

Data types στην OWL

Το XML Schema παρέχει ένα μηχανισμό για την κατασκευή data types καθορισμένα από ένα χρήστη για τις δικές του ανάγκες όπως π.χ. ένα data type ονόματι childAge που περιέχει όλους τους ακέραιους μικρότερους από το 18. Τέτοιας προέλευσης data types δε μπορούν να χρησιμοποιηθούν στην OWL, καθώς περιέχει μόνο τα συνηθέστερα και ευρέως αξιοποιήσιμα data types όπως string, integer, Boolean, time και date.

Property Chaining

Σε πολλές εφαρμογές το property chaining είναι μια χρήσιμη διαδικασία (π.χ. η δήλωση μιας ιδιότητας ως πρότυπο για τις μετέπειτα ιδιότητες) αλλά η OWL δεν επιτρέπει την σύμπλεξη των ιδιοτήτων λόγω αδυναμίας αποφασιστικότητας. Αυτό έγκειται στο γεγονός ότι συγκεκριμένα μόνο προβλήματα μπορούν να επιλυθούν μέσω αλγορίθμων και κάποια άλλα όχι. Ο καθορισμός της ύπαρξης ενός αποδοτικού και εφικτού τρόπου για την συμμετοχή ή όχι ενός τύπου σε μια εικασία αποκαλείται decidability. Η ενσωμάτωση αναπαράστασης γνώσης βασισμένη σε κανόνες και αναπαράστασης γνώσης σε Description Logics αποτελεί ακόμα ένα πεδίο έρευνας.

2. Web Ontology Language (OWL)

2.1 Εισαγωγή στην OWL

Η Web Ontology Language (OWL) είναι μια οικογένεια από γλώσσες αντιπροσώπευσης γνώσης για την δημιουργία οντολογιών. Οι γλώσσες χαρακτηρίζονται από την επίσημη σημασιολογία και τα βασισμένα στο RDF/XML serializations για το σημασιολογικό ιστό (Semantic Web). Η OWL επικυρώνεται από τον World Wide Web Consortium (W3C) και έχει προσελκύσει το ακαδημαϊκό, ιατρικό και εμπορικό ενδιαφέρον.

Τον Οκτώβριο του 2007, μια νέα ομάδα εργασίας του W3C άρχισε να επεκτείνει την OWL με διάφορα νέα χαρακτηριστικά όπως προτείνονται από την υποβολή μελών της OWL 1.1. Αυτή η νέα έκδοση, αποκαλούμενη OWL 2, σύντομα βρήκε το δρόμο της στους σημασιολογικούς συντάκτες όπως το Protégé και σημασιολογικούς reasoners όπως οι Pellet, RacerPro, FaCT++ και HermiT. Ο W3C ανακοίνωσε την νέα έκδοση στις 27 Οκτωβρίου 2009.

Η οικογένεια OWL περιέχει πολλά είδη, serializations, συντακτικά και προδιαγραφές με παρόμοια ονόματα. Αυτό μπορεί να προκαλεί σύγχυση εκτός και αν μια σταθερή και σύμφωνη προσέγγιση υιοθετηθεί. Οι OWL και OWL 2 θα χρησιμοποιηθούν ως σημείο αναφοράς στις προδιαγραφές του 2004 και του 2009, αντίστοιχα. Πλήρη ονόματα ειδών θα χρησιμοποιηθούν, συμπεριλαμβανομένης της έκδοσης προδιαγραφής (παραδείγματος χάριν, OWL2 EL). Σε περιπτώσεις γενικότερης αναφοράς, θα χρησιμοποιηθεί η OWL Family.

2.2 OWL και Semantic Web

Το Semantic Web αποτελεί ένα όραμα για το μέλλον του Web κατά το οποίο δίνεται στην πληροφορία λεπτομερής σημασιολογία, κάνοντας για τις μηχανές ευκολότερη την αυτόματη επεξεργασία και την ενσωμάτωση της πληροφορίας που υπάρχει διαθέσιμη στο Web. Το Semantic Web θα χτιστεί στην ικανότητα του XML να καθορίζει tagging schemes και στην ευελιξία του RDF για την αναπαράσταση των δεδομένων. Το πρώτο επίπεδο πάνω από το RDF που απαιτείται για το Semantic Web είναι μια γλώσσα οντολογίας που μπορεί να περιγράψει επίσημα την έννοια της ονοματολογίας που χρησιμοποιείται στα έγγραφα του διαδικτύου. Σε περίπτωση που οι μηχανές προβλέπεται να εκτελέσουν χρήσιμες συλλογιστικές εργασίες σε αυτά τα έγγραφα, η γλώσσα αυτή θα πρέπει να ξεπεράσει τα βασικά σημασιολογικά στοιχεία του RDF Schema.

Η OWL σχεδιάστηκε για να καλύψει την ανάγκη μιας τέτοιας γλώσσας οντολογίας διαδικτύου (Web Ontology Language) και αποτελεί μέρος της συνεχώς αναπτυσσόμενης στοίβας της σύστασης W3C που σχετίζεται με το Semantic Web. Σε αυτή περιέχονται:

- `XML`, παρέχει μια επιφανειακή σύνταξη για την δόμηση εγγράφων, αλλά δεν επιβάλλεται κανένας σημασιολογικός περιορισμός στην έννοια αυτών των εγγραφών
- `XML Schema`, είναι μια γλώσσα για τον περιορισμό στην δομή των XML εγγράφων και επίσης επεκτείνει την XML με datatypes
- `RDF`, είναι ένα μοντέλο δεδομένων για αντικείμενα (resources) και σχέσεις μεταξύ αυτών, το οποίο παρέχει ένα απλό σημασιολογικό αποτέλεσμα, με αυτά τα datamodels να μπορούν να αναπαρασταθούν σε σύνταξη XML
- `RDF Schema`, είναι ένα λεξιλόγιο για την περιγραφή των ιδιοτήτων και των κλάσεων των αντικειμένων του RDF, με σημασιολογία για την γενίκευση ιεραρχιών τέτοιων ιδιοτήτων και κλάσεων

Η OWL προσθέτει επιπλέον λεξιλόγιο για τη περιγραφή ιδιοτήτων και κλάσεων, συσχετίσεων μεταξύ κλάσεων (π.χ. disjointness), αριθμούς στοιχείων συνόλου (π.χ. ακριβώς ένα), ισότητες, πλουσιότερη γραφή ιδιοτήτων, χαρακτηριστικά γνωρίσματα ιδιοτήτων (π.χ. συμμετρία) και enumerated κλάσεις.

2.3 Τα 3 είδη OWL

Η OWL παρέχει τρείς επιμέρους αυξανόμενης εκφραστικότητας υπογλώσσες σχεδιασμένες για χρήση από συγκεκριμένες κοινότητες χρηστών για τις εφαρμογές τους:

- `OWL Lite`, για τους χρήστες που πρώτιστα χρειάζονται μια ιεραρχία ταξινόμησης και απλούς περιορισμούς. Παραδείγματος χάριν, ενώ υποστηρίζει περιορισμούς αριθμών στοιχείων συνόλου, επιτρέπονται μόνο τιμές αριθμών στοιχείων συνόλου 0 ή 1. Είναι απλούστερη η παροχή εργαλείων υποστήριξης για την OWL Lite σε σχέση με τους εκφραστικότερους συγγενείς της, ενώ παρέχει και ένα γρήγορο migration path για θησαυρούς και άλλες ταξονομίες. Επίσης, η OWL Lite έχει χαμηλότερη επίσημη πολυπλοκότητα σε σχέση με την OWL DL
- `OWL DL`, για τους χρήστες που θέλουν την μέγιστη εκφραστικότητα διατηρώντας την υπολογιστική πληρότητα (όλα τα συμπεράσματα είναι εγγυημένα να είναι υπολογίσιμα) και το decidability (όλοι οι υπολογισμοί θα τελειώσουν στον πεπερασμένο χρόνο). Η OWL DL περιλαμβάνει όλα τα OWL γλωσσικά κατασκευάσματα, αλλά μπορούν να χρησιμοποιηθούν μόνο βάση συγκεκριμένων περιορισμών (για παράδειγμα, ενώ μια κλάση μπορεί να είναι υποκλάση πολλών κλάσεων, μια κλάση δεν μπορεί να είναι στιγμιότυπο μια άλλης κλάσης). Η OWL DL ονομάζεται έτσι λόγω της αντιστοιχίας της με description logics, ένας τομέας έρευνας που έχει μελετήσει τις λογικές που διαμορφώνουν την επίσημη δομή της OWL
- `OWL Full`, για τους χρήστες που θέλουν την μέγιστη εκφραστικότητα και τη συντακτική ελευθερία του RDF χωρίς υπολογιστικές εγγυήσεις. Παραδείγματος χάριν, στο OWL Full μια κλάση μπορεί να αντιμετωπιστεί ταυτόχρονα ως μια συλλογή από άτομα (individuals) και ως άτομο αποκλειστικά ξεχωριστό. Η OWL Full επιτρέπει σε μια οντολογία να αυξήσει την έννοια του προκαθορισμένου RDF ή OWL λεξιλογίου. Είναι απίθανο για οποιοδήποτε λογισμικό συλλογισμού να είναι ικανό να υποστηρίξει πλήρης συλλογιστική για κάθε χαρακτηριστικό γνώρισμα της OWL Full

Κάθε μια από αυτές τις υπογλώσσες είναι μια επέκταση της απλούστερης προκατόχου της, με τις δύο θεμιτά σχετιζόμενες να έχουν και έγκυρη κατάληξη. Οι υπεύθυνοι για την ανάπτυξη οντολογιών που υιοθετούν την OWL θα πρέπει να εξετάσουν που υπογλώσσα ταιριάζει καλύτερα στις ανάγκες τους. Η επιλογή ανάμεσα στην OWL Lite και την OWL DL εξαρτάται από το εάν οι χρήστες κρίνουν αναγκαία την ύπαρξη περισσότερων εκφραστικών κατασκευασμάτων που παρέχονται από την OWL DL. Η επιλογή ανάμεσα στην OWL DL και στην OWL Full έγκειται κυρίως στο σημείο εάν οι χρηστές χρειάζονται μια μονάδα meta-modeling του RDF Schema (π.χ. καθορίζοντας κλάσεις των κλάσεων, ή συνδέοντας ιδιότητες στις κλάσεις). Όταν χρησιμοποιείται η OWL Full σε σχέση με την OWL DL, η υποστήριξη συλλογισμών είναι λιγότερο προβλεπόμενη αφού πλήρεις εφαρμογές OWL Full δεν υπάρχουν μέχρι στιγμής.

Η OWL Full μπορεί να επισημανθεί ως μια επέκταση του RDF , ενώ η OWL Lite και OWL DL μπορούν να επισημανθούν ως επεκτάσεις μιας περιορισμένης οπτικής γωνίας του RDF. Κάθε OWL

έγγραφο, βασισμένο σε μια από αυτές τις υπογλώσσες, και κάθε RDF έγγραφο είναι ένα OWL Full έγγραφο, αλλά μόνο κάποια RDF έγγραφα μπορούν να είναι θεμιτά OWL Lite ή OWL DL έγγραφα. Εξαιτίας αυτού, πρέπει να υπάρχει μια προσεκτικότητα όταν ο χρήστης θέλει να μετατρέψει ένα RDF έγγραφο σε OWL. Όταν η εκφραστικότητα της OWL DL ή της OWL Lite θεωρείται κατάλληλη, κάποιες επιφυλάξεις πρέπει να παρθούν ώστε να εγγυηθεί ότι το αυθεντικό RDF έγγραφο συμπάσχει με τους επιπλέον περιορισμούς που καθορίζονται από τις OWL DL και OWL Lite. Ανάμεσα σε αυτές, κάθε URI που χρησιμοποιείται ως όνομα κλάσης πρέπει ρητά να αποδοθεί ως τύπος owl:Class (παρομοίως για τις ιδιότητες), κάθε άτομο πρέπει να διευκρινιστεί ότι ανήκει σε μια τουλάχιστον κλάση (ακόμα και αν είναι η owl:Thing), τα URI που χρησιμοποιούνται για τις κλάσεις, τις ιδιότητες και τα άτομα πρέπει να είναι αμοιβαία ανεξάρτητα (disjoint).

2.4 Βασικά στοιχεία

Τα περισσότερα στοιχεία μιας OWL οντολογίας αφορούν κλάσεις, ιδιότητες, στιγμιότυπα αυτών των κλάσεων, και συσχετίσεις μεταξύ αυτών των στιγμιότυπων.

Υπάρχουν περιπτώσεις κατά τις οποίες πολλές χρήσεις μιας οντολογίας θα εξαρτώνται από την ικανότητα συλλογισμού γύρω από τα άτομα. Για να το καταφέρουμε αυτό με ένα εύχρηστο τρόπο θα πρέπει να έχουμε ένα μηχανισμό για την περιγραφή των κλάσεων που ανήκουν τα άτομα και των ιδιοτήτων που κληρονομούν από την χρηστότητα των κλάσεων που συμμετέχουν. Μπορούμε πάντα να παρέμβουμε σε συγκεκριμένες ιδιότητες που αφορούν τα άτομα, αλλά η μεγαλύτερη δύναμη των οντολογιών προέρχεται από τον συλλογισμό βασισμένο στις κλάσεις.

Όμως ένας κόσμος κλάσεων και ατόμων δε θα προξενούσε κάποιο ιδιαίτερο ενδιαφέρον αν το μόνο που μπορούσε να προσφέρει είναι ο καθορισμός ταξινομιών. Μέσω των ιδιοτήτων μπορούμε να επιβάλλουμε γενικά γεγονότα για τα μέλη των κλάσεων και ειδικά γεγονότα για τα άτομα.

2.4.1 Κλάσεις (Classes)

Τα βασικότερα συστατικά μέρη ενός τομέα θα πρέπει να αφορούν τις κλάσεις που αποτελούν τις ρίζες πολλών ταξινομικών δέντρων. Οποιοδήποτε άτομο σε ένα κόσμο OWL είναι μέλος της κλάσης owl:Thing. Κατ' αυτό τον τρόπο κάθε δηλωμένη κλάση από ένα χρήστη είναι ανεπιφύλακτα υποκλάση της owl:Thing. Συγκεκριμένες root κλάσεις ενός τομέα καθορίζονται απλά δηλώνοντας ένα όνομα για τις κλάσεις αυτές. Η OWL επίσης καθορίζει και την άδεια κλάση, owl:Nothing.

Ο ορισμός της υποκλάσης σχετίζει μια συγκεκριμένη κλάση σε μια περισσότερο γενική κλάση. Αν X είναι μα υποκλάση του Y, τότε κάθε στιγμιότυπο της X είναι επίσης και στιγμιότυπο της Y. Επίσης, η σχέση της υποκλάσης είναι μεταβατική, που σημαίνει ότι αν X είναι μια υποκλάση της Y και Y μια υποκλάση της Z τότε η X είναι και υποκλάση της Z.

Η δήλωση μιας κλάσης έχει δύο μέρη: ένα εισαγωγικό όνομα ή παραπομπή και μια λίστα με περιορισμούς. Κάθε μία από τις άμεσα περιεχόμενες εκφράσεις στην καθορισμένη κλάση περιορίζει παραπέρα τα στιγμιότυπα της δηλωθέντας κλάσης. Τα στιγμιότυπα της κλάσης ανήκουν στην τομή των περιορισμών, με τις περισσότερες οντολογίες να έχουν τουλάχιστον ένα περιορισμό, που αναγκάζει μια καινούργια κλάση να είναι υποκλάση κάποιας άλλης επονομαζόμενης κλάσης.

Αν και οι παραπάνω κλάσεις είναι περιορισμένες και μπορούν να χαρακτηρίσουν μόνο μια ατελής οντολογία, οι πληροφορίες που παρέχονται είναι και πάλι αρκετές για τη δημιουργία και το συλλογισμό ατόμων.

2.4.2 Άτομα (Individuals)

Σε αντίθεση με τις κλάσεις, θέλουμε να είναι δυνατή και η περιγραφή των μελών αυτών των στοιχείων. Στο δικό μας κόσμο πραγμάτων θεωρούμε αυτά τα στοιχεία ως άτομα. Ένα άτομο αποκτάει ύπαρξη μέσω της δήλωσής του ως μέλος σε μια κλάση.

Επί το πλείστον, δημιουργούνται προβλήματα σχετικά με την διάκριση ανάμεσα σε μια κλάση και σε ένα άτομο στην OWL. Μια κλάση είναι απλά ένα όνομα και μια συλλογή από ιδιότητες που περιγράφουν συσχετίσεις ανάμεσα σε ένα σύνολο ατόμων. Τα άτομα είναι τα μέλη αυτού του

συνόλου. Εξ' αυτών, προκύπτει ότι οι κλάσεις θα πρέπει να αντιστοιχούν στα σύνολα των πραγμάτων του τομέα αναφοράς, και τα άτομα θα πρέπει να αντιστοιχούν στις πραγματικές οντότητες που μπορούν να ομαδοποιηθούν σε αυτές τις κλάσεις.

2.4.3 Ιδιότητες (Properties)

Μια ιδιότητα είναι μια σχέση που αναφέρεται σε ζεύγη. Διακρίνονται δύο τύποι ιδιοτήτων, οι datatype properties που αφορούν σχέσεις ανάμεσα σε στιγμιότυπα κλάσεων και RDF literals και XML Schema datatypes, και οι object properties που αφορούν σχέσεις μεταξύ στιγμιότυπων δύο κλάσεων.

Κατά τη δήλωση μια ιδιότητας χρησιμοποιείται μια μέθοδος για τον περιορισμό της σχέσης, η οποία καθορίζεται μέσω του domain και του range. Μια ιδιότητα μπορεί να αποτελεί εξειδίκευση (subproperty) μιας άλλης υπάρχουσας ιδιότητας.

Μια ιδιότητα X που έχει ως domain την κλάση Y και ως range την κλάση Z, έχει ως αποτέλεσμα να συσχετίσει τα στιγμιότυπα της κλάσης Y με τα στιγμιότυπα της κλάσης Z. Πολλαπλά domains σημαίνουν ότι το domain της ιδιότητας αποτελεί την τομή των αναφερόμενων κλάσεων (ομοίως και για το range).

Αξιοσημείωτο είναι πως η χρήση των πληροφοριών στα domain και range στην OWL είναι διαφορετική σε σχέση με τον τύπο των πληροφοριών μιας γλώσσας προγραμματισμού. Ανάμεσα σε άλλα, τέτοιοι τύποι χρησιμοποιούνται για τον έλεγχο της συνεκτικότητας σε μια γλώσσα προγραμματισμού. Παραδείγματος χάριν, στην OWL ένα range μπορεί να χρησιμοποιηθεί για να συμπεράνει ένα τέτοιο τύπο.

2.4.4 Datatypes

Προηγουμένως ξεχωρίσαμε τις ιδιότητες ανάλογα με το αν συσχετίζουν άτομα με άτομα (object properties) ή αν συσχετίζουν άτομα με datatypes (datatype properties). Οι datatype properties μπορούν να ποικίλουν από RDF literals σε απλούς τύπους που καθορίζονται βάση του XML Schema dataypes. Παρόλο που η OWL χρησιμοποιεί τα περισσότερα built-in XML Schema datatypes, συνιστώνται ανεπιφύλακτα μόνο τα παρακάτω, συμπεριλαμβανομένου και του RDF literal:

xsd:string	xsd:normalizedString	xsd:boolean	xsd:string
xsd:decimal	xsd:float	xsd:double	xsd:decimal
xsd:integer	xsd:nonNegativeInteger	xsd:positiveInteger	xsd:integer
xsd:nonPositiveInteger	xsd:negativeInteger	xsd:NMTOKEN	xsd:nonPositiveInteger
xsd:long	xsd:int	xsd:short	xsd:long
xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort	xsd:unsignedLong
xsd:hexBinary	xsd:base64Binary	xsd:anyURI	xsd:hexBinary
xsd:dateTime	xsd:time	xsd:date	xsd:dateTime
xsd:gYear	xsd:gMonthDay	xsd:gDay	xsd:gYear
xsd:anyURI	xsd:token	xsd:language	
xsd:NMTOKEN	xsd:Name	xsd:NCName	rdfs:Literal

Πίνακας 1. Datatypes που συγκροτούν τα built-in OWL Datatypes

2.4.5 Χαρακτηριστικά ιδιοτήτων

Ακολουθεί περιγραφή των μηχανισμών που χρησιμοποιούνται για την περαιτέρω ειδίκευση των ιδιοτήτων. Μέσω αυτών, είναι δυνατό να καθορίσουμε χαρακτηριστικά στις ιδιότητες, τα οποία παρέχουν ένα ισχυρό μηχανισμό για τον συλλογισμό της ιδιότητας. Τα χαρακτηριστικά που μπορεί να έχει μια ιδιότητα είναι:

- `TransitiveProperty`, στην περίπτωση που μια ιδιότητα P καθοριστεί ως μεταβατική τότε για κάθε x, y και z θα ισχύει πάντα ότι αν P(x, y) και P(y, z) υπονοείται ότι P(x, z)

- `SymmetricProperty`, στην περίπτωση που μια ιδιότητα P καθοριστεί ως συμμετρική τότε για κάθε x και y θα ισχύει πάντα ότι P(x, y) αν και μόνο αν P(y, x)
- `FunctionalProperty`, στην περίπτωση που μια ιδιότητα P καθοριστεί ως λειτουργική τότε για όλα τα x, y και z θα ισχύει πάντα ότι αν P(x, y) και P(x, z) υπονοείται ότι y = z
- `inverseOf`, στην περίπτωση που μια ιδιότητα P_1 καθοριστεί ως αντίστροφη μιας P_2 τότε για όλα τα x και y θα ισχύει πάντα ότι $P_1(x, y)$ αν και μόνο αν $P_2(y, x)$
- `inverseFunctionalProperty`, στην περίπτωση που μια ιδιότητα P καθοριστεί ως αντίστροφα λειτουργική τότε για όλα τα x, y, z θα ισχύει πάντα ότι αν P(y, x) και P(z, x) υπονοείται ότι y = z

2.4.6 Περιορισμοί ιδιοτήτων

Αντί να καθορίσουμε χαρακτηριστικά στις ιδιότητες, είναι δυνατό να περιορίσουμε περαιτέρω το range μιας ιδιότητας σε συγκεκριμένα πλαίσια με μια ποικιλία διαφορετικών τρόπων. Αυτό είναι εφικτό μέσω των περιορισμών των ιδιοτήτων. Όλοι αυτοί οι περιορισμοί γίνονται μέσα στα πλαίσια του owl:Restriction και είναι οι παρακάτω:

- `allValuesFrom`, ο συγκεκριμένος περιορισμός καθορίζει ότι για κάθε στιγμιότυπο μιας κλάσης που έχει στιγμιότυπα μιας συγκεκριμένης ιδιότητας, όλες οι δυνατές τιμές αυτής της ιδιότητας αποτελούν μέλη της κλάσης που προσδιορίζεται από το owl:allValuesFrom. Πρόκειται για τοπικό μόνο περιορισμό, δηλαδή εφαρμόζεται ξεχωριστά σε κάθε ιδιότητα
- `someValuesFrom`, ο συγκεκριμένος περιορισμός καθορίζει ότι για κάθε στιγμιότυπο μιας κλάσης που έχει στιγμιότυπα μιας συγκεκριμένης ιδιότητας, η ιδιότητα αυτή θα έχει ως τιμή τουλάχιστον ένα ή περισσότερα μέλη της κλάσης που προσδιορίζεται από το owl:allValuesFrom. Πρόκειται για τοπικό μόνο περιορισμό, δηλαδή εφαρμόζεται ξεχωριστά σε κάθε ιδιότητα
- `Cardinality`, ο συγκεκριμένος περιορισμός προσδίδει ισχυρούς καθορισμούς για ένα αριθμό στοιχείων σε ένα σύνολο ή και ακριβώς ένα καθορισμό στοιχείου από αυτό το σύνολο. Συνηθίζεται να χρησιμοποιείται σε functional ιδιότητες επιβάλλοντας συγκεκριμενοποίηση του στοιχείου. Οι εκφράσεις cardinality που περιέχουν τιμές περιορίζονται στις 0 και 1, και αποτελούν μέρος του OWL Lite. Αυτό επιτρέπει στο χρήστες να επισημαίνουν εκφράσεις που αντιστοιχούν στα λογικά “τουλάχιστον ένα”, “όχι περισσότερο από ένα” και “ακριβώς ένα”. Θετικές ακέραιες τιμές εκτός από το 0 και 1 επιτρέπονται στην OWL DL. Αξιοποιώντας το owl:maxCardinality μπορούμε να καθορίσουμε ένα ανώτατο επιτρεπόμενο όριο, ενώ με το owl:minCardinality μπορούμε να καθορίσουμε το κατώτερο επιτρεπόμενο όριο. Συνδυασμός αυτών το δύο εκφράσεων οδηγεί στην δημιουργία αριθμητικού ορίου στην εκάστοτε ιδιότητα
- `hasValue`, χρησιμοποιείται για να καθορίσουμε κλάσεις που είναι βασισμένες στην ύπαρξη μιας συγκεκριμένης τιμής μιας ιδιότητας. Οπότε, ένα άτομο θα είναι μέλος μιας τέτοιας κλάσης αν οποιαδήποτε στιγμή ισχύει ότι τουλάχιστον μια από τις τιμές αυτής της ιδιότητάς είναι ίση με την τιμή του hasValue. Πρόκειται για τοπικό μόνο περιορισμό, δηλαδή εφαρμόζεται ξεχωριστά σε κάθε ιδιότητα

2.5 Ontology Mapping

Για να μπορέσουν οι οντολογίες να έχουν όσο το δυνατόν μεγαλύτερο αντίκτυπο, θα πρέπει να είναι ευρέως διαμοιρασμένες. Παράλληλα, για να ελαχιστοποιήσουμε την πνευματική προσπάθεια που έχει ως προϋπόθεση την ανάπτυξη μιας οντολογίας χρειάζεται να είναι ικανή η επαναχρησιμοποίησή της. Η καλύτερη δυνατή περίπτωση θα αποτελούσε η σύνθεσή της. Για παράδειγμα, θα μπορούσαμε να υιοθετήσουμε από μια πηγή μια οντολογία που φέρει τυπικές χρονολογίες και από μια άλλη πηγή μια δεύτερη οντολογία που φέρει υπαρκτές τοποθεσίες. Μετέπειτα, να επεκτείνουμε την αντίληψη της τοποθεσίας ώστε να περιλαμβάνει την χρονική περίοδο που λαμβάνει μέρος κάτι.

Είναι σημαντικό να συνειδητοποιήσουμε ότι μεγάλο μέρος της προσπάθειας της ανάπτυξης μιας οντολογίας αφιερώνεται στην συγκέντρωση των κλάσεων και των ιδιοτήτων με τρόπους που

μεγιστοποιούν την συνυπαιτιότητα. Θέλουμε απλούς ισχυρισμούς για τις συμμετέχοντες κλάσεις ώστε να έχουμε ευρείς και χρήσιμες συνεκδοχές. Αυτό είναι το δυσκολότερο μέρος στην ανάπτυξη της οντολογίας. Αν είναι δυνατή η εύρεση μιας υπάρχουσας οντολογίας που έχει ήδη εκτεταμένη λειτουργία και είναι σε κάποιο βαθμό τελειοποιημένη, τότε είναι απόλυτα λογικό να την υιοθετήσουμε.

Παρόλα αυτά, εξακολουθεί να συνιστά πρόκληση η ένωση μιας συλλογής από οντότητες καθώς κρίνεται αναγκαία η διατήρηση της συνεκτικότητάς της μέσω εργαλείων υποστήριξης.

2.5.1 Ισοδυναμίες ανάμεσα σε κλάσεις και ιδιότητες

Για να συνδέσουμε μαζί σύνολα συστατικών στοιχείων από δύο διαφορετικές οντολογίες ως μέρος μας τρίτης, είναι συνήθως αναγκαία η δυνατότητα ένδειξης κάποιας κλάσης ή ιδιότητας της πρώτης οντολογίας ως ισοδύναμη κάποιας κλάσης ή ιδιότητας μιας δεύτερης οντολογίας. Αυτή η ικανότητα πρέπει να χρησιμοποιείται με σύνεση. Αν οι συνδεόμενες οντολογίες είναι αντιφατικές (όλα τα A είναι και B σε αντιπαράθεση με όλα τα A δεν είναι και B) δεν θα μπορέσει να υπάρξει κάποια επέκταση (σε άτομα και σχέσεις) που να ικανοποιήσει το αποτέλεσμα αυτού του συνδυασμού.

Η ιδιότητα `owl:equivalentClass` χρησιμοποιείται για να δείξει ότι δύο κλάσεις έχουν ακριβώς ίδια στιγμιότυπα. Για να δέσουμε μαζί ιδιότητες χρησιμοποιείται ομοίως η `owl:equivalentProperty`. Αξιοσημείωτο πάντως, είναι πως στην OWL DL, οι κλάσεις απλά υποδηλώνουν σύνολα από άτομα αλλά δεν αποτελούν άτομα καθαυτές. Στην OWL Full, παρόλα αυτά, μπορούμε να χρησιμοποιήσουμε την owl:sameAs ανάμεσα σε δύο κλάσεις για να δείξουμε ότι είναι πανομοιότυπες σε οτιδήποτε.

2.5.2 Ταυτοποίηση ανάμεσα σε άτομα

Ο μηχανισμός παραμένει ο ίδιος με αυτός των κλάσεων, με τη διαφορά ότι εδώ δηλώνονται δύο άτομα να είναι πανομοιότυπα. Αυτό φέρνει στο προσκήνιο ένα σημαντικό ζήτημα. Βάση κοινής λογικής με unique name assumption, διαφορετικά ονόματα πάντα αναφέρονται σε διαφορετικές οντότητες σε ένα κόσμο. Η OWL δεν κάνει αυτή την υπόθεση, αλλά διευκρινίζει μέσω κατηγορηματικών εκφράσεων για το αν δύο ονόματα αφορούν την ίδια ή διαφορετική οντότητα. Για αυτόν τον λόγο, έχοντας δύο διαφορετικά ονόματα δεν είναι απαραίτητο ότι αναφέρονται και σε διαφορετικά άτομα.

Χρησιμοποιώντας την `sameAs` για να εξισώσουμε δύο κλάσεις δεν είναι το ίδιο με το να τις εξισώσουμε χρησιμοποιώντας την equivalentClass. Αντίθετα, αυτό που κάνει είναι να προκαλέσει τις κλάσεις να ερμηνευτούν ως άτομα με αποτέλεσμα να έχουμε την δυνατότητα να κατηγοριοποιήσουμε την οντολογία μας σαν OWL Full. Στην OWL Full η sameAs μπορεί να χρησιμοποιηθεί για να εξισώσει οτιδήποτε: μια κλάση και ένα άτομο, μια ιδιότητα και μια κλάση, κ.τ.λ., και προκαλεί και τα δύο κατηγορήματα να ερμηνευτούν ως άτομα.

2.5.3 Διαφοροποίηση ατόμων

Πρόκειται για το μηχανισμό `owl:differentFrom` που παρέχει τα αντίθετα αποτελέσματα από την sameAs, δηλαδή καθορίζει ότι οι τιμές του είναι αμοιβαία ξεχωριστές. Χάρις σε αυτούς τους ισχυρισμούς καλύπτουμε περιπτώσεις κατά τις οποίες είναι σημαντικό να διασφαλιστεί η διαφοροποίηση οντοτήτων, ώστε να αποφευχθούν αντικρούσεις.

Όμως, υπάρχει και ένας πιο βολικός μηχανισμός για τον καθορισμό ενός συνόλου από αμοιβαία ξεχωριστά μεταξύ τους άτομα. Χρησιμοποιώντας τον ισχυρισμό `owl:AllDifferent` εμπεριέχουμε ένα πλήθος από άτομα που καθορίζονται ότι είναι όλα ξεχωριστά. Αυτό επιτυγχάνεται αξιοποιώντας το owl:distinctMembers σε συνδυασμό με το owl:AllDiferent. Παρόλα αυτά, σε περίπτωση προσθήκης επιπλέον οντολογίας και υποθέτοντας ότι είναι αποσπασμένη από τα ήδη δηλωμένα άτομα, κρίνεται αναγκαία η επέμβαση στον κώδικα (κόβοντας και επικολλώντας τον αρχικό συσχετισμό owl:AllDifferent και προσθέτοντας τα νέα κατασκευάσματα στη λίστα). Δυστυχώς δεν υπάρχει απλούστερος τρόπος για να επεκτείνουμε μια συλλογή της owl:AllDifferent στην OWL

DL. Στην OWL Full, χρησιμοποιώντας RDF τριπλέτες και τα στοιχεία rdf:List μπορούν να προσδιοριστούν εναλλακτικές λύσεις.

2.6 Σύνθετες κλάσεις (OWL DL)

Η OWL παρέχει επιπλέον constructors για το σχηματισμό κλάσεων. Αυτοί οι constructors χρησιμοποιούνται για την δημιουργία των αποκαλούμενων class expressions. Η OWL υποστηρίζει το βασικό σύνολο λειτουργιών, τα οποία είναι ονομαστικά η ένωση (union), η τομή (intersection) και το συμπλήρωμα (complement). Αυτές υλοποιούνται μέσω των owl:unionOf, owl:intersectionOf και owl:complementOf αντίστοιχα. Επιπλέον, οι κλάσεις μπορούν να αριθμηθούν αλλά και οποιαδήποτε class extension μπορεί να αποδοθεί ρητά μέσω του oneOf constructor. Επίσης είναι δυνατό να επιβληθεί η αποσπασματικότητα (disjoint) αυτών των class extensions.

Άξιο αναφοράς είναι ότι οι class expressions μπορούν να είναι ενθυλακωμένες χωρίς να απαιτείται η δημιουργία ονομάτων για κάθε ενδιάμεση κλάση. Αυτό επιτρέπει την χρήση των λειτουργιών πάνω στα σύνολα για το χτίσιμο σύνθετων κλάσεων από ανώνυμες κλάσεις ή κλάσεις με περιορισμούς στις τιμές τους.

2.6.1 Set Operators

Όπως προαναφέραμε οι OWL class extensions είναι σύνολα αποτελούμενα από άτομα που είναι μέλη κάποιας κλάσης. Η OWL παρέχει τα μέσα για τον χειρισμό των class extensions βάση λογικών set operators, οι οποίοι αναφέρονται παρακάτω:

- `Intersection`, οι κλάσεις που κατασκευάζονται χρησιμοποιώντας set operations μοιάζουν περισσότερο με προσδιορισμούς. Τα μέλη της κλάσης καθορίζονται απόλυτα από ένα set operation. Παραδείγματος χάριν, μια κλάση Α μπορεί να θεωρηθεί ως η τομή μιας κλάσης Β και ενός συνόλου πραγμάτων που φέρουν μια συγκεκριμένη τιμή σε μια ιδιότητα. Αυτό παράλληλα, συνεπάγεται στο γεγονός ότι αν κάτι φέρει αυτή την τιμή και είναι άτομο της κλάσης Β, αυτόματα θα αποτελεί και στιγμιότυπο της κλάσης Α. Χωρίς τον καθορισμό της τομής θα ίσχυε αποκλειστικά μόνο το πρώτο μέρος και όχι το αντίστροφο. Περαιτέρω, αποτελεί και ένα σημαντικό εργαλείο για την κατηγοριοποίηση των ατόμων
- `Union`, ακολουθεί την ίδια λογική με το intersectionOf με τη διαφορά ότι προσδιορίζει την ένωση. Παραδείγματος χάριν, μια κλάση Α είναι η ένωση δύο class extensions Β και Γ, εφόσον η τομή της Β και της Γ θα ήταν το κενό
- `Complement`, επιλέγει όλα τα άτομα από τον τομέα αναφοράς που δεν ανήκουν σε κάποια συγκεκριμένη κλάση. Αυτός ο set operator δεν έχει ευρεία χρήση αλλά συνήθως χρησιμοποιείται σε μεγάλο πλήθος ατόμων και σε συνδυασμό με άλλους set operators. Παραδείγματος χάριν, μια κλάση Α θα μπορούσε να ήταν η τομή μιας κλάσης Β με ένα σύνολο από πράγματα (άτομα προσδιοριζόμενα μέσω ιδιοτήτων) που δεν υπάρχουν σε μια κλάση Γ

2.6.2 Enumerated Classes

Η OWL παρέχει τα μέσα για τον καθορισμό μιας κλάσης διαμέσου μιας ευθύς καταμέτρησης των μελών της. Αυτό είναι δυνατό με την αξιοποίηση του `oneOf` construct. Αυτός ο προσδιορισμός καθορίζει απόλυτα την class extension, έτσι ώστε κανένα άλλο επιπλέον άτομο δεν μπορεί να δηλωθεί ότι θα ανήκει σε αυτή την κλάση. Κάθε στοιχείο του oneOf πρέπει να είναι ένα ορθά δηλωμένο άτομο και κάθε άτομο θα πρέπει να ανήκει σε κάποια κλάση. Το κομμάτι που απαιτεί την μέγιστη προσοχή μας, είναι ότι κανένα επιπλέον άτομο δε μπορεί να είναι έγκυρο μέλος μιας κλάσης εφόσον δεν εμπεριέχεται στην καταμέτρηση.

2.6.3 Disjoint Classes

Για τον καθορισμό ενός συνόλου από κλάσεις που δεν έχουν κανένα στοιχείο τους κοινό χρησιμοποιείται ο constructor `owl:disjointWith`. Αυτός εγγυάται ότι κανένα άτομο που αποτελεί μέλος μιας κλάσης δε μπορεί την ίδια στιγμή να αποτελεί και στιγμιότυπο μιας άλλης κλάσης. Μια τυπική απαίτηση που ικανοποιεί αυτός ο constructor είναι ο καθορισμός μιας κλάσης ως η ένωση δύο disjoint υποκλάσεων. Όμως, όσο αυξάνει ο αριθμός των disjoint κλάσεων, αυξάνει και ο αριθμός των ισχυρισμών αναλογικά κατά n2. Παρόλα αυτά, σε αναλύσεις και ελέγχουν που έχουν γίνει, το n είναι τυπικά μικρό. Όταν το n είναι μεγάλο, εναλλακτικές προσεγγίσεις μπορούν να χρησιμοποιηθούν για την αποφυγή ανάπτυξης τετραγώνων στον αριθμό των ισχυρισμών.

Μια μέθοδος επίλυσης αυτού του προβλήματος, που αναφέρεται και στο OWL test suite, αποτελεί η ακόλουθη. Περιγράφουμε μια γονική κλάση της οποίας τα στοιχεία έχουν μια ιδιότητα με cardinality ίσο με ένα. Αυτό σημαίνει ότι κάθε στιγμιότυπο πρέπει να έχει μία και μόνο μία τιμή για αυτήν την ιδιότητα. Μετά, για κάθε υποκλάση του γονέα απαιτείται τα στιγμιότυπά της να έχουν μια συγκεκριμένη μοναδική τιμή για την ιδιότητα. Οπότε σε καμία περίπτωση οι ξεχωριστές υποκλάσεις δε μπορούν να έχουν κοινά μέλη.

2.7 Έκδοση Οντολογίας

Οι οντολογίες είναι σαν ένα λογισμικό, κάποια στιγμή θα χρειαστούν συντήρηση, πράγμα που θα οδηγήσει στην αλλαγή τους. Μέσα στο στοιχείο owl:Ontology, μπορεί να περιέχεται ένας σύνδεσμος που οδηγεί σε προηγούμενη (άρα και παλιότερη) έκδοση της οντολογίας που αναφέρεται. Η ιδιότητα `owl:priorVersion` είναι υπεύθυνη για την παροχή αυτού του συνδέσμου, και μπορεί να χρησιμοποιηθεί για τον εντοπισμό του ιστορικού των εκδόσεων μιας οντολογίας.

Οι εκδόσεις μιας οντολογίας μπορεί να είναι ασύμβατες μεταξύ τους. Παραδείγματος χάριν, μιας προγενέστερη έκδοση μιας οντολογίας μπορεί να περιέχει δηλώσεις οι οποίες αντικρούονται με την τρέχουσα έκδοση. Εξαιτίας αυτού, μέσα στο στοιχείο owl:Ontology, χρησιμοποιούμε τις ετικέτες `owl:backwardCompatibleWith` και `owl:incompatibleWith` για να υποδηλώσουμε την συμβατότητα ή την έλλειψη αυτής με πρότερες εκδόσεις. Στην περίπτωση που δεν δηλωθεί το στοιχείο owl:backwardCompatibleWith, τότε η συμβατότητα δεν νοείται και αυτομάτως δεν θα πρέπει να ληφθεί υπόψη μας. Σε αντιπαράθεση με τις προηγούμενες τρείς ετικέτες, το owl:versionInfo περιέχει ένα αλφαριθμητικό που παρέχει πληροφορίες για την τρέχουσα έκδοση και η ετικέτα αυτή να μπορεί να χρησιμοποιηθεί για την επισήμανση κλάσεων και ιδιοτήτων σε σχέση με τις οντολογίες.

Για πολλούς λόγους, κάνοντας συνεχείς και αναλυτικές ανιχνεύσεις εκδόσεων ολόκληρης της οντολογίας δεν είναι αρκετό. Ακόμα και αν οι συντηρητές επιθυμούν να κρατήσουν τις πληροφορίες μιας έκδοσης γύρω από τις κλάσεις, τις ιδιότητες και τα άτομα της – και πάλι αυτό μπορεί να μην είναι επαρκές. Η φυσική τάση αύξησης των class expressions στην OWL μπορεί να οδηγήσει σιωπηρά μια οντολογία στην προσθήκη περιορισμών σε κάποια κλάση που καθορίζεται από μια άλλη οντολογία, και αυτοί οι επιπλέον περιορισμοί να απαιτούν πληροφορία διαφορετικής έκδοσης.

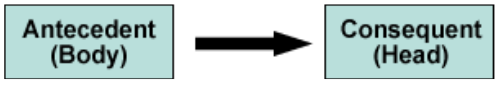
Η OWL Full παρέχει την εκφραστική δύναμη να κάνει οποιοδήποτε τύπο ισχυρισμού σε μια κλάση, π.χ. ότι είναι στιγμιότυπο μιας άλλης κλάσης, ή ότι καθαυτή η κλάση έχει μια ιδιότητα και μια τιμή για αυτή την ιδιότητα. Αυτός ο τρόπος μπορεί να χρησιμοποιηθεί για να χτιστεί μια οντολογία από κλάσεις και ιδιότητες για τον εντοπισμό της πληροφορίας μιας έκδοσης. Η OWL namespace περιέχει δύο προκαθορισμένες κλάσεις που μπορούν να χρησιμοποιηθούν για αυτόν τον σκοπό: η owl:DeprecatedClass και η owl:DeprecatedProperty. Ωστόσο, έχει ήδη διασαφηνιστεί ότι αυτές οι κλάσεις είναι πολύ πιθανό να έχουν ασύμφωνη συμπεριφορά σε επερχόμενη έκδοση. Ακόμα περαιτέρω, δεν φέρουν κανένα επιπλέον σημασιολογικό χαρακτηριστικό και είναι αποκλειστικά επιλογή και ευθύνη των tool developers και των OWL χρηστών η διασφάλιση της λειτουργίας τους όπως αυτή έχει προοριστεί.

3. SWRL - SQWRL

3.1 Εισαγωγή στην SWRL

Η SWRL(Semantic Web Rule Language) αποτελεί μια πρόταση που τείνει να γίνει η γλώσσα κανόνων του Semantic Web και είναι βασισμένη στον συνδυασμό των OWL DL και OWL Lite υπογλωσσών της OWL με τις Unary/Binary Datalog RuleML υπογλώσσες της Rule Markup Language. Αυτή η πρόταση επεκτείνει το σύνολο των OWL axioms με την εισαγωγή Horn-like rules. Με αυτόν τον τρόπο επιτρέπει τον συνδυασμό των Horn-like rules με μια βάση δεδομένων OWL. Παράλληλα, παρέχεται μια υψηλού επιπέδου αφηρημένη σύνταξη για τους Horn-like κανόνες, οι οποίοι εκφράζονται σε έννοιες της γλώσσας OWL (κλάσεις, ιδιότητες, άτομα).

Οι κανόνες έχουν την μορφή της συνυπαιτιότητας μεταξύ ενός antecedent και ενός consequent. Το antecedent μέρος αποτελεί το κυρίως σώμα ενός κανόνα και είναι γνωστό ως body, ενώ το consequent μέρος αποτελεί την συνεπαγωγή του κανόνα και είναι γνωστό ως head. Το νόημα στο οποίο αποσκοπεί ένας κανόνας μπορεί να ερμηνευθεί ως εξής: οποτεδήποτε ισχύουν οι συνθήκες που έχουν καθοριστεί στο antecedent μέρος ενός κανόνα, τότε πρέπει να ισχύουν και οι συνθήκες που έχουν καθοριστεί στο consequent μέρος αυτού του κανόνα.



Εικόνα 1. SWRL Rule Format

Τα antecedent και consequent μέρη αποτελούνται από μηδέν ή περισσότερα atoms. Ένα κενό antecedent μέρος αντιμετωπίζεται ως αναξιόλογα αληθές καθώς ικανοποιείται από οποιαδήποτε

αποδιδόμενη έννοια, οπότε το consequent μέρος πρέπει επίσης να ικανοποιείται από οποιαδήποτε αποδιδόμενη έννοια. Ένα άδειο consequent μέρος αντιμετωπίζεται ως αναξιόλογα ψευδές καθώς δεν ικανοποιείται από καμία αποδιδόμενη έννοια, οπότε το antecedent μέρος πρέπει επίσης να μην ικανοποιείται από καμία αποδιδόμενη έννοια. Για την χρήση πολλαπλών atoms χρησιμοποιούνται σύνδεσμοι (conjunction). Αξιοσημείωτο είναι το γεγονός ότι συνδεόμενα consequents μπορούν εύκολα να μετασχηματιστούν (μέσω των μετασχηματισμών Lloyd-Topor) σε πολλαπλούς κανόνες, ο καθένας από τους οποίους φέρει ένα μόνο ξεχωριστό consequent. Η συλλογιστική βασισμένη σε SWRL συνηθίζεται να γίνεται στα επίπεδα των ιδιοτήτων και των στιγμιότυπων.

Τα atoms σε αυτούς τους κανόνες μπορούν να είναι της μορφής C(x), P(x,y), sameAs(x,y) ή differentFrom(x,y), όπου το C είναι μια περιγραφή OWL, το P είναι μια ιδιότητα OWL, και τα x, y είναι είτε μεταβλητές, είτε άτομα OWL ή OWL data values. Η σύνταξη αυτών των atoms στην SWRL απεικονίζεται πιο περιληπτικά στο παρακάτω σχήμα:

Atom ← C(x) | D(z1) | R(x, y) | U(x, z1) | built-in(p ; z1 ; ::: ; zn) | x = y | x != y

όπου:

C = class
D = data type
R = object Property
U = data type Property
x, y = variable name or invividual name
z1 ; ::: ; zn = data type variable name or data type value name
p = built-in name

Σχήμα 1. SWRL Atom Syntax

Οι κανόνες αποθηκεύονται ως μέρος της οντολογίας και παράλληλα είναι δυνατή η αλληλεπίδρασή τους με reasoners. Αν και συνεχώς αυξάνεται η παροχή εργαλείων συλλογιστικής που υποστηρίζουν την SWRL, ξεχωρίζουν τα Bossam, R2ML, Hoolet, Pellet, KAON2, RacerPro και SWRLTab. Είναι γνωστό πως η OWL φέρει συμπερασματικές δυνατότητες διαμέσου των OWL χαρακτηριστικών των ιδιοτήτων, όπως αναστροφή, συμμετρία και συγκρότηση έμμεσων σχέσεων μέσω συνδυασμού άμεσων σχέσεων (transitive property). Η SWRL φέρει συμπερασματικές δυνατότητες μέσω των κανόνων. Προκειμένου να αποφευχθεί η αναγκαιότητα επανάληψης ανάμεσα στους OWL συμπερασμούς και στους SWRL συμπερασμούς, θα ήταν επιθυμητό οι μηχανές κανόνων να μπορούσαν να εφαρμόσουν τους OWL χαρακτηρισμούς. Αυτό σημαίνει ότι τα OWL χαρακτηριστικά θα πρέπει να μεταφραστούν στα SWRL ισοδύναμά τους. Γι' αυτόν τον λόγο, στην SWRL είναι απόλυτα εφικτός ο καθορισμός κανόνων για χαρακτηριστικά που αντιπροσωπεύουν συμμετρία, αναστροφή και transitive δυνατότητα.

3.2 SWRL Built-ins

Το σύνολο των built-ins για την SWRL στηρίχτηκε από μια εμβατική προσέγγιση η οποία θα επιτρέψει περαιτέρω επεκτάσεις στις μελλοντικές εκδόσεις μέσα σε μια ιεραρχική ταξονομία. Συγχρόνως, θα παράσχει την ευελιξία για τις διάφορες εφαρμογές να επιλέξουν τα πρότυπα που υποστηρίζονται από κάθε έκδοση του SWRL. Επίσης, η προσέγγιση των SWRL built-ins βασίζεται στην επαναχρησιμοποίηση των υπαρχόντων built-ins για τις XQery και XPath, που από μόνες τους βασίζονται στο datatype XML Schema. Επιπλέον, ένα τέτοιο σύστημα θα μπορούσε να βοηθήσει στην λειτουργικότητα της SWRL με άλλα Web formalisms παρέχοντας μια επεκτάσιμη, μορφωματική υποδομή από built-ins για τις σημασιολογικές γλώσσες ιστού (Semantic Web Languages), τις υπηρεσίες ιστού (Web Services) και τις εφαρμογές ιστού (Web applications). Τα SWRL built-ins χρησιμοποιούνται στα builtin atoms και αναγνωρίζονται χρησιμοποιώντας το namespace http://www.w3.org/2003/11/swrlb.

Κατηγορίες Built-Ins	**Διαθέσιμα Built-Ins**
Built-ins for Comparisons	swrlb:equal, swrlb:notEqual, swrlb:lessThan, swrlb:lessThanOrEqual, swrlb:greaterThan, swrlb:greaterThanOrEqual
Math Built-Ins	swrlb:add, swrlb:subtract, swrlb:multiply, swrlb:divide, swrlb:integerDivide, swrlb:mod, swrlb:pow, swrlb:unaryPlus, swrlb:unaryMinus, swrlb:abs, swrlb:ceiling, swrlb:floor, swrlb:round, swrlb:roundHalfToEven, swrlb:sin, swrlb:cos, swrlb:tan
Built-ins for Boolean Values	swrlb:booleanNot
Built-Ins for Strings	swrlb:stringEqualIgnoreCase, swrlb:stringConcat, swrlb:substring, swrlb:stringLength, swrlb:normalizeSpace, swrlb:upperCase, swrlb:lowerCase, swrlb:translate, swrlb:contains, swrlb:containsIgnoreCase, swrlb:startsWith, swrlb:endsWith, swrlb:substringBefore, swrlb:substringAfter, swrlb:matches, swrlb:replace, swrlb:tokenize
Built-Ins for Date, Time and Duration	swrlb:yearMonthDuration, swrlb:dayTimeDuration, swrlb:dateTime, swrlb:date, swrlb:time, swrlb:addYearMonthDurations, swrlb:subtractYearMonthDurations, swrlb:multiplyYearMonthDurations, swrlb:divideYearMonthDurations, swrlb:addDayTimeDurations, swrlb:subtractDayTimeDurations, swrlb:multiplyDayTimeDurations, swrlb:divideDayTimeDurations, swrlb:subtractDates, swrlb:subtractTimes, swrlb:addYearMonthDurationToDateTime, swrlb:addDayTimeDurationToDateTime, swrlb:subtractYearMonthDurationFromDateTime, swrlb:subtractDayTimeDurationFromDateTime, swrlb:addYearMonthDurationToDate, swrlb:addDayTimeDurationToDate, swrlb:subtractYearMonthDurationFromDate, swrlb:subtractDayTimeDurationFromDate, swrlb:addDayTimeDurationToTime, swrlb:subtractDayTimeDurationFromTime, swrlb:subtractDateTimesYieldingYearMonthDuration, swrlb:subtractDateTimesYieldingDayTimeDuration
Built-Ins for URIs	swrlb:resolveURI, swrlb:anyURI
Built-Ins for Lists	swrlb:listConcat, swrlb:listIntersection, swrlb:listSubtraction, swrlb:member, swrlb:length, swrlb:first, swrlb:rest, swrlb:sublist, swrlb:empty

Πίνακας 2. SWRL Built-Ins

3.3 Η SWRL στο Protégé-OWL

3.3.1 Protégé-OWL Editor

Ο συντάκτης Protégé-OWL είναι μια επέκταση του Protégé που υποστηρίζει την Web Ontology Language (OWL). Η OWL είναι η πιο πρόσφατη εξέλιξη στις τυπικές γλώσσες οντολογίας, που καθιερώθηκε από τον World Wide Web (W3C) για να προωθήσει το Semantic Web vision. “Μια OWL οντολογία μπορεί να περιέχει περιγραφές κλάσεων, ιδιοτήτων και των στιγμιότυπών τους.”

Δεδομένης μιας τέτοιας οντολογίας, η διαδικαστική σημασιολογία της OWL καθορίζει πώς να αποκομίσουμε λογικά επακόλουθα, π.χ. γεγονότα τα οποία δεν βρίσκονται κυριολεκτικά στην οντολογία, αλλά συνεπάγονται από την σημασιολογία. Αυτές οι συνεπαγωγές μπορούν να βασίζονται σε ένα μόνο έγγραφο ή πολλαπλά έγγραφα που έχουν συνδυαστεί χρησιμοποιώντας προκαθορισμένους μηχανισμούς της OWL. Ο συντάκτης Protégé-OWL παρέχει τις ακόλουθες δυνατότητες στους χρήστες:

- Φόρτωση και αποθήκευση OWL και RDF οντολογιών
- Διαχείριση και απεικόνιση κλάσεων, ιδιοτήτων και κανόνων SWRL
- Καθορισμοί λογικών χαρακτηριστικών στις κλάσεις μέσω των OWL expressions
- Διαχείριση των OWL ατόμων για το Semantic Web markup

Η ευπροσάρμοστη αρχιτεκτονική του Protégé-OWL κάνει εύκολη την διαμόρφωση και την επέκτασή του. Το Protégé-OWL είναι στενά συνδεδεμένο με την Jena και έχει ένα ανοικτό λογισμικό τύπου Java API για την ανάπτυξη στοιχείων περιβάλλοντος ή αυθαίρετων υπηρεσιών σημασιολογικού ιστού, ανάλογα με τις ανάγκες και τις επιλογές του χρήστη. Για την εκπόνηση των εργασιών μας, χρησιμοποιήθηκε η έκδοση Protégé-OWL 3.4.4 με την προκαθορισμένη Java Virtual Machine που εμπεριέχεται στο πακέτο εγκατάστασης.

3.3.2 SWRLTab

Το SWRLTab είναι ένα περιβάλλον ανάπτυξης για την δημιουργία και την διαχείριση SWRL κανόνων μέσα στο Protégé-OWL. Όπως υποδηλώνει και το όνομά του, διατίθεται με τη μορφή καρτέλας και ενεργοποιείται μέσω των μενού Project → Configure… → Tab Widgets. Παρέχει ένα πλήθος από βιβλιοθήκες που μπορούν να χρησιμοποιηθούν στους κανόνες, συμπεριλαμβανομένων βιβλιοθηκών που αλληλεπιδρούν με XML έγγραφα, spreadsheets, και βιβλιοθήκες για μαθηματικές πράξεις, αλφαριθμητικά, RDFS, και προσωρινούς χειριστές. Επίσης παρέχεται και μια γλώσσα ερωτημάτων πάνω στην OWL βασισμένη στο πρότυπο SWRL που αποκαλείται SQWRL. Αν και το SWRLTab απαρτίζεται από πολλά και διαφορετικά μέρη λογισμικού, τα ακόλουθα αφορούν το αντικείμενο ενασχόλησής μας:

- `SWRL Editor`, υποστηρίζει την συγγραφή και την αποθήκευση των SWRL κανόνων σε μια OWL οντολογία
- `SWRL Built-in Libraries`, ένα πλήθος από ενσωματωμένες βιβλιοθήκες παρέχονται από το SWRLTab. Αυτές περιλαμβάνουν τις βασικές built-in βιβλιοθήκες της SWRL καθώς και κάποιες επιπλέον για την διενέργεια ερωτημάτων στις OWL οντολογίες
- `SQWRL Query Tab`, το query tab παρέχει ένα γραφικό περιβάλλον για την απεικόνιση των αποτελεσμάτων των SQWRL ερωτημάτων
- `SWRL Jess Bridge`, μια γέφυρα είναι απαραίτητη για την συγκρότηση μιας μηχανής κανόνων μέσα στο Protégé-OWL ως ενιαίο σύνολο ώστε να γίνει δυνατή η εκτέλεση των SWRL κανόνων. Μια γέφυρα για την Jess rule engine διατίθεται με την διανομή του Protégé-OWL. Περιλαμβάνεται και το γραφικό περιβάλλον SWRLJessTab που είναι ικανό για διάδραση με αυτήν την γέφυρα

Εκτός των παραπάνω στοιχείων, υπάρχουν και αρκετά APIs όπως τα SWRL Query API (παρέχει μια συλλογή από JDBC-like Java interface για την ανάκτηση των αποτελεσμάτων των SQWRL ερωτημάτων) και SWRL Factory (παρέχει ένα Java API για την δημιουργία και την διαχείριση SWRL κανόνων σε μια οντολογία), που χρησιμοποιούνται σε προγραμματιστικό περιβάλλον από τους χρήστες για την ανάπτυξη κανόνων σε δικές τους εφαρμογές.

Το SWRLTab θα πρέπει να είναι ορατό για όλες τις οντολογίες OWL που έχουν μια εγγραφή namespace, η οποία αντιστοιχεί στο σύμφωνο με την SWRL namespace, http://www.w3.org/2003/11/swrl που φέρει το ψευδώνυμο swrl. Εάν η οντολογία δεν έχει την παραπάνω εγγραφή, το SWRLTab απενεργοποιείται εξ' ορισμού. Για την ενεργοποίηση του SWRLTab σε ένα project που δεν περιέχει την εγγραφή του SWRL namespace, πηγαίνουμε στο Project → Configure… → Tab Widgets και τσεκάρουμε το κουτάκι "SWRLTab". Άμεσα εμφανίζεται η καρτέλα και προστίθεται μια namespace εγγραφή για την SWRL στην τρέχουσα οντολογία. Εξ' ορισμού, το SWRLTab αυτόματα θα προσθέσει και τις οντολογίες SWRLA και SQWRL στην λίστα των ενεργών οντολογιών, όταν αυτό ενεργοποιηθεί. Στην περίπτωση που δεν είναι επιθυμητή αυτή η

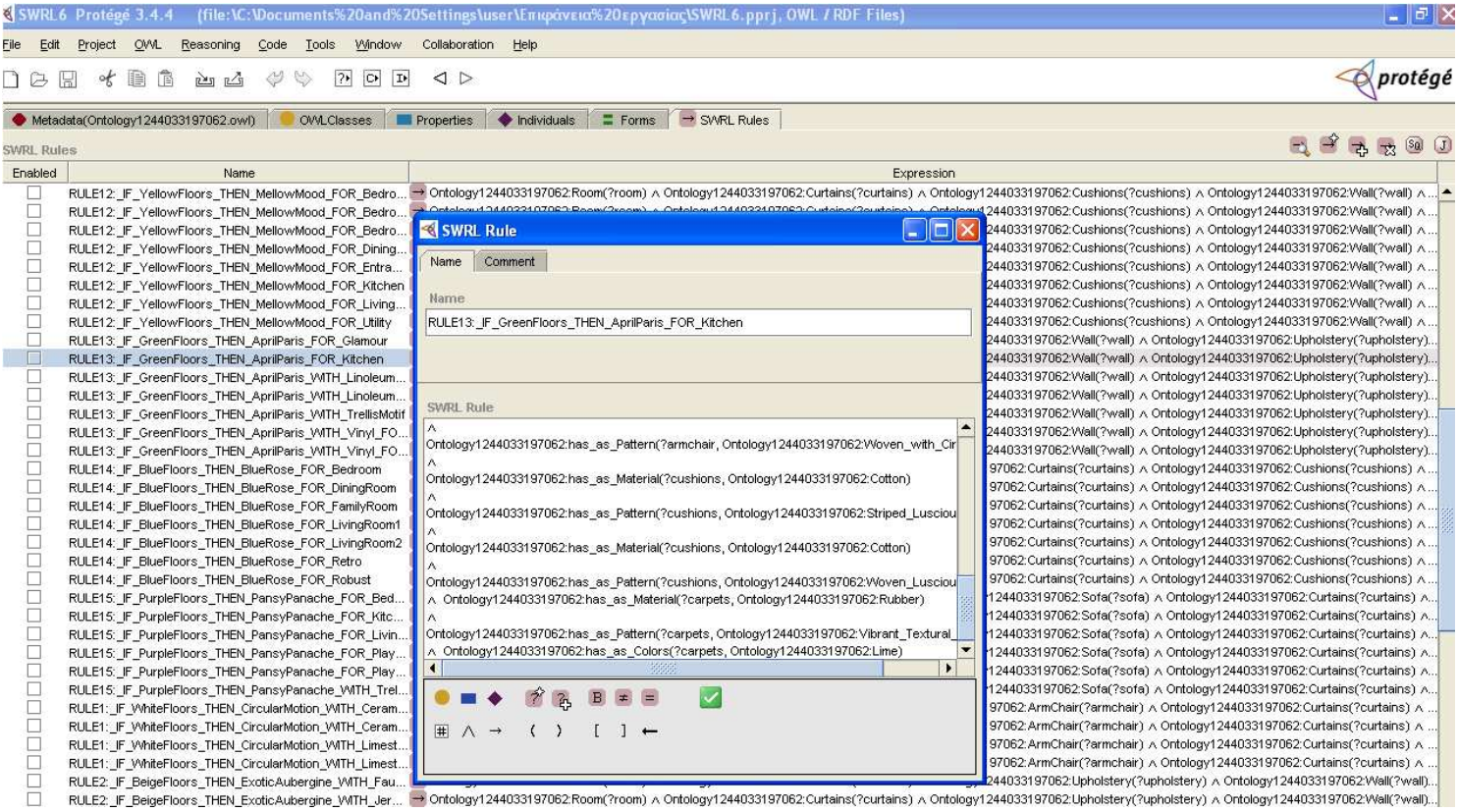
αυτόματη εισαγωγή και κρίνεται απαραίτητη η απενεργοποίησή της, τότε προσθέτουμε και θέτουμε την ιδιότητα protege.owl.swrl.exclude_standard_imports ίση με μηδέν, στο αρχείο protege.properties που βρίσκεται στον κύριο φάκελο εγκατάστασης του Protégé. Επίσης, μπορεί να απενεργοποιηθεί προσθέτοντάς αυτήν την ιδιότητα και μέσω του Protégé GUI από το Project→ Preferences → Property Files → protégé.properties.

Καθώς το SWRLTab αποτελεί μέρος του Protégé-OWL 3.4 δεν χρειάζεται να "κατέβει" και να εγκατασταθεί ξεχωριστά. Παρόλα αυτά, επειδή πολλά από τα κύρια στοιχεία που αποτελούν το SWRLTab είναι συνεχώς υπό ανάπτυξη, κρίνεται αναγκαίο να χρησιμοποιείται πάντα η πιο πρόσφατη υποέκδοση (η εργασία εκπονήθηκε πάνω στο Protégé-OWL 3.4.4 Build 579). Επιπλέον, η διανομή του Protégé-OWL δεν περιέχει καμία μηχανή κανόνων. Αν κάποια συγκεκριμένη μηχανή κανόνων χρησιμοποιηθεί, πρέπει να εγκατασταθεί ξεχωριστά. Προς το παρών, μόνο η Jess rule engine υποστηρίζεται.

3.4 SWRL Editor

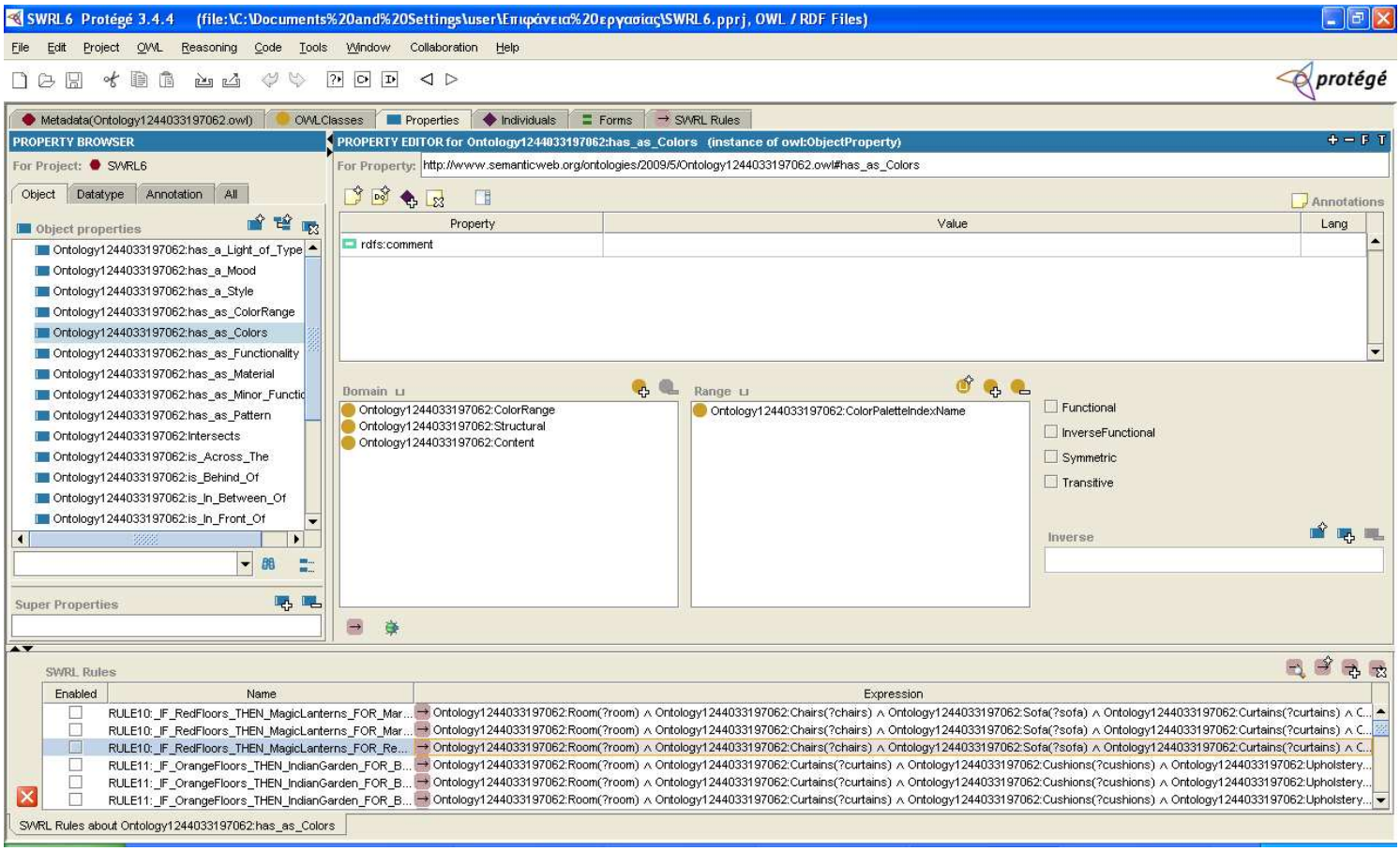
3.4.1 Εισαγωγή

Ο SWRL Editor είναι μια επέκταση του SWRLTab του Protégé-OWL που επιμελείται την σύνταξη των SWRL κανόνων. Μπορεί να χρησιμοποιηθεί για την δημιουργία νέων SWRL κανόνων, την τροποποίηση ήδη υπαρχόντων SWRL κανόνων, και την γενική ανάγνωση και συγγραφή αυτών των κανόνων. Ο SWRL Editor αποτελεί μέρος του Protégé-OWL και βρίσκεται μέσα στο SWRLTab. Αυτό το tab θα πρέπει να είναι ορατό για όλες τις OWL οντολογίες που φέρουν την SWRL ontology. Απενεργοποιείται εξ' ορισμού στην περίπτωση που η "φορτωμένη" οντολογία δεν αναφέρεται σε αυτήν την οντολογία. Η παρακάτω εικόνα παρουσιάζει το SWRLTab μαζί με τον multi-line editor:



Εικόνα 2. Ο SWRL Editor μέσα στο SWRLTab

Υποθέτοντας πως το SWRLTab είναι ενεργοποιημένο υπάρχουν δύο τρόποι αλληλεπίδρασης με τον SWRL Editor. Ο πρώτος μηχανισμός παρέχεται μέσω του SWRLTab. Διαθέτει μια κύρια καρτέλα που απεικονίζει όλους τους SWRL κανόνες σε γραμμές και στήλες. Επιλέγοντας κάποιον κανόνα εμφανίζεται ο multiline editor για την διαχείριση του κανόνα. Ο δεύτερος μηχανισμός επιτρέπει στους χρήστες να βρουν τους κανόνες που σχετίζονται με την τρέχων επιλεγμένη OWL κλάση, ιδιότητα, ή άτομο στις αντίστοιχες καρτέλες του Protégé-OWL. Παραδείγματος χάριν, αν κάποιος χρήστης εξετάζει μια ιδιότητα μέσω της καρτέλας "Properties" του Protégé-OWL, μπορεί να επιλέξει το σύμβολο του βέλους στην κάτω αριστερή γωνία της καρτέλας. Αυτό θα εμφανίσει μια λίστα των SWRL κανόνων που αναφέρονται σε αυτήν την ιδιότητα. Η ίδια διαδικασία ισχύει και για τις καρτέλες των κλάσεων και ατόμων. Η εικόνα 2 παρουσιάζει τον πρώτο μηχανισμό ενώ η εικόνα 3 τον δεύτερο μηχανισμό.



Εικόνα 3. Ο SWRL Editor στην καρτέλα των ιδιοτήτων του Protégé-OWL

3.4.2 Επεξεργασία κανόνων

Υπάρχουν δύο τρόποι επεξεργασίας των κανόνων. Οι κανόνες μπορούν να επιμεληθούν είτε στον πίνακα του SWRL Rules που τους περιέχει ή στον multi-line editor που παρέχεται. Η διαφορά ανάμεσα στους δύο τρόπους είναι καθαρά και μόνο οπτική. Όποιος τρόπος και να ακολουθηθεί, οι ίδιοι μηχανισμοί εφαρμόζονται, αν και στην περίπτωση του multi-line editor υπάρχουν περισσότερες επιλογές:

- Για την επεξεργασία ενός κανόνα στον πίνακα, απλώς κάνουμε διπλό κλικ στον κανόνα που επιθυμούμε στην στήλη Expression. Τότε, ένας κέρσορας που αναβοσβήνει και ένα panel εικονιδίων θα εμφανιστούν στο κάτω μέρος του επιλεγμένου κανόνα. Το μόνο που απομένει

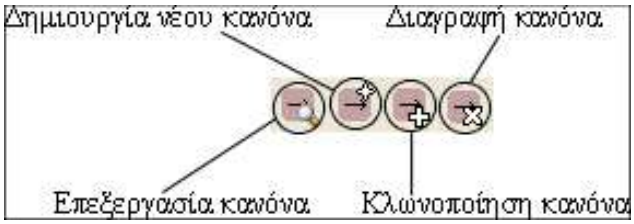
να κάνουμε είναι να τον επεξεργαστούμε και να τον σώσουμε πατώντας το κουμπί Enter. Για να εγκαταλείψουμε την επεξεργασία του κανόνα πατάμε το κουμπί Escape

- Για την επεξεργασία ενός κανόνα στον multi-line editor, πατάμε δεξί κλικ στον κανόνα που επιθυμούμε και επιλέγουμε “Edit rule in multi-line editor…”. Επίσης αυτός ο τρόπος επεξεργασίας κανόνων παρέχεται και μέσω του πρώτου εικονιδίου από την σειρά εικονιδίων που υπάρχουν στην πάνω δεξιά γωνία του πίνακα SWRL Rules. Αυτό προκαλεί την εμφάνιση ενός παραθύρου που περιέχει μια περιοχή κειμένου για την επεξεργασία του κανόνα και το ίδιο panel εικονιδίων με την προηγούμενη περίπτωση. Για την αποθήκευση του κανόνα πατάμε το κουμπί "Ok", ενώ για να εγκαταλείψουμε την διαδικασία επεξεργασίας πατάμε το κουμπί "Cancel". Ο multi-line editor επιτρέπει επιπλέον την αλλαγή του ονόματος και του annotation του κανόνα

Σε οποιονδήποτε από τους δύο τρόπους και αν επιλέξουμε, ο editor δεν πρόκειται να επιτρέψει την αποθήκευση ενός ημιτελούς ή λανθασμένου SWRL κανόνα. Κατά την επεξεργασία ενός κανόνα, το κουμπί "√" δείχνει την συντακτική του ορθότητα. Αν κάποια στιγμή αλλάξει σε "E" επισημαίνει την ύπαρξη λάθους, το οποίο αναφέρεται στο κάτω μέρος του multi-line editor. Στην περίπτωση που χρησιμοποιούμε την πρώτη μέθοδο επεξεργασίας κανόνων θα πρέπει να κλικάρουμε στο "E" για την επισήμανση αυτού του λάθους.

Οι SWRL κανόνες μπορούν να γραφούν εξ’ ολοκλήρου από το πληκτρολόγιο, αλλά στην περισσότερες περιπτώσεις είναι άβολο, χρονοβόρο και απαιτείται επιλογή συγκεκριμένων OWL οντοτήτων από την φορτωμένη οντολογία. Τα εικονίδια του panel παρέχουν μια σειρά από κουτιά διαλόγου για την επιλογή των OWL κλάσεων, ιδιοτήτων, ατόμων και άλλων τύπων οντοτήτων. Σε κάθε εικονίδιο εμφανίζεται περιγραφή της λειτουργίας που επιτελεί. Επίσης, ο SWRL Editor παρέχει την δυνατότητα αυτόματης συμπλήρωσης της οντότητας που πληκτρολογούμε με την χρήση του κουμπιού Tab εφόσον έχει μοναδική επέκταση. Ειδάλλως, εμφανίζει μα λίστα με τις πιθανές επιλογές. Παραδείγματος χάριν, πληκτρολογώντας στην περιοχή του κειμένου του SWRL Editor το “xsd:” και πατώντας αμέσως το κουμπί Tab, εμφανίζεται ένα κουτί διαλόγου με την λίστα όλων των πιθανών datatypes. Επίσης, το datatypes εικονίδιο μπορεί να επιλεγεί από το panel εικονιδίων για την εμφάνιση της ίδιας λίστας. Παρομοίως, πληκτρολογώντας “swrlb:” και πατώντας το κουμπί Tab, εμφανίζεται ένα κουτί διαλόγου με την λίστα όλων των δυνατών built-ins που χρησιμοποιούν αυτό το namespace. Το built-in εικονίδιο μπορεί να εμφανίσει την ίδια λίστα. Ο editor παρέχει την ίδια δυνατότητα αυτόματης συμπλήρωσης και για annotation τιμές που αναφέρονται σε OWL κλάσεις, ιδιότητες και άτομα.

Για την δημιουργία ενός SWRL κανόνα επιλέγουμε το δεύτερο εικονίδιο από την σειρά εικονιδίων που υπάρχουν στην πάνω δεξιά γωνία του πίνακα SWRL Rules. Αμέσως εμφανίζεται ο multi-line editor για την συγγραφή του κανόνα και την αποθήκευσή του. Επιπλέον, υπάρχει και η δυνατότητα κλωνοποίησης ενός SWRL κανόνα με την χρήση του τρίτου εικονιδίου από την ίδια σειρά εικονιδίων. Ενώ το τέταρτο εικονίδιο διαγράφει τον επιλεγμένο SWRL κανόνα από την λίστα κανόνων του πίνακα SWRL Rules. Ένας διαγεγραμμένος κανόνας δεν μπορεί να ανακτηθεί με κανέναν τρόπο, καθώς δεν υποστηρίζεται επιλογή αναίρεσης. Οι SWRL κανόνες αποθηκεύονται σαν OWL άτομα μαζί με την σχετιζόμενη OWL οντολογία, έτσι ώστε να είναι δυνατή η αποθήκευσή τους χρησιμοποιώντας τον κλασσικό μηχανισμό αποθήκευσης του Protégé-OWL. Όταν μια οντολογία που περιέχει SWRL κανόνες φορτωθεί στο Protégé-OWL, ο SWRL Editor ενεργοποιείται αυτόματα, και μπορεί να χρησιμοποιηθεί για την εξέταση αυτών των αποθηκευμένων κανόνων.



Εικόνα 4. Λειτουργίες Εικονιδίων του SWRL Editor

3.4.3 Επικύρωση SWRL κανόνων

Ο SWRL Editor εκτελεί μόνο δύο βασικές λειτουργίες κατά τη συγγραφή των κανόνων, ελέγχει το συντακτικό και τη σημασιολογία των κανόνων. Με απλά λόγια, εγγυάται ότι ένας κανόνας είναι συντακτικά σωστός πριν αποθηκευτεί και επίσης εγγυάται ότι οποιαδήποτε αναφορά σε OWL οντότητες είναι έγκυρη. Παρόλα αυτά, ο editor δε μπορεί να πραγματοποιήσει κανένα απολύτως έλεγχο ακεραιότητας ανάμεσα στους κανόνες σε μια οντολογία, δε μπορεί να εγγυηθεί την αποτροπή της διένεξης των κανόνων με τα OWL axioms, και δε μπορεί να αποτρέψει τον χρήστη από το να γράψει ορθά συντακτικούς και σημασιολογικούς κανόνες, αλλά χωρίς κανένα απολύτως νόημα.

Ο SWRL Editor από μόνος του δεν μπορεί να πραγματοποιήσει καμία συνεπαγωγή κανόνων. Για αυτόν τον λόγο, απαιτείται ένας μηχανισμός γέφυρας για την συνεργασία με rule engines. Προς το παρών, μόνο η Jess rule engine υποστηρίζεται και είναι προσβάσιμη από το SWRLJessTab. Αξιοσημείωτο είναι το γεγονός ότι ο SWRL Editor χρησιμοποιεί ένα ξεχωριστό back-end factory για την διαχείριση των OWL ατόμων που αναπαριστούν τους SWRL κανόνες. Αυτό το factory μπορεί να χρησιμοποιηθεί για την δημιουργία και την εξέταση των στοιχείων των SWRL κανόνων προγραμματιστικά. Αυτό επιτυγχάνεται μέσω ενός Java API που είναι ικανό για την δημιουργία, τον έλεγχο και την μεταχείριση υπαρχόντων SWRL κανόνων σε μια βάση γνώσης. Εφαρμογές που έχουν αναπτυχθεί σε Java μπορούν να χρησιμοποιήσουν αυτό το factory για την διάδραση με τους SWRL κανόνες σε επίπεδο Protégé-OWL API και να παραλείψουν εντελώς τον editor.

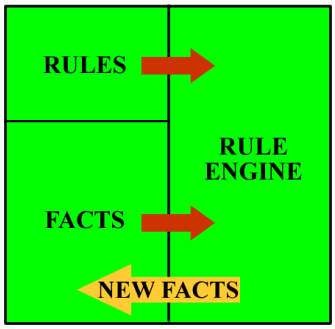
Οι SWRL κανόνες αποθηκεύονται σε OWL άτομα ώστε να διαχειρίζονται κατευθείαν μέσω των μηχανισμών που παρέχονται στο Protégé-OWL. Ωστόσο, η απευθείας διαχείριση αυτών των ατόμων δεν συνίσταται καθώς μπορεί να καταστήσει μη λειτουργικούς τους κανόνες και να προκαλέσει προβλήματα διενέξεων στην οντολογία. Μόνο έμπειροι χρήστες δύναται να ασχοληθούν με αυτό το αντικείμενο, αφού οι λειτουργίες που προσφέρει είναι όμοιες με αυτές του SWRL Editor, και απλά αποτελεί μια εναλλακτική μορφή μεταχείρισης των SWRL κανόνων.

Οι SWRL κανόνες αναπαριστώνται ως OWL άτομα που περιέχονται στις OWL οντολογίες, που συνήθως αποθηκεύονται ως αρχεία τύπου OWL/RDF. Αυτό έχει ως αποτέλεσμα προγράμματα λογισμικού τρίτων, να μπορούν να αλληλεπιδράσουν με τους κανόνες που περιγράφονται σε αυτά τα αρχεία. Πάντως, ένας μικρός περιορισμός εντοπίζεται στο γεγονός ότι το Protégé-OWL χρησιμοποιεί την Jena parser library για την παραγωγή αυτών των αρχείων, με συνέπεια την δύσκολη – αλλά όχι αδύνατη – ανάλυση με κανονικούς XML parsers. Παρόλα αυτά, η Jena μπορεί να χρησιμοποιηθεί ωε parser αυτών των αρχείων και σε Java εφαρμογές.

Όπως αναφέρθηκε σε προηγούμενη παράγραφο, ο SWRL Editor επιτρέπει αποκλειστικά και μόνο την αποθήκευση κανόνων που αναφέρονται σε τρέχοντες έγκυρες OWL κλάσεις, ιδιότητες και άτομα. Όμως, αυτές οι οντότητες μπορούν κάποια στιγμή να χάσουν την εγκυρότητά τους λόγω διαγραφής τους ή αλλαγής του ονόματός τους. Παραδείγματος χάριν, αν ένας SWRL κανόνας αναφέρεται σε μια κλάση A σε μια εισαχθείσα οντολογία με πρόθημα ontol, και αργότερα το όνομα αυτής της κλάσης αλλάξει, τότε η αναφορά δεν είναι πια έγκυρη. Όταν εξετάσουμε τον κανόνα με την λανθασμένη αναφορά, ο editor θα έχει αντικαταστήσει την προηγουμένως έγκυρη αναφορά ονόματος με την ένδειξη σφάλματος <INVALID_CLASS[ontol:A]>. Παρόμοιες ενδείξεις σφαλμάτων υπάρχουν και στις περιπτώσεις λανθασμένης αναφοράς σε άτομα και ιδιότητες. Επίσης, για την διαγραφή κλάσεων, ιδιοτήτων ή ατόμων από την ίδια οντολογία, οι ενδείξεις σφάλματος έχουν την μορφή <DELETED_CLASS>, <DELETED_PROPERTY> και <DELETED_INDIVIDUAL>. Είναι εμφανές όμως, ότι σε αυτή την περίπτωση δεν παρέχεται η δυνατότητα αναγραφής της έγκυρης αναφοράς που υπήρχε πριν την διαγραφή της οντότητας. Πάντως, ο editor επιτρέπει την παραμονή μη έγκυρων αναφορών σε μια οντολογία επ’ αόριστον, εφόσον οι κανόνες-παραβάτες δεν έχουν μεταχειριστεί μετά την λανθασμένη τους αναφορά. Εάν προβούν σε εξέταση από τον χρήστη, δεν θα μπορούν να αποθηκευτούν εωσότου διορθωθούν όλες αυτές οι αναφορές.

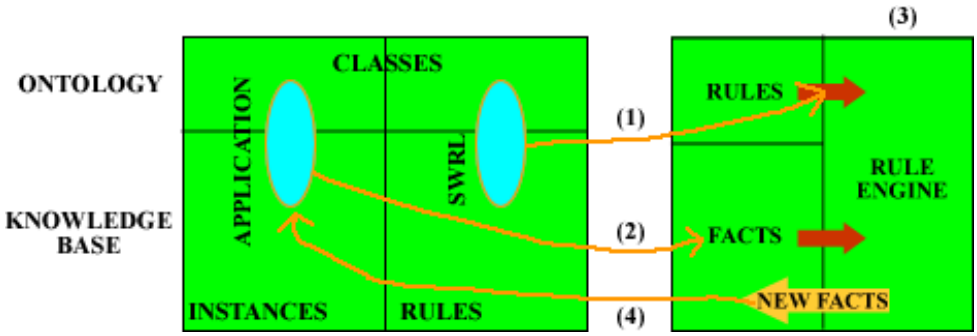
3.4.4 Εκτέλεση SWRL κανόνων

Ο συνδυασμός του συντάκτη οντολογιών Protégé-OWL και της ευχρηστίας που παρέχει το SWRLTab στην συγγραφή SWRL κανόνων, δεν είναι αρκετός για την υποστήριξη εκτέλεσης αυτών των κανόνων, καθώς κρίνεται αναγκαία η ύπαρξη μιας μηχανής κανόνων (rule engine). Η γενική εικόνα μιας rule engine είναι η διενέργεια συλλογισμού (reasoning) χρησιμοποιώντας έναν αριθμό από κανόνες και γεγονότα ως είσοδό της. Οποιαδήποτε νέα γεγονότα τεκμηριωθούν μπορούν να χρησιμοποιηθούν ως είσοδο για την εκτέλεση νέων κανόνων.



Σχήμα 2. Rule Engine Execution

Ωστόσο οι κανόνες και τα γεγονότα πρέπει να είναι διαθέσιμα σε ένα format που θα γνώριμο και προσβάσιμο στην rule engine. Για αυτόν τον λόγο γίνονται συνεχώς μεταφράσεις της τρέχων state-of-the-art κατάστασης για να γίνει δυνατή η εκτέλεση των SWRL κανόνων βάση των δεδομένων που παρέχει το Protégé. Πρώτο βήμα είναι η μετάφραση των SWRL κανόνων και η εισαγωγή τους στην rule engine. Έπειτα, η οντολογία και η βάση γνώσης μεταφράζονται και εισάγονται στην rule engine. Ακολουθεί η διαδικασία του reasoning και τα αποτελέσματα αυτής της διαδικασίας μεταφράζονται πίσω στο Protégé format. Το παρακάτω σχήμα απεικονίζει αυτά τα βήματα:



Σχήμα 3. Βήματα εκτέλεσης ενός SWRL κανόνα

3.5 SWRLTab Built-in Libraries

Η διανομή του Protégé-OWL περιλαμβάνει αρκετές ενσωματωμένες βιβλιοθήκες που παρέχονται από το SWRLTab. Σε αυτήν την παράγραφο, ακολουθεί αναλυτική περιγραφή της δήλωσης και της λειτουργίας όλων αυτών των βιβλιοθηκών:

3.5.1 Core SWRL Built-Ins Library

Πρόκειται για μια βιβλιοθήκη που παρέχει το σύνολο των επίσημων built-ins που σχεδιάστηκαν για την αξιοποίησή τους στους SWRL κανόνες. Ανάμεσά τους βρίσκονται βασικοί μαθηματικοί χειριστές και διαδικασίες για την διαχείριση αλφαριθμητικών και ημερομηνιών. Τα built-ins που

περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την Core SWRL Built-in Ontology και είναι μια οντολογία που διανέμεται μαζί με το Protégé-OWL. Όλα τα built-ins αναφέρονται μαζί με τις κατηγορίες τους στον Πίνακα 1. του παρόντος κεφαλαίου. Το κοινότυπο πρόθημα (prefix) είναι το swrlb.

Όλα τα μαθηματικο-ειδή built-ins μπορούν να εφαρμοστούν για τους συνήθεις αριθμητικούς τύπους. Επιπλέον, όλα τα Boolean και αλφαριθμητικά built-ins μπορούν επίσης να χρησιμοποιηθούν. Μόνη ένσταση αποτελούν τα string built-ins substringBefore και substringAfter, αφού σε σχέση με την υποβολή του SWRL δεν επιτρέπουν την χρήση του προαιρετικού τρίτου argument που περιγράφεται στο έγγραφο W3C XQuery 1.0 and XPath 2.0 Functions and Operators. Ομοίως, η εφαρμογή του tokenize built-in δεν υποστηρίζει το προαιρετικό τρίτο argument που περιγράφεται στο ίδιο έγγραφο, και ταυτόχρονα έχει και περιορισμένες επιλογές pattern για το δεύτερο argument (που ανταποκρίνονται στο pattern της Java StringTokenizer class).

Όλα τα temporal built-ins μπορούν να εφαρμοστούν, ενώ το SWRLTab παρέχει μια επιπλέον temporal built-in library που στις περισσότερες περιπτώσεις αποδεικνύεται πιο εύχρηστη. Παρόλα αυτά, τα list και URI built-ins δεν έχουν υλοποιηθεί ακόμα με αποτέλεσμα να μην υποστηρίζεται η εφαρμογή τους.

3.5.2 SQWRL Built-In Library

Περιέχει έναν αριθμό από built-ins που επεκτείνουν την SWRL στην SQWRL. Η SQWRL (Semantic Query-Enhanced Web Rule Language, προφέρεται squirrel) είναι μια γλώσσα βασισμένη στο πρότυπο SWRL για την διενέργεια ερωτημάτων σε OWL οντολογίες. Διαθέτει παρόμοιες μεθόδους με την SQL για την ανάκτηση γνώσης από την OWL. Ο καθορισμός της SQWRL γίνεται χρησιμοποιώντας μια βιβλιοθήκη από SWRL built-ins που δομούν αποδοτικά μια γλώσσα ερωτημάτων πάνω από την SWRL. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την SQWRL Ontology. Το κοινότυπο πρόθημα είναι το sqwrl. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής “Import Ontology” στην καρτέλα Metadata του Protégé-OWL. Η Jess rule engine είναι απαραίτητη για την εκτέλεση των SQWRL ερωτημάτων. Πληροφορίες σχετικά με την εγκατάστασή της υπάρχουν στην ενότητα 3.7.2 του παρόντος κεφαλαίου. Όσον αφορά την πρόσβαση στα αποτελέσματα των ερωτημάτων χρησιμοποιήσαμε το SQWRLQueryTab που αποτελεί ένα γραφικό περιβάλλον χρήστη, με την ικανότητα της απευθείας διάδρασης των ερωτημάτων και της εμφάνισης των αποτελεσμάτων τους μέσα στο SWRLTab. Περισσότερες πληροφορίες σχετικά με το SQWRLQueryTab υπάρχουν στην ενότητα 3.6 του παρόντος κεφαλαίου.

Τα SQWRL ερωτήματα όντας χτισμένα πάνω στην SWRL μπορούν να χρησιμοποιηθούν και για την ανάκτηση γνώσης που έχει συμπεραθεί από SWRL κανόνες. Επιπλέον, τα SQWRL ερωτήματα μπορούν ελεύθερα να συνεργαστούν με άλλες built-in libraries. Αυτή η δυνατότητα ελεύθερου καθορισμού και αξιοποίησης των built-ins στα ερωτήματα, παρέχει ένα μέσο συνεχούς επέκτασης της εκφραστικής δύναμης της γλώσσας ερωτημάτων.

Τα SQWRL ερωτήματα λειτουργούν πάνω σε γνωστά άτομα της τρέχοντος φορτωμένης OWL οντολογίας. Είναι σημαντικό να κατανοήσουμε ότι η SQWRL δεν παρέχει κανένα τρόπο πρόσβασης στην πληροφορία που προκύπτει μέσα από ένα κανόνα. Οπότε, τα αποτελέσματα των ερωτημάτων δεν μπορούν να ενσωματωθούν πίσω στην οντολογία, αφού ένας τέτοιος μηχανισμός θα παραβίαζε την open world assumption φύση του OWL και θα οδηγούσε σε non-monotonicity. Ο όρος monotonicity είναι μια ιδιότητα πολλών συστημάτων λογικής που δηλώνει πως η υπόθεση κάθε προερχόμενου γεγονότος μπορεί ελεύθερα να επεκταθεί με επιπλέον εικασίες. Αυτό συνεπάγεται στο γεγονός ότι οποιαδήποτε αληθής δήλωση σε μια πρόταση λογικής (logics) που φέρει αυτήν την ιδιότητα, εξακολουθεί να είναι αληθής πάντα, ακόμα και μετά την προσθήκη νέων axioms. Logics που φέρουν αυτήν την ιδιότητα καλούνται monotonic, σε αντίθεση με αυτά που δεν την φέρουν και αναφέρονται ως non-monotonic.

Παρόλο που οι δύο βασικές κατηγορίες χειριστών (operators) του SQWRL εκπληρώνονται ως SWRL built-ins, δεν λειτουργούν όπως όλα τα καθιερωμένα built-ins. Σε αντίθεση με αυτά, δεν αξιολογούν τα ορίσματά τους (arguments) και επιστρέφουν αληθής τιμή αν τα arguments ικανοποιούν κάποιο κατηγόρημα. Επιπλέον, επιστρέφουν πάντα αληθής τιμή και δρουν ως συσσωρευτές χτίζοντας δομές δεδομένων με μορφή πίνακα εκτός των ορίων της οντολογίας. Αυτό τους κάνει ανεξάρτητους

από τους όρους της οντολογίας, εφόσον δεν πραγματοποιούν καμία τροποποίηση στην οντολογία. Ύψιστης σημασίας ζήτημα, δεν παραβιάζουν τα SWRL semantics. Τέλος, σε αντιπαράθεση με την OWL και την SWRL, η SQWRL υιοθετεί την έννοια του unique name assumption.

Οι δύο μεγάλες κατηγορίες χειριστών που παρέχονται στην SQWRL για την διενέργεια εκτέλεσης των ερωτημάτων, μαζί με τα built-ins και τις λειτουργίες που αυτά παρέχουν, είναι οι εξής:

- `Core Operators`, η core SQWRL γλώσσα που παίρνει το καθιερωμένο μέρος ενός antecedent SWRL κανόνα και το χρησιμοποιεί αποδοτικά ως περιγραφικό σχέδιο για ένα ερώτημα. Μετέπειτα, αντικαθιστά το consequent μέρος του κανόνα με ένα πρότυπο ανάκτησης. Η SQWRL χρησιμοποιεί, ως ένα βαθμό, τις δυνατότητες των SWRL built-ins για την επέκτασή της. Ο βασικός χειριστής είναι ο sqwrl:select. Παίρνει ένα ή περισσότερα arguments, τα οποία είναι συνήθως μεταβλητές που χρησιμοποιήθηκαν στο περιγραφικό σχέδιο του ερωτήματος, και χτίζει ένα πίνακα χρησιμοποιώντας τα arguments ως στήλες του πίνακα. Τα παραδείγματα που ακολουθούν βρίσκονται στο http://protege.cim3.net/cgi-bin/wiki.pl?CoreSQWRL, και επιδεικνύουν τους χειριστές αυτής της κατηγορίας και την αντίστοιχη χρήση τους. Βασίζονται στην ύπαρξη μιας υποθετικής οντολογίας με μια κλάση Person, η οποία έχει ως υποκλάσεις Male και Female με σχετιζόμενες ιδιότητες hasAge και hasName. Επιπλέον μια κλάση Car, που μπορεί να συσχετιστεί με άτομα της κλάσης Person μέσω της ιδιότητας hasCar:
  - **Βασικά ερωτήματα**, χρησιμοποιούν ως επί το πλείστον τον χειριστή sqwrl:select μαζί με άλλα built-ins ή χειριστές. Παραδείγματος χάριν, για την εμφάνιση μιας λίστας με όλα τα αμάξια που ανήκουν σε κάθε άνθρωπο, μπορούμε να γράψουμε:
    ```
    Person(?p) ^ hasCar(?p, ?c) -> sqwrl:select(?p, ?c)
    ```
    Για την εξαγωγή όλων των ανθρώπων σε μια οντολογία που η ηλικία τους είναι μικρότερη του 25, μαζί με την ηλικία τους, μπορούμε να γράψουμε:
    ```
    Person(?p) ^ hasAge(?p, ?a) ^ swrlb:lessThan(?a, 25) ->
                    sqwrl:select(?p, ?a)
    ```
    Αν υπάρξουν πανομοιότυπα ζευγάρια που ταιριάζουν σε ένα ερώτημα, τότε θα εμφανιστούν πολλαπλές φορές. Ο χειριστής sqwrl:selectDistinct μπορεί να χρησιμοποιηθεί για την αφαίρεση αυτών των ομοιοτήτων
  - **Ερωτήματα καταμέτρησης (Counting)**, δυνατότητα απλής καταμέτρησης παρέχεται μέσω του χειριστή sqwrl:count, ο οποίος δέχεται ένα μόνο argument. Παραδείγματος χάριν, για την καταμέτρηση αποκλειστικά του αριθμού των αμαξιών που ανήκουν σε ανθρώπους σε μια οντολογία, μπορούμε να γράψουμε:
    ```
    Person(?p) ^ hasCar(?p, ?c) -> sqwrl:count(?c)
    ```
    Στην περίπτωση που υπάρξουν πανομοιότυπα στοιχεία, κάθε στοιχείο θα συμπεριληφθεί στην μέτρηση. Η καταμέτρηση λειτουργεί στα αποτελέσματα που παράγει – όχι στην δομή της δοθείσας οντολογίας. Οι χειριστές sqwrl:count και sqwrl:countDistinct υπολογίζουν τον αριθμό των σχετιζόμενων αντικειμένων που ταιριάζουν στο ερώτημα, και όχι τον αριθμό των αντικειμένων της οντολογίας που ερωτάται. Παραδείγματος χάριν, το προηγούμενο ερώτημα δεν πρόκειται να ταιριάξει άτομα της κλάσης Person που δεν έχουν αμάξι στην κατοχή τους, επειδή το hasCar(?p, ?c) atom στον κανόνα μας θα έχει ψευδής τιμή για αυτά τα άτομα. Με άλλα λόγια, ο count χειριστής στην SQWRL δεν πρόκειται ποτέ να γυρίσει τιμή 0. Επίσης, αυτός ο χειριστής υιοθετεί την έννοια του unique name assumption για τα άτομα που ταιριάζει, οπότε κάθε άτομο που καταμετράει είναι ξεχωριστό (ακόμα και αν στην σχετιζόμενη οντολογία αυτά τα πολλαπλά ονόματα ίσως να αναφέρονται στο ίδιο άτομο)
  - **Συνάθροιση (Aggregation)**, δυνατότητα απλής συνάθροισης παρέχεται μέσω των τεσσάρων χειριστών, sqwrl:min, sqwrl:max, sqwrl:sum και sqwrl:avg, οι οποίοι δέχονται ένα μόνο argument αριθμητικού τύπου. Παραδείγματος χάριν, για τον υπολογισμό του μέσου όρου ηλικίας όλων των ανθρώπων σε μια οντολογία, μπορούμε να γράψουμε:
    ```
    Person(?p) ^ hasAge(?p, ?age) -> sqwrl:avg(?age)
    ```
    Οποιαδήποτε αριθμητική μεταβλητή δεν έχει αναφερθεί από έναν sqwrl:select χειριστή μπορεί να συναθροιστεί. Αν προσπαθήσουμε να συναθροίσουμε κάποια μεταβλητή που έχει ήδη διέρθει από τον sqwr:select χειριστή θα εμφανιστεί ένα μήνυμα λάθους

- **Ομαδοποίηση (Grouping)**, οι χειριστές καταμέτρησης και συνάθροισης μπορούν να εφαρμοστούν σε ομαδοποιημένες οντότητες που καθορίζονται στο sqwrl:select clause (ο όρος clause αποτελεί μια ρήτρα που καθορίζει το group entity). Παραδείγματος χάριν, για την καταμέτρηση των ονομάτων κάθε ανθρώπου που εμφανίζονται σε μια οντολογία, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
                                   sqwrl:count(?name)
```

Αυτό το ερώτημα θα επιστρέψει μια λίστα με τα ονόματα των ατόμων και τον αριθμό εμφάνισής τους, στην οποία κάθε άτομο μαζί με την καταμέτρησή του, αντιστοιχεί σε μια γραμμή της λίστας. Άτομα που δεν έχουν όνομα δεν πρόκειται να ταιριαστούν από αυτό το ερώτημα. Όταν οι χειριστές sqwrl:count και συνάθροισης χρησιμοποιούνται σε ένα ερώτημα με τον χειριστή sqwrl:select, όλες οι μεταβλητές που αναφέρονται στον χειριστή sqwrl:select δημιουργούν ένα ενιαίο αποδοτικό σύνολο. Αυτό συνεπάγεται στο γεγονός ότι οποιαδήποτε ίδια γραμμή τιμών συγχωνεύεται, μια λειτουργία που είναι παρόμοια με το GROUP BY clause της SQL

- **Ταξινόμηση αποτελεσμάτων**, είναι εφικτή μέσω της χρησιμοποίησης των χειριστών sqwrl:orderBy και sqwrl:orderByDescending. Παραδείγματος χάριν, για την ταξινόμηση των αποτελεσμάτων βάση του αριθμού των αμαξιών που έχει κάθε άνθρωπος κατά φθίνουσα σειρά, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) ^ hasCar(?p, ?c) ->
      sqwrl:select(?name) ^ sqwrl:count(?c) ^
            sqwrl:orderByDescending(?c)
```

Σε περίπτωση που προσπαθήσουμε να συνδυάσουμε αύξουσα και φθίνουσα ταξινόμηση στον ίδιο κανόνα, θα εμφανιστεί ένα μήνυμα λάθους

- **Επιλογή υποσυνόλου αποτελεσμάτων**, η SQWRL παρέχει χειριστές για την επιλογή και τον περιορισμό ενός μόνο υποσυνόλου των αποτελεσμάτων ενός ερωτήματος. Ο χειριστής sqwrl:limit επιτρέπει στους χρήστες να περιορίσουν τα αποτελέσματα σε συγκεκριμένο αριθμό γραμμών. Παίρνει ένα μόνο ακέραιο argument που καθορίζει τον αριθμό των γραμμών που επιστρέφονται. Αν τα αποτελέσματα είναι μη ταξινομημένα, η επιλογή των γραμμών θα είναι αυθαίρετη. Παραδείγματος χάριν, για ένα ερώτημα που περιορίζει τα ονόματα που επιστρέφονται σε δύο, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
                                   sqwrl:limit(2)
```

Υπάρχουν επίσης χειριστές για την επιλογή υποσυνόλου ταξινομημένων αποτελεσμάτων. Σε αυτούς περιλαμβάνονται οι sqwrl:firstN, sqwrl:lastN, καθώς και οι αρνητικές τους μορφές, sqwrl:notFirstN και sqwrl:notLastN. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει το πρώτο όνομα βάση αλφαβητικής σειράς, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
            sqwrl:orderBy(?name) ^ sqwrl:firstN(1)
```

Ενώ, για ένα ερώτημα που επιστρέφει όλα τα ονόματα εκτός του τελευταίο και το προτελευταίο βάση αλφαβητικής σειράς, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
           sqwrl:orderBy(?name) ^ sqwrl:notLastN(2)
```

Η SQWRL παρέχει εναλλακτικά και τα sqwrl:leastN, sqwrl:greatestN, sqwrl:notLeastN, και sqwrl:notGreatestN, αντίστοιχα. Επίσης, είναι δυνατή η επιλογή μιας αυθαίρετης σειράς αποτελεσμάτων χρησιμοποιώντας τον χειριστή sqwrl:nth, που παίρνει ένα μόνο ακέραιο argument το οποίο είναι ο δείκτης της επιλεγμένης γραμμής. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει το τρίτο όνομα βάση αλφαβητικής σειράς, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
             sqwrl:orderBy(?name) ^ sqwrl:nth(3)
```

Επιπλέον, ο χειριστής sqwrl:nthLast επιτρέπει την επιλογή τιμών βάση του μεγαλύτερου ή του τελευταίου στοιχείου σε μια συλλογή. Τα built-ins sqwrl:notNth και sqwrl:notNthLast αποτελούν τις αρνητικές μορφές των χειριστών αυτών. Παραδείγματος χάριν, για ένα

ερώτημα που επιστρέφει όλα τα ονόματα βάση αλφαβητικής σειράς, εκτός το προτελευταίο, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
        sqwrl:orderBy(?name) ^ sqwrl:notNth(2)
```

Τέλος, η SQWRL παρέχει χειριστές όπως οι sqwrl:nthSlice και sqwrl:nthLastSlice που επιλέγουν ένα εύρος στοιχείων. Η πρώτη παράμετρος που δέχονται, είναι ένας δείκτης για την αρχική κοπή, ενώ η δεύτερη παράμετρος είναι το πλήθος των στοιχείων που θα επιλεχθούν. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει το δεύτερο και τρίτο όνομα των ανθρώπων βάση αλφαβητικής σειράς, μπορούμε να γράψουμε:

```
Person(?p) ^ hasName(?p, ?name) -> sqwrl:select(?name) ^
        sqwrl:orderBy(?name) ^ sqwrl:nthSlice(2, 2)
```

Φυσικά, παρέχεται ο χειριστής sqwrl:notNthGreatestSlice καθώς και οι αρνητικές μορφές, sqwrl:notNthSlice και sqwrl:notNthLastSlice που υποστηρίζουν την επιλογή στοιχείων εκτός του καθορισμένου εύρους

- **Στήλες αποτελεσμάτων**, εξ' ορισμού οι στήλες στα αποτελέσματα κάθε ερωτήματος ονομάζονται αυτόματα. Αν πρόκειται για στήλες που πραγματοποιούν επιλογή, έχουν το όνομα της σχετιζόμενης μεταβλητής. Αν πρόκειται για στήλες συνάθροισης, έχουν το όνομα της λειτουργίας που εκτελούν με την σχετιζόμενη μεταβλητή σε παρένθεση, κοκ. Έχοντας ως δεδομένο το παρακάτω ερώτημα, προκύπτουν τρείς στήλες με ονόματα "?name", "[Number of cars]", και "count(?c)":

```
Person(?p) ^ hasName(?p, ?name) ^ hasCar(?p, ?c) ->
sqwrl:select(?name, "Number of cars") ^ sqwrl:count(?c)
```

Στην περίπτωση που ο χρήστης επιθυμεί να καθορίσει ο ίδιος τα ονόματα των στηλών χρησιμοποιείται ο χειριστής sqwrl:columnNames. Ο συγκεκριμένος χειριστής δέχεται μια λίστα από αλφαριθμητικά arguments τα οποία χρησιμοποιεί για την ονομασία των στηλών που φέρουν τα αποτελέσματα. Η απόδοση των ονομάτων γίνεται από αριστερά προς τα δεξιά. Αν δοθούν λιγότερα arguments από τις στήλες, τότε οι εναπομείναντες θα έχουν το αυτόματο όνομα που παράγεται. Αν δοθούν περισσότερα, τα εναπομείναντα arguments θα αγνοηθούν. Παραδείγματος χάριν, το προηγούμενο ερώτημα με δοθέντα από τον χρήστη ονόματα, μπορούσε να γραφτεί ως εξής:

```
Person(?p) ^ hasName(?p, ?name) ^ hasCar(?p, ?c) ->
sqwrl:select(?name, "Number of cars") ^ sqwrl:count(?c) ^
    sqwrl:columnNames("Name", "Description", "Count")
```

Όπως είναι φανερό, οι στήλες των αποτελεσμάτων εμφανίζονται από αριστερά προς τα δεξιά βάση των χειριστών που καθορίζουν την κατάταξη. Επίσης, ο χειριστής sqwrl:select δέχεται και literal τιμές ως arguments, με τις τιμές αυτές να εμφανίζονται ως απλό περιεχόμενο στη γραμμή. Παραδείγματος χάριν, το ακόλουθο ερώτημα θα επιστρέψει το αλφαριθμητικό "Cars and Persons" στα περιεχόμενα της τρίτης στήλης:

```
Person(?p) ^ hasCar(?p, ?c) -> sqwrl:select(?p, ?c, "Cars
                    and Persons")
```

- `Collection Operators`, για την υποστήριξη εξειδικευμένων δυνατοτήτων στα ερωτήματα, η SQWRL έχει μια συλλογή από χειριστές που παρέχουν προηγμένες λειτουργίες ομαδοποίησης, συνάθροισης, περιορισμένες μορφές άρνησης ως διάψευση, και disjunction. Είναι γνωστό ότι η OWL και η SWRL υιοθετούν το open world assumption, με αποτέλεσμα συγκεκριμένα ερωτήματα σε OWL οντολογίες να είναι δύσκολο ή και αδύνατο να γίνουν. Αυτό συμβαίνει επειδή κρίνεται αναγκαία η ύπαρξη closure operators (χειριστές που οδηγούν σε close world assumption) για τον σωστό σχηματισμό γνώμης. Οι Core SQWRL operators υποστηρίζουν closure μέχρι ένα βαθμό, κατά τη διάρκεια εκτέλεσης των ερωτημάτων, και χωρίς να παραβιάζουν την open world assumption της OWL. Όμως ερωτήματα με περίπλοκες closure απαιτήσεις δεν μπορούν να εκφραστούν με core operators, αλλά χρησιμοποιώντας collections operators, δηλαδή χειριστές που δημιουργούν συλλογές στοιχείων:
  - **Basic Collections**, παρέχονται δύο είδη συλλογών, sets και bags, με την ειδοποιό διαφορά να έγκειται στο ότι τα sets δεν επιτρέπουν την δήλωση διπλότυπων στοιχείων ενώ οι bags μπορούν. Για την δημιουργία ενός set χρησιμοποιείται το built-in sqwrl:makeSet που συντάσσεται ως sqwrl:makeSet(<set>, <element>). Το πρώτο argument καθορίζει το set

που πρόκειται να κατασκευαστεί και το δεύτερο καθορίζει το στοιχείο που πρόκειται να προστεθεί σε αυτό το set. Αυτό το built-in θα χτίσει ένα set για κάποιο ερώτημα και θα τοποθετήσει το δοθέν στοιχείο σε αυτό το set. Αν μια μεταβλητή τοποθετηθεί στη θέση του στοιχείου, όλες οι σχετιζόμενες οντότητες αυτής της μεταβλητής σε ένα ερώτημα, θα εισαχθούν στο set. Αντίστοιχα, για την δημιουργία μιας bag χρησιμοποιείται το built-in sqwrl:makeBag που συντάσσεται ως sqwrl:makeBag(<bag>, <element>). Το πεδίο δράσης κάθε συλλογής περιορίζεται στο ερώτημα που περιέχεται. Collection operators όπως ο sqwrl:size, μπορούν μετέπειτα να εφαρμοστούν πάνω σε αυτές τις συλλογές, με τα αποτελέσματα τους να είναι διαθέσιμα προς χρησιμοποίηση από άλλα built-ins για την πρόσβαση στα αποτελέσματα αυτών των closure operations. Η γενική σύνταξη ενός SQWRL ερωτήματος που χρησιμοποιεί τέτοιους collection operators είναι η ακόλουθη:

```
<SWRL Pattern Specification> °
<Collection Construction Clause> °
<Collection Operation Clause>
→ <Select Clause>
```

Το collection construction clause ακολουθεί μετά από ένα προκαθορισμένο SWRL σχήμα, διαχωρίζεται με τον ° χαρακτήρα, και χρησιμοποιεί μόνο χειριστές δημιουργίας συλλογών όπως οι sqwrl:makeSet και sqwrl:makeBag. Το collection operation clause ακολουθεί μετά το collection construction clause, διαχωρίζεται με τον ° χαρακτήρα, χρησιμοποιεί μόνο built-ins που λειτουργούν σε συλλογές όπως ο sqwrl:size, και δεν είναι συμβατό με κανένα άλλο τύπο SWRL atom. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει το πλήθος των ανθρώπων σε μια οντολογία, μπορούμε να γράψουμε:

```
Person(?p) ° sqwrl:makeSet(?s, ?p) ° sqwrl:size(?size, ?s) →
                    sqwrl:select(?size)
```

Επίσης, η SQWRL διαθέτει χειριστές για τα set όπως οι sqwrl:union, sqwrl:difference και sqwrl:intersection, για την ενασχόληση με set semantics και την παραγωγή set. Τέλος, ένας ακόμα χειριστής, ο sqwrl:append, χρησιμοποιείται για την πρόσθεση δύο συλλογών με την επακόλουθη δημιουργία μιας bag.

Η διαδικασία τοποθέτησης στοιχείων σε συλλογές αποτελεί έναν αποδοτικό closure μηχανισμό. Η εισαγωγή του νέου διαχωριστικού χαρακτήρα ° δεν αποτρέπει την χρήση της βασικής SWRL serialization για τα SQWRL ερωτήματα που τον χρησιμοποιούν. Ο συγκεκριμένος χαρακτήρας δεν χρειάζεται να αποθηκευτεί, δεν εμφανίζεται στον panel εικονιδίων του SWRL Editor, και η θέση που ανήκει συμπεραίνεται από τον Editor κατά τη διάρκεια του serialization. Η σημασία υποστήριξης κοινού serialization, εμπίπτει στην δυνατότητα αποθήκευσης όλων των SQWRL ερωτημάτων στις OWL οντολογίες μαζί με τους κανόνες, και τον διαμοιρασμό τους σε εργαλεία OWL ακόμα και αν δεν υποστηρίζουν την γλώσσα

- **Ομαδοποίηση (Grouping)**, για την δυνατότητα εκτέλεσης πιο πολύπλοκων ερωτημάτων που ομαδοποιούν τα σχετιζόμενα set οντοτήτων. Αυτή την λειτουργία διατελεί το built-in sqwrl:groupBy έχοντας ως γενική συντακτική μορφή:

```
sqwrl:makeSet(<set>, <element>) ^ sqwrl:groupBy(<set>,
                        <group>)
                            ή
sqwrl:makeBag(<bag>, <element>) ^ sqwrl:groupBy(<bag>,
                        <group>)
```

Αυτή η ομαδοποίηση μπορεί να περιέχει μια ή περισσότερες οντότητες. Το πρώτο argument είναι η συλλογή και το δεύτερο – ή περισσότερα επακόλουθα arguments - είναι οι οντότητες που θα ομαδοποιηθούν. Μόνο μια ομαδοποίηση μπορεί να εφαρμοστεί σε κάθε συλλογή. Ο μηχανισμός ομαδοποίησης είναι παρόμοιος με το GROUP BY clause της SQL. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει μια συλλογή με τα φάρμακα που παίρνει κάθε άνθρωπος, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) °
   sqwrl:makeSet(?s, ?d) ^ sqwrl:groupBy(?s, ?p) →
                sqwrl:select(?p, ?d)
```

Στο παραπάνω ερώτημα δημιουργείται μια συλλογή κάθε φορά που κάποιος άνθρωπος ικανοποιεί την συνθήκη – παίρνει κάποιο φάρμακο – και όλα τα φάρμακα για κάθε άνθρωπο προστίθενται στην αντίστοιχη συλλογή τους. Ένα εναλλακτικό ερώτημα που θα επέστρεφε όλους τους ανθρώπους που παίρνουν πάνω από ένα φάρμακα, θα ήταν με την χρησιμοποίηση του χειριστή sqwrl:size, ο οποίος θα εφαρμοζόταν ξεχωριστά σε κάθε ομαδοποιημένη συλλογή:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) ˚
   sqwrl:makeSet(?s, ?d) ^ sqwrl:groupBy(?s, ?p) ˚
   sqwrl:size(?n, ?s) ^ swrlb:greaterThan(?n, 1) →
                  sqwrl:select(?p)
```

Όσο αυξάνεται η πολυπλοκότητα της ομαδοποίησης θα απαιτούνται περισσότερες ομαδοποιήσεις οντοτήτων. Παραδείγματος χάριν, για ένα ερώτημα που φτιάχνει bags που περιέχουν τις δόσεις κάθε φαρμάκου που παίρνει κάθε άνθρωπος, όπου κάθε φάρμακο αποθηκεύεται στην θεραπεία μαζί με την δόση του, απαιτείται οι bags να ομαδοποιηθούν μαζί με τους ανθρώπους και τις θεραπείες:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) ^
                  hasDose(?t, ?dose) ˚
   sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p, ?d) →
                sqwrl:select(?p, ?d, ?t, ?b)
```

Μέσω αυτής της προσέγγισης, για ένα ερώτημα που επιστρέφει τον αριθμό των δόσεων κάθε φαρμάκου που παίρνει κάθε άνθρωπος, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) ^
                  hasDose(?t, ?dose) ˚
   sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p, ?d) ˚
                  sqwrl:size(?n, ?b) →
                sqwrl:select(?p, ?d, ?n)
```

Ο μηχανισμός ομαδοποίησης της SQWRL επεκτείνει δραματικά την δύναμη αυτής της γλώσσας. Επιτρέπει στα ερωτήματα να εκτελούν closure μεθόδους, μέσω επιλεκτικού διαμερισμού OWL οντοτήτων σε συλλογές. Έπειτα, υποστηρίζει μια σειρά από τυπικούς collection operators για αυτές τις απομονωμένες οντότητες, που δίνουν την δυνατότητα στον χρήστη να ανακτήσει γνώση από πολύπλοκα ερωτήματα

- **Συνάθροιση (Aggregation)**, η SQWRL υποστηρίζει την χρησιμοποίηση χειριστών συνάθροισης πάνω στις συλλογές που έχουν μια φυσική ταξινόμηση. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει όλους του ανθρώπους σε μια οντολογία που έχουν ηλικία κάτω του μέσου όρου, μπορούμε να γράψουμε:

```
Person(?p) ^ hasAge(?p, ?age) ˚
   sqwrl:makeBag(?b, ?age) ˚
sqwrl:avg(?avg, ?b) ^ swrlb:lessThan(?age, ?avg)
   → sqwrl:select(?p, ?age)
```

Περισσότερο πολύπλοκα ερωτήματα δημιουργούνται συνδυάζοντας τον μηχανισμό ομαδοποίησης με τους χειριστές συνάθροισης. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει την μέγιστη δόση κάθε φαρμάκου για κάθε άνθρωπο, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) ^
                  hasDose(?t, ?dose) ˚
   sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p, ?d) ˚
                  sqwrl:max(?max, ?b)
   → sqwrl:select(?p, ?d, ?max)
```

Ομαδοποιημένες και μη ομαδοποιημένες συλλογές μπορούν να συνυπάρξουν στο ίδιο ερώτημα, καθώς και οποιοσδήποτε αριθμός συλλογών. Παραδείγματος χάριν, για ένα ερώτημα που κάνει μια λίστα με το μέσο όρο των δόσεων των φαρμάκων που αντιστοιχούν στους ανθρώπους που παίρνουν παραπάνω από δύο φάρμακα, και παράλληλα κανένα από αυτά τα φάρμακα δεν ανήκει στην κατηγορία Anesthetic ή στην κατηγορία Psychedelic, μπορούμε να γράψουμε:

```
Person(?p) ^ hasDrug(?p,?drug) ^ hasDose(?drug,?dose) ^
        Anesthetic(?an) ^ Psychedelic(?ps) °
sqwrl:makeSet(?s1, ?dose) ^ sqwrl:groupBy(?s1, ?p, ?drug) ^
   sqwrl:makeSet(?s2, ?drug) ^ sqwrl:groupBy(?s2, ?p) ^
            sqwrl:makeSet(?s3, ?an, ?ps) °
      sqwrl:avg(?avg, ?s1) ^ sqwrl:size(?n, ?s2) ^
            swrlb:greaterThan(?n, 2) ^
  sqwrl:intersection(?s4, ?s2, ?s3) ^ sqwrl:isEmpty(?s4)
          → sqwrl:select(?p, ?drug, ?avg)
```

- **Ανάκτηση όλων των στοιχείων από μια συλλογή**, το built-in sqwrl:element μπορεί να χρησιμοποιηθεί για να προσδιορίσει αν ένα συγκεκριμένο στοιχείο ανήκει σε μια συλλογή ή για την επιλογή όλων των στοιχείων από μια συλλογή. Το πρώτο argument που παίρνει είναι το στοιχείο ενώ το δεύτερο είναι η συλλογή. Αν το πρώτο argument είναι unbound, τότε θα προσδιορίσει κάθε στοιχείο της συλλογής. Επίσης παρέχεται και η αρνητική μορφή sqwrl:notElement. Παραδείγματος χάριν, για ένα ερώτημα που βρίσκει όλους τους ανθρώπους που έχουν παραπάνω από δύο θεραπείες και τουλάχιστον μία από αυτές είναι DPO θεραπεία, μπορούμε να γράψουμε:

```
Person(?p) ^ hasDrug(?p,?d) °
  sqwrl:makeSet(?sd, ?d) ^ sqwrl:groupBy(?sd, ?p) °
sqwrl:size(?sd, ?size) ^ swrlb:greaterThan(?size, 2) ^
          sqwrl:element(DPO, ?sd)
            → sqwrl:select(?p)
```

- **Ανάκτηση συγκεκριμένων στοιχείων από μια συλλογή**, εξ' ορισμού οι συλλογές δεν είναι ταξινομημένες. Όμως, αν τα στοιχεία μιας συλλογής έχουν μια φυσική ταξινόμηση, τότε η SQWRL διαθέτει collection operators για αυτές τις συλλογές. Αυτοί οι χειριστές δέχονται δύο arguments, που αντιστοιχούν στο στοιχείο που πρόκειται να ανακτηθεί και στην συλλογή που βρίσκεται το στοιχείο, αντίστοιχα. Τέτοιοι χειριστές είναι sqwrl:least, sqwrl:greatest, sqwrl:nth, sqwrl:nthLast, sqwrl:first και sqwrl:last. Οι συλλογές που θα δεχτούν αυτούς τους χειριστές ταξινομούνται αυτόματα. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει την χαμηλότερη δόση του φαρμάκου DPO για κάθε άνθρωπο, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?tr) ^ hasDrug(?tr, DPO) ^
                hasDose(?tr, ?dose) °
   sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p) °
sqwrl:least(?leastDose, ?b) ^ swrlb:equal(?leastDose, ?dose)
            → sqwrl:select(?p, ?leastDose)
```

Στην περίπτωση που το στοιχείο που αναφέρεται από το πρώτο argument του built-in δεν υπάρχει, τότε το ερώτημα είναι ψευδές. Στην θέση του στοιχείου μπορεί να τοποθετηθεί και bound argument. Παραδείγματος χάριν, για ένα ερώτημα που ανακτά όλους τους ανθρώπους που παίρνουν τουλάχιστον τρεις δόσεις από το φάρμακο DPO, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?tr) ^ hasDrug(?tr, DPO) ^
                hasDose(?tr, ?dose) °
   sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p) °
             sqwrl:least(3.0, ?b)
              → sqwrl:select(?p)
```

- **Ανάκτηση υποσυνόλου μιας συλλογής**, είναι εφικτή με χειριστές που δέχονται τρία arguments. Το πρώτο είναι η συλλογή που θα φέρει το τελικό αποτέλεσμα, το δεύτερο είναι η συλλογή στην οποία θα λάβει μέρος η διαδικασία, και το τρίτο αποτελεί έναν δείκτη για την επιλεγμένη γραμμή. Ο σημαντικότερος χειριστής είναι ο sqwrl:nthSlice, που υποστηρίζει την επιλογή μιας συλλογής καθορισμένου μεγέθους, αρχίζοντας από μια αυθαίρετη γραμμή του αποτελέσματος. Παραδείγματος χάριν, για ένα ερώτημα που υπολογίζει το μέσο όρο της τρίτης μέχρι και της πέμπτης χαμηλότερης δόσης του φαρμάκου DPO για κάθε άνθρωπο, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?tr) ^ hasDrug(?tr, DPO) ^
                    hasDose(?tr, ?dose) °
      sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p) °
 sqwrl:nthSlice(?thirdToFifthB, ?b, 3, 2) ^ sqwrl:avg(?avg,
                    ?thirdToFifthB)
                  → sqwrl:select(?p, ?avg)
```

Επίσης, η SQWRL υποστηρίζει τους χειριστές sqwrl:greatestN και sqwrl:leastN, που μπορούν να χρησιμοποιηθούν για την επιλογή μιας αλληλουχίας τιμών από μια συλλογή. Παραδείγματος χάριν, ένα ερώτημα πολύπλοκης φύσης που υπολογίζει τον μέσο όρο των δύο χαμηλότερων και των δύο υψηλότερων δόσεων του φαρμάκου DPO είναι ως εξής:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDose(?t, ?dose) ^
                    hasDrug(?t, DPO) °
      sqwrl:makeBag(?s, ?dose) ^ sqwrl:groupBy(?s, ?p) °
            sqwrl:leastN(?lowest2B, ?s, 2) ^
         sqwrl:greatestN(?greatest2B, ?s, 2) ^
       sqwrl:avg(?avgL2, ?lowest2B) ^ sqwrl:avg(?avgG2,
                    ?greatest2B)
                  → sqwrl:select(?p, ?avgL2, ?avgG2)
```

Συλλογές που προκύπτουν από αυτούς τους χειριστές μπορούν να συνδυαστούν μέσω των κλασσικών χειριστών sqwrl:union, sqwrl:difference, sqwrl:intersection και sqwrl:append, ενώ ο χειριστής sqwrl:element μπορεί να χρησιμοποιηθεί για την εξαγωγή στοιχείων από αυτές τις συλλογές

- **Αρνητική μορφή των selection operators**, είναι χειριστές που επιστρέφουν συλλογές που δημιουργήθηκαν από στοιχεία μιας επιλεγμένης συλλογής που δεν ικανοποιούν τα κριτήρια που καθορίζονται στον αντίστοιχο θετικό χειριστή. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει όλες τις δόσεις του φαρμάκου DPO για κάθε άνθρωπο εκτός από την τρίτη χαμηλότερη, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, DPO) ^
                    hasDose(?t, ?dose) °
      sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p) °
sqwrl:notNth(?notThirdB, ?b, 3) ^ swrlb:contains(?notThirdB,
                    ?e)
                  → sqwrl:select(?p, ?e)
```

Επίσης, η SQWRL υποστηρίζει τους χειριστές sqwrl:notGreatestN, sqwrl:notLeastN, sqwrl:notNthSlice και sqwrl:notNthLastSlice, για την επιλογή στοιχείων που δεν ανήκουν σε ένα συγκεκριμένο εύρος. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει όλες τις δόσεις του φαρμάκου DPO για κάθε άνθρωπο εκτός από τις τρεις χαμηλότερες, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?tr, DPO) ^
                    hasDose(?t, ?dose) °
      sqwrl:makeBag(?b, ?dose) ^ sqwrl:groupBy(?b, ?p) °
          sqwrl:notLeastN(?notLeast3DosesC, ?b, 3) ^
             swrlb:contains(?notLeast3DosesC, ?e)
                  → sqwrl:select(?p, ?e)
```

Η SQWRL παρέχει για τους χειριστές sqwrl:notNth, sqwrl:notFirstN, sqwrl:notLastN και sqwrl:notNthLastSlice, τούς εναλλακτικά αντίστροφους χειριστές sqwrl:notNthLast, sqwrl:notLeastN, sqwrl:notGreatestN και sqwrl:notNthGreatestSlice αντίστοιχα

- **Σύγκριση συλλογών**, είναι δυνατή μέσω του χειριστή sqwrl:equal και της αρνητικής μορφής του, sqwrl:notEqual, οι οποίοι δέχονται δύο συλλογές και τις συγκρίνουν. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει τους ανθρώπους που παίρνουν αποκλειστικά και μόνο τα φάρμακα DPO και FDA, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) °
  sqwrl:makeSet(?pds, ?d) ^ sqwrl:groupBy(?pds, ?p) ^
  sqwrl:makeSet(?ds, DPO) ^ sqwrl:makeSet(?ds, FDA) °
            sqwrl:equal(?pds, ?ds)
```

```
→ sqwrl:select(?p)
```

Επίσης, ο χειριστής sqwrl:contains (ή εναλλακτικά sqwrl:notContains) χρησιμοποιείται για να ελέγξει αν μια συλλογή περιέχει όλα τα στοιχεία μιας άλλης συλλογής. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει τους ανθρώπους που παίρνουν τα φάρμακα DPO, FDA, και οποιοδήποτε άλλο περιέχεται στη συλλογή, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, ?d) ˚
 sqwrl:makeSet(?pds, ?d) ^ sqwrl:groupBy(?pds, ?p) ^
 sqwrl:makeSet(?ds, DPO) ^ sqwrl:makeSet(?ds, FDA) ˚
            sqwrl:contains(?pds, ?ds)
                → sqwrl:select(?p)
```

- **Συνεργασία με άλλες built-in βιβλιοθήκες**, όπως προαναφέρθηκε μόνο built-ins μπορούν να χρησιμοποιηθούν στο collection operation clause της SQWRL. Ωστόσο, δεν υπάρχει περιορισμός στον αριθμό των built-ins και μπορεί να χρησιμοποιηθεί οποιαδήποτε βιβλιοθήκη της SWRL. Παραδείγματος χάριν, χρησιμοποιώντας το built-in swrl:eval της βιβλιοθήκης Mathematical Expressions (ενότητα 3.5.6 παρόντος κεφαλαίου), για ένα ερώτημα που εμφανίζει μια λίστα των ανθρώπων που παίρνουν δόσεις του φαρμάκου DPO μεγαλύτερες από το 10% της μέσης δόσης αυτού του φαρμάκου, μπορούμε να γράψουμε:

```
Person(?p) ^ hasTreatment(?p, ?t) ^ hasDrug(?t, DPO) ^
                hasDose(?t, ?dose)  ˚
 sqwrl:makeBag(?bp, ?dose) ^ sqwrl:groupBy(?bp, ?p) ^
            sqwrl:makeBag(?bddi, ?dose)  ˚
  sqwrl:avg(?avgP, ?bp) ^ sqwrl:avg(?avgDPO, ?bddi) ^
swrlm:eval(?r, "(avgP - avgDPO) / avgDPO * 100", ?avgP,
                        ?avgDPO) ^
                swrlb:greaterThan(?r, 10)
            → sqwrl:select(?p, ?avgP, ?avgDPO)
```

Οι collection operators μπορούν να συνεργαστούν και με arguments που είναι bound (αναφέρονται ρητά σε κάποια οντότητα) από built-ins. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει τις κοινές υπερκλάσεις των κλάσεων που ονομάζονται Male και Female, μπορούμε να γράψουμε:

```
tbox:isSuperClassOf(?sm, Male) ^ tbox:isSuperClassOf(?sf,
                        Female) ˚
  sqwrl:makeSet(?sms, ?sm) ^ sqwrl:makeSet(?sfm, ?sf) ˚
sqwrl:intersection(?r, ?sms, ?sfm) ^ sqwrl:element(?e, ?r)
                → sqwrl:select(?e)
```

- **Negation**, με τη μορφή άρνησης ως διάψευση να πραγματοποιείται χρησιμοποιώντας collection operators. Παραδείγματος χάριν - θεωρώντας την ύπαρξη μιας οντολογίας με μια κλάση Drug και διάφορες υποκλάσεις, όπου ανάμεσά τους και η κλάση Anesthetic – για ένα ερώτημα που επιστρέφει τον αριθμό των φαρμάκων χωρίς να συμπεριλάβει στην καταμέτρηση φάρμακα που ανήκουν στην κατηγορία Anesthetic, μπορούμε να γράψουμε:

```
Drug(?d) ^ Anesthetic(?a) ˚
  sqwrl:makeSet(?s1, ?d) ^ sqwrl:makeSet(?s2, ?a) ˚
sqwrl:difference(?s3, ?s1, ?s2) ^ sqwrl:size(?n, ?s3)
                → sqwrl:select(?n)
```

- **Disjunction**, είναι εφικτό μέσω της χρησιμοποίησης του χειριστή set, sqwrl:union. Παραδείγματος χάριν, για ένα ερώτημα που επιστρέφει το πλήθος των αναισθητικών ή των ψυχεδελικών φαρμάκων, μπορούμε να γράψουμε:

```
        Anesthetic(?a) ^ Psychedelic(?p) ˚
 sqwrl:makeSet(?s1, ?a) ^ sqwrl:makeSet(?s2, ?p) ˚
  sqwrl:union(?s3, ?s1, ?s2) ^ sqwrl:size(?n, ?s3)
                → sqwrl:select(?n)
```

Φυσικά, είναι αναγκαίο οι κλάσεις Anesthetic και Psychedelic να είναι υποκλάσεις της Drug στην υποτιθέμενη οντολογία μας

3.5.3 Temporal Built-Ins Library

Καθορίζει έναν αριθμό από built-ins που μπορούν να χρησιμοποιηθούν για να διεξάγουν temporal λειτουργίες. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται στην SWRL Temporal Built-in Library Ontology. Το κοινότυπο πρόθημα είναι το temporal. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής "Import Ontology" στην καρτέλα Metadata του Protégé-OWL. Τα built-ins αυτής της βιβλιοθήκης φέρουν δύο τρόπους εφαρμογής:

- `Basic Mode`, τα built-ins "δουλεύουν" αποκλειστικά με temporal πληροφορία που καθορίζεται από το XML Schema date και dateTime types, και καθορίζουν αν αυτή η πληροφορία ικανοποιεί το σχετιζόμενο temporal κατηγόρημα για μια συγκεκριμένη περιγραφή που δόθηκε (γνωστό ως granularity argument). Αυτό το argument παρέχεται ως xsd:string στα built-ins που το χρειάζονται, με πιθανή τιμή: Years, Months, Days, Hours, Minutes, Seconds ή Milliseconds. Αν δεν δοθεί κάποιο granularity, τότε το πιο ακριβές granularity που υποστηρίζεται από την βιβλιοθήκη (π.χ. milliseconds) θα χρησιμοποιηθεί. Το ίδιο θα συμβεί και στην περίπτωση ημιτελών XML Schema date και dateTime types. Για παράδειγμα, το 2001-11-03T11 επιτρέπεται ως τιμή dateTime, με το 2001-11-03T11:00:00.000 να συμπληρώνεται αυτόματα
- `Advanced Mode`, τα built-ins "δουλεύουν" με temporal πληροφορία που καθορίζεται από κάποιο valid-time temporal model. Η μέθοδος λειτουργεί πάνω στην κωδικογραφημένη temporal πληροφορία βάση του μοντέλου, και προσδιορίζει αν η πληροφορία ικανοποιεί το σχετιζόμενο temporal κατηγόρημα για ένα συγκεκριμένο granularity. Η SWRL Valid-Time Temporal Ontology καθορίζει ένα temporal μοντέλο που μπορεί να χρησιμοποιηθεί για να μοντελοποιήσει ένα πολύπλοκο σύστημα από temporal πληροφορίες στις OWL οντολογίες. Επίσης καθορίζει μια βιβλιοθήκη από SWRL built-ins για την εκπλήρωση temporal υπολογισμών πάνω σε πληροφορίες που περιγράφονται χρησιμοποιώντας αυτή την οντολογία

Το πρόθημα παραμένει το ίδιο και στους δύο τρόπους εφαρμογής, όπως και τα built-ins που χρησιμοποιεί αυτή η βιβλιοθήκη:

duration, durationLessThan, durationEqualTo, durationGreaterThan, notDurationLessThan, notDurationEqualTo, notDurationGreaterThan, equals, before, after, meets, metBy, overlaps, overlappedBy, contains, during, starts, startedBy, finishes, finishedBy, notEquals, notBefore, notAfter, notMeets, notMetBy, notOverlaps, notOverlappedBy, notContains, notDuring, notStarts, notStartedBy, notFinishes, notFinishedBy, add

Πίνακας 3. SWRL Temporal Built-ins

3.5.4 ABox Built-Ins Library

Καθορίζει built-ins που μπορούν να χρησιμοποιηθούν για την διενέργεια ερωτημάτων σε ένα OWL ABox. Σε γενικές γραμμές, αυτά τα built-ins θα μπορούσαν να είναι αναγκαία μόνο σε SQWRL ερωτήματα σε συνδυασμό με τα built-ins της βιβλιοθήκης TBox. Το μεγαλύτερο μέρος της λειτουργικότητας που παρέχουν είναι διαθέσιμο απευθείας στην SWRL χωρίς την αναγκαιότητα χρήσης τους. Επίσης, τα built-ins αυτής της βιβλιοθήκης λειτουργούν προς το παρόν μόνο στην asserted πληροφορία μιας οντολογίας – είναι αδύνατο να επιλέξουν γνώση που έχει τεκμηριωθεί (inferred knowledge).

Για τα built-ins που δεν αναφέρουν ρητά κάποιο argument αλλά αυτό αντιπροσωπεύεται από μια απλή μεταβλητή τότε θεωρούμε ότι έχουμε ένα unbound argument. Αυτό έχει ως αντίκτυπο την αλλαγή της συμπεριφοράς των built-ins, άρα και των αποτελεσμάτων που θα μπορούσαν να εξάγουν τα ερωτήματα, αφού προκαλείται μια γενίκευση στην πληροφορία. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την ABox Ontology. Το κοινότυπο πρόθημα είναι το abox. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL,

και να εισαχθεί μέσω της επιλογής “Import Ontology” στην καρτέλα Metadata του Protégé-OWL. Τα ακόλουθα είναι τα built-ins που καθορίζονται από αυτή την βιβλιοθήκη:

- `isIndividual`, προσδιορίζει αν το argument που καθορίζει είναι ένα άτομο (individual). Αν το argument είναι unbound, τότε θα το προσδιορίσει σε όλα τα άτομα στην οντολογία
- `hasProperty`, είναι αληθές αν το άτομο που καθορίζεται από το πρώτο argument έχει τουλάχιστον μια τιμή για την ιδιότητα που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις ιδιότητες που έχουν τουλάχιστον μια τιμή για αυτό το άτομο
- `hasValue`, είναι αληθές αν το άτομο που καθορίζεται από το πρώτο argument έχει μια ιδιότητα που καθορίζεται από το δεύτερο argument με την τιμή που καθορίζεται από το τρίτο argument. Αν το τρίτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις τιμές για αυτήν την ιδιότητα
- `hasIndividual`, είναι αληθές αν η κλάση που καθορίζεται από το πρώτο argument έχει ένα άτομο που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει σε όλα τα άτομα της κλάσης
- `hasClass`, είναι αληθές αν το άτομο που καθορίζεται από το πρώτο argument είναι ένα στιγμιότυπο της κλάσης που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις κλάσεις για τις οποίες μπορεί να καθοριστεί αυτό το άτομο
- `hasURI`, είναι αληθές αν η OWL κλάση, ιδιότητα, ή άτομο που καθορίζεται από το πρώτο argument έχει ένα URI που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει στο URI του resource
- `isLiteral`, προσδιορίζει αν το argument είναι ένα literal
- `notLiteral`, προσδιορίζει αν το argument δεν είναι ένα literal
- `isNumeric`, προσδιορίζει αν το argument είναι αριθμητικό
- `notNumeric`, προσδιορίζει αν το argument δεν είναι αριθμητικό

3.5.5 TBox Built-Ins Library

Καθορίζει built-ins που μπορούν να χρησιμοποιηθούν για την διενέργεια ερωτημάτων σε ένα OWL TBox. Παραδείγματος χάριν, δίνει τη δυνατότητα στους χρήστες να εντοπίσουν όλες τις transitive ιδιότητες σε μια οντολογία. Όλα τα built-ins σε αυτήν την βιβλιοθήκη, επιτρέπουν τον απευθείας συλλογισμό (direct reasoning) γύρω από τις OWL κλάσεις και τις OWL ιδιότητες, κάτι που δεν επιτρέπεται στην OWL ή την SWRL. Δεν θα πρέπει να χρησιμοποιηθούν σε SWRL κανόνες, αλλά μόνο σε SQWRL ερωτήματα. Επίσης, τα ίδια built-ins λειτουργούν προς το παρόν μόνο στην asserted πληροφορία μιας οντολογίας – είναι αδύνατο να επιλέξουν γνώση που έχει τεκμηριωθεί (inferred knowledge).

Για τα built-ins που δεν αναφέρουν ρητά κάποιο argument αλλά αυτό αντιπροσωπεύεται από μια απλή μεταβλητή τότε θεωρούμε ότι έχουμε ένα unbound argument. Αυτό έχει ως αντίκτυπο την αλλαγή της συμπεριφοράς των built-ins, άρα και των αποτελεσμάτων που θα μπορούσαν να εξάγουν τα ερωτήματα, αφού προκαλείται μια γενίκευση στην πληροφορία. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την TBox Ontology. Το κοινότυπο πρόθημα είναι το tbox. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής “Import Ontology” στην καρτέλα Metadata του Protégé-OWL. Τα ακόλουθα είναι τα built-ins που καθορίζονται από αυτή την βιβλιοθήκη:

- `isClass`, προσδιορίζει αν το argument που καθορίζει είναι μια ονομαζόμενη OWL κλάση. Αν το argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις ονομαζόμενες κλάσεις στην οντολογία
- `isProperty`, προσδιορίζει αν το argument που καθορίζει είναι μια ιδιότητα. Αν το argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις ιδιότητες στην οντολογία
- `isObjectProperty`, προσδιορίζει αν το argument που καθορίζει είναι ένα object property. Αν το argument είναι unbound, τότε θα το προσδιορίσει σε όλα τα object properties στην οντολογία

- `isDataProperty`, προσδιορίζει αν το argument που καθορίζει είναι ένα data property. Αν το argument είναι unbound, τότε θα το προσδιορίσει σε όλα τα data properties στην οντολογία
- `isTransitiveProperty`, προσδιορίζει αν το argument είναι ένα transitive property
- `isSymmetricProperty`, προσδιορίζει αν το argument είναι ένα symmetric property
- `isFunctionalProperty`, προσδιορίζει αν το argument είναι ένα functional property
- `isInverseFunctionalProperty`, προσδιορίζει αν το argument είναι ένα αντίστροφο transitive property
- `isAnnotationProperty`, προσδιορίζει αν το argument είναι ένα annotation property
- `isDirectSuperClassOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument είναι απευθείας υπερκλάση της κλάσης που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις απευθείας υπερκλάσεις του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isSuperClassOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument είναι υπερκλάση της κλάσης που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις υπερκλάσεις του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isDirectSubClassOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument είναι απευθείας υποκλάση της κλάσης που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις απευθείας υποκλάσεις του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isSubClassOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument είναι υποκλάση της κλάσης που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις υποκλάσεις του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isDirectSuperPropertyOf`, ελέγχει αν η ιδιότητα που καθορίζεται από το πρώτο argument είναι απευθείας υπερ-ιδιότητα της ιδιότητας που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις απευθείας υπερ-ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isSuperPropertyOf`, ελέγχει αν η ιδιότητα που καθορίζεται από το πρώτο argument είναι υπερ-ιδιότητα της ιδιότητας που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις υπερ-ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isDirectSubPropertyOf`, ελέγχει αν η ιδιότητα που καθορίζεται από το πρώτο argument είναι απευθείας υπο-ιδιότητα της ιδιότητας που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις απευθείας υπο-ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isSubPropertyOf`, ελέγχει αν η ιδιότητα που καθορίζεται από το πρώτο argument είναι υπο-ιδιότητα της ιδιότητας που καθορίζεται από το δεύτερο argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις υπο-ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποια)
- `isEquivalentTo`, προσδιορίζει αν οι δύο κλάσεις ή ιδιότητες που καθορίζονται από τα δύο argument, αναπαριστούν ισοδύναμες κλάσεις ή ιδιότητες μεταξύ τους. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις ισοδύναμες κλάσεις ή ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποιες)
- `isDisjointWith`, προσδιορίζει αν οι δύο κλάσεις ή ιδιότητες που καθορίζονται από τα δύο argument, αναπαριστούν μη ισοδύναμες κλάσεις ή ιδιότητες μεταξύ τους. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει σε όλες τις μη ισοδύναμες κλάσεις ή ιδιότητες του δεύτερου argument (εφόσον υπάρχει κάποιες)
- `isInRangeOf`, προσδιορίζει αν η κλάση που καθορίζεται από το πρώτο argument αποτελεί μέρος του range μιας ιδιότητας που καθορίζεται από το δεύτερο argument, περιλαμβάνοντας και τις υπερ-ιδιότητες της. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει στο range του δεύτερου argument (εφόσον υπάρχει κάποιο)
- `isInDirectRangeOf`, προσδιορίζει αν η κλάση που καθορίζεται από το πρώτο argument αποτελεί μέρος του range μιας ιδιότητας που καθορίζεται από το δεύτερο argument,

αποκλείοντας όμως τις υπερ-ιδιότητες της. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει στο range του δεύτερου argument (εφόσον υπάρχει κάποιο)

- `isInDomainOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument αποτελεί μέρος του domain μιας ιδιότητας που καθορίζεται από το δεύτερο argument, περιλαμβάνοντας και τις υπερ-ιδιότητες της. Αν το πρώτο argument είναι unbound και το δεύτερο argument είναι bound, τότε θα προσδιορίσει το πρώτο argument σε όλα τα domain του δεύτερου argument(εφόσον υπάρχει κάποιο). Αν το πρώτο argument που καθορίζει την κλάση είναι bound και το δεύτερο είναι unbound, τότε θα προσδιορίσει το δεύτερο argument στις ιδιότητες οι οποίες έχουν την κλάση στο domain τους (αν υπάρχει κάποια). Αν είναι και τα δύο arguments unbound, τότε θα προκύψει σφάλμα
- `isInDirectDomainOf`, ελέγχει αν η κλάση που καθορίζεται από το πρώτο argument αποτελεί μέρος του domain μιας ιδιότητας που καθορίζεται από το δεύτερο argument, αποκλείοντας όμως τις υπερ-ιδιότητες της. Αν το πρώτο argument είναι unbound και το δεύτερο argument είναι bound, τότε θα προσδιορίσει το πρώτο argument σε όλα τα domain του δεύτερου argument(εφόσον υπάρχει κάποιο). Αν το πρώτο argument που καθορίζει την κλάση είναι bound και το δεύτερο είναι unbound, τότε θα προσδιορίσει το δεύτερο argument στις ιδιότητες οι οποίες έχουν την κλάση στο domain τους (αν υπάρχει κάποια). Αν είναι και τα δύο arguments unbound, τότε θα προκύψει σφάλμα
- `equalTo`, ελέγχει αν οι δύο κλάσεις ή ιδιότητες που καθορίζονται από τα δύο arguments, αναφέρονται στην ίδια κλάση ή ιδιότητα. Το SWRL construct sameAs μπορεί να χρησιμοποιηθεί για να προσδιορίσει αν δύο OWL άτομα είναι τα ίδια. Παρομοίως το swrlb:equal built-in μπορεί να χρησιμοποιηθεί για να προσδιορίσει αν δύο OWL data values είναι οι ίδιες
- `notEqualTo`, ελέγχει αν οι δύο κλάσεις ή ιδιότητες που καθορίζονται από τα δύο arguments, αναφέρονται σε διαφορετικές κλάσεις ή ιδιότητες. Το SWRL construct differentFrom μπορεί να χρησιμοποιηθεί για να προσδιορίσει αν δύο OWL άτομα είναι διαφορετικά. Παρομοίως το swrlb:notEqual built-in μπορεί να χρησιμοποιηθεί για να προσδιορίσει αν δύο OWL data values είναι διαφορετικές

3.5.6 Mathematical Expressions Built-Ins Library

Καθορίζει built-ins που μπορούν να χρησιμοποιηθούν για την εκτέλεση μαθηματικών πράξεων πέρα των βασικών πράξεων που παρέχει η Core SWRL built-in library. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την SWRLTab Mathematical Ontology. Το κοινότυπο πρόθημα είναι το swrlm. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής “Import Ontology” στην καρτέλα Metadata του Protégé-OWL. Τα ακόλουθα είναι τα built-ins που καθορίζονται από αυτή την βιβλιοθήκη:

- `sqrt`, είναι αληθές αν το πρώτο argument είναι ίσο με την τετραγωνική ρίζα του δεύτερου argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει στην τετραγωνική ρίζα του δεύτερου argument. Παραδείγματος χάριν, swrlm:sqrt(?r, 100)
- `log`, είναι αληθές αν το πρώτο argument είναι ίσο με τον λογάριθμο (με βάση το e) του δεύτερου argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει στον φυσικό λογάριθμο του δεύτερου argument
- `eval`, είναι αληθές αν το πρώτο argument είναι ίσο με την μαθηματική έκφραση που καθορίζεται στο δεύτερο argument, και μετέπειτα δηλώνει τις μεταβλητές αυτής της μαθηματικής έκφρασης ως επακόλουθα arguments. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει στο αποτέλεσμα της μαθηματικής έκφρασης. Παραδείγματος χάριν, swrlm:eval(?r, "(12 * x) / y", ?x, ?y)

Το swrlm:eval built-in χρησιμοποιεί την πιο πρόσφατη δωρεάν έκδοση (2.4.0) του Java Math Expression Parser για τον επικύρωση της ορθής σύνταξης μιας μαθηματικής έκφρασης. Μια αναλυτική περιγραφή για τους τύπους των εκφράσεων που επιτρέπονται, μπορεί να βρεθεί στον ιστότοπο του JEP project. Πάντως, είναι ρυθμισμένος ώστε να υποστηρίζει τις κλασσικούς σταθερούς

συντελεστές, όπως τα pi και e, και τις συνήθεις μαθηματικές συναρτήσεις, όπως τα ln και log. Επίσης υποστηρίζει τεχνικές implicit multiplication. Παραδείγματος χάριν, επιτρέπει εκφράσεις όπως "2 x" να ερμηνευτούν ως "2 * x".

3.5.7 XML Built-Ins Library

Καθορίζει built-ins που μπορούν να χρησιμοποιηθούν για την διενέργεια ερωτημάτων σε XML έγγραφα. Πρόκειται για μια πρώιμη βιβλιοθήκη που βρίσκεται συνεχώς σε ανάπτυξη και μέχρι στιγμής υποστηρίζει μόνο απλά XML έγγραφα. Σε μελλοντικές εκδόσεις θα υπάρξει και υποστήριξη για XPath. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την OWL XML Ontology. Το κοινότυπο πρόθημα είναι το swrlxml. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής “Import Ontology” στην καρτέλα Metadata του Protégé-OWL.

Η OWL mapping ontology περιέχει κλάσεις που μπορούν να χρησιμοποιηθούν για την διατήρηση των περιεχομένων ενός XML εγγράφου. Αποτελείται από τρείς κλάσεις:

- `XMLDocument`, περιέχει μια ιδιότητα ονόματι hasRootElement που έχει ως range ένα μόνο XMLElement, το οποίο καθορίζει το root element του XML εγγράφου. Επίσης περιέχει μια ιδιότητα ονόματι hasElements που περιέχει όλα τα XML elements του εγγράφου
- `XMLElement`, περιέχει μια ιδιότητα ονόματι hasName που καθορίζει το όνομα του XML element και δέχεται αλφαριθμητικά. Επίσης περιέχει τις ιδιότητες hasSubElements που περιέχει τα sub-elements κάθε element, την hasAttibutes που περιέχει οποιαδήποτε attributes σχετίζονται με το XML element, και την hasContent που καθορίζει το περιεχόμενο του element. Namespaces και prefixes καθορίζονται από τις ιδιότητες hasNamespacePrefix και hasNamespaceURI
- `XMLAttribute`, περιέχει τις ιδιότητες hasName που καθορίζει το όνομα του XML attribute, και hasValue που περιέχει μια τιμή για το attribute. Όπως και στην περίπτωση της κλάσης XMLElement, namespaces και pefixes για το attribute καθορίζονται από τις ιδιότητες hasNamespacePrefix και hasNamespaceURI

Η ύπαρξη του παρακάτω built-in που καθορίζεται από αυτή την βιβλιοθήκη κρίνεται αναγκαία, και χρησιμοποιείται μαζί με τα built-ins που προκύπτουν από τις τρείς κλάσεις που αναφέρθηκαν στην προηγούμενη παράγραφο (swrlxml:hasRootElement, swrlxml:hasElements, swrlxml:hasName, swrlxml:hasSubElements, swrlxml:hasAttributes, swrlxml:hasContent, swrlxml:hasNamespacePrefix, swrlxml:hasNamespaceURI):

- `makeXMLDocument`, είναι αληθές αν το πρώτο argument είναι ένα άτομο της κλάσης swrlxml:XMLDocument, που αντιστοιχεί στην OWL XML αναπαράσταση των περιεχομένων ενός XML εγγράφου και το όνομά του καθορίζεται από το URL του δεύτερου argument. Αν το πρώτο argument είναι unbound, τότε θα το προσδιορίσει το άτομο που αντιστοιχεί σε αυτό το έγγραφο

Υποθέτοντας την ύπαρξη τουλάχιστον ενός swrlxml:makeXMLDocument σε ένα σύνολο από ενεργά ερωτήματα, τότε είναι δυνατή η διατύπωση επιπλέων XML ερωτημάτων χωρίς την επανάληψη του swrlxml:makeXMLDocument.

3.5.8 Extensions Built-Ins Library

Καθορίζει κάποια built-ins που βρίσκονται ακόμα σε πειραματικό στάδιο και μπορούν να χρησιμοποιηθούν για να αυξήσουν την εκφραστικότητα της SWRL μέσω εξυπηρετικών μεθόδων. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την SWLRX Ontology. Το κοινότυπο πρόθημα είναι το swrlx. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής "Import Ontology" στην καρτέλα Metadata του Protégé-OWL. Η μόνη μέθοδος που καθορίζεται μέχρι στιγμής από αυτήν την βιβλιοθήκη είναι:

- `makeOWLThing`, παρέχει ένα ελεγχόμενο τρόπο δημιουργίας OWL ατόμων σε ένα κανόνα. Δέχεται δύο ή περισσότερα arguments. Το πρώτο argument θα πρέπει να είναι μια unbound μεταβλητή, στην οποία θα προσδιοριστεί ένα OWL άτομο που δημιουργείται από το built-in. Το δεύτερο argument και οποιοδήποτε επακόλουθό του, αναπαριστούν ένα σχέδιο. Για κάθε ομάδα τιμών που ταιριάζουν σε αυτό το σχέδιο, το built-in θα δημιουργήσει ένα OWL άτομο του τύπου OWL:Thing και θα το προσδιορίσει στο πρώτο argument. Παραδείγματος χάριν, το built-in atom swrlx:makeOWLThing(?x, ?y) θα οδηγήσει στην δημιουργία ενός ατόμου που θα το προσδιορίσει στην μεταβλητή x για κάθε τιμή της μεταβλητής y που ταιριάζει σε ένα κανόνα. Το built-in atom swrlx:makeOWLThing(?x, ?y, ?z) θα οδηγήσει στην δημιουργία ενός ατόμου που θα το προσδιορίσει στην μεταβλητή x για κάθε συνδυασμό των τιμών των μεταβλητών y και z σε ένα κανόνα. Αν το πρώτο argument είναι ήδη bound όταν το built-in καλείται, τότε αυτή η μέθοδος είναι αληθής και κανένα άτομο δεν δημιουργείται. Αυτή η μέθοδος επιτρέπει αποτελεσματικά την περαιτέρω απευθείας εισαγωγή των existential στους SWRL κανόνες. Ενώ η SWRL επιτρέπει θεωρητικά την χρήση existential με τον OWL περιορισμό owl:someValuesFrom, existential σχηματισμοί χρησιμοποιώντας αυτόν τον περιορισμό είναι υπερβολικά άβολοι. Παρόλα αυτά, αν και δεν είναι εύχρηστο, η δημιουργία existential με αυτόν τον τρόπο είναι θεωρητικά ασφαλής. Σε αντίθεση, οι existential που θα δημιουργηθούν χρησιμοποιώντας αυτό το built-in θα είναι πέραν του πεδίου δράσης των OWL classifiers. Για αυτόν τον λόγο, προτείνεται να μην αποθηκεύονται μόνιμα σε μια OWL οντολογία

3.5.9 RDF Built-Ins Library

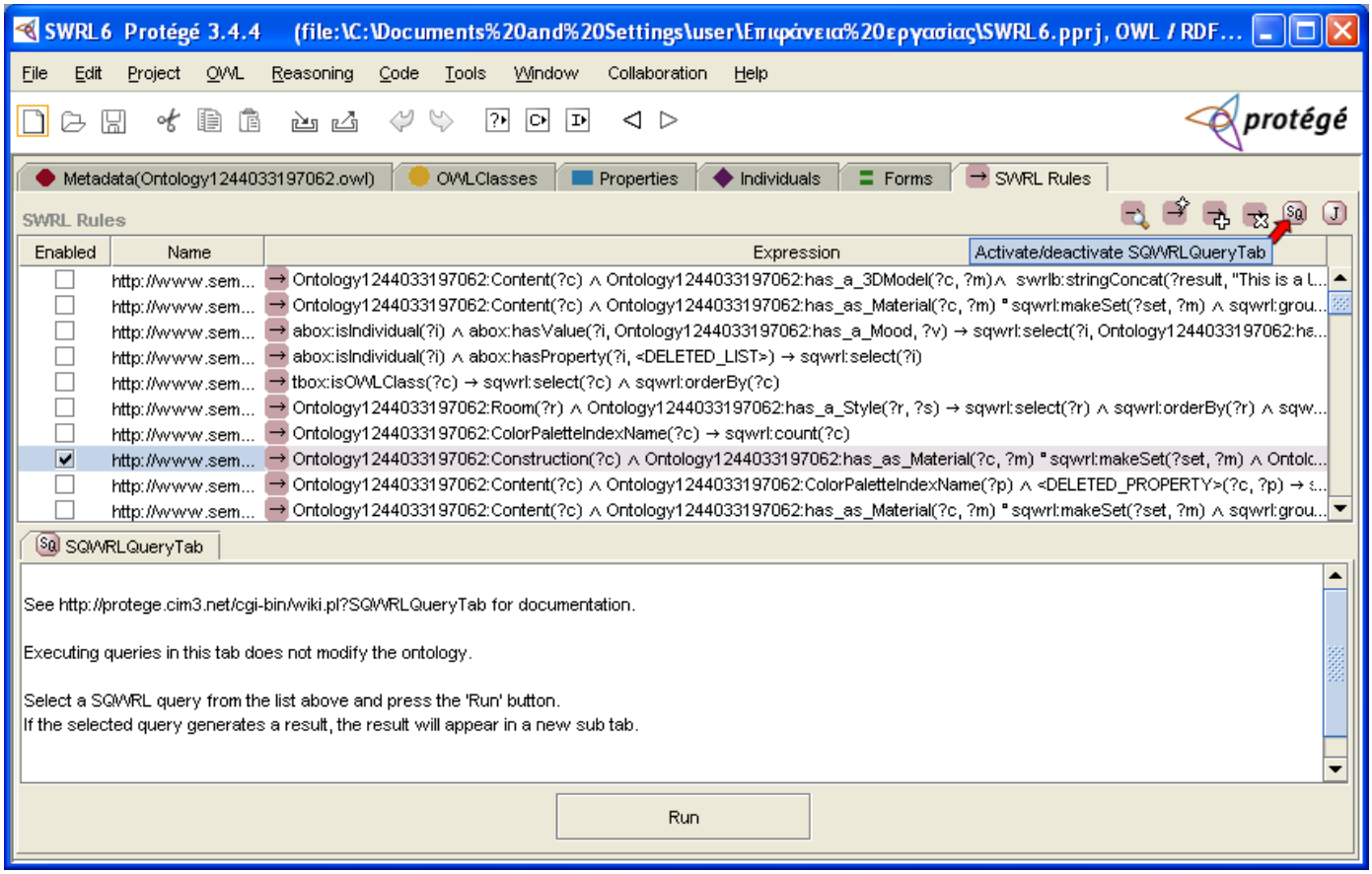
Καθορίζει ένα μικρό αρχικό σύνολο από built-ins για την ενασχόληση με RDF. Αυτή η βιβλιοθήκη σύντομα θα επεκταθεί και θα εμπλουτιστεί με περισσότερα built-ins. Τα built-ins που περιέχονται σε αυτή τη βιβλιοθήκη καθορίζονται από την RDFB Ontology. Το κοινότυπο πρόθημα είναι το rdfb. Ένα αντίγραφο αυτής της οντολογίας μπορεί να βρεθεί στους διαθέσιμους repositories του Protégé-OWL, και να εισαχθεί μέσω της επιλογής "Import Ontology" στην καρτέλα Metadata του Protégé-OWL. Οι ακόλουθες μέθοδοι καθορίζονται από αυτήν την βιβλιοθήκη είναι:

- `hasLabel`, είναι αληθής αν η OWL κλάση, η ιδιότητα, ή το άτομο που καθορίζεται από το πρώτο argument έχει ετικέτα που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει σε όσες ετικέτες έχει το resource. Παραδείγματος χάριν, hasLabel(?i, ?label)
- `hasLabelLanguage`, είναι αληθής αν η OWL κλάση, η ιδιότητα, ή το άτομο που καθορίζεται από το πρώτο argument έχει ως γλώσσα ετικέτας που καθορίζεται από το δεύτερο argument. Αν το δεύτερο argument είναι unbound, τότε θα το προσδιορίσει σε όσες γλώσσες ετικέτας έχει το resource. Παραδείγματος χάριν, hasLabelLanguage(?i, ?language)

3.6 SQWRL Query Tab

Το SWRL Query Tab είναι ένα plug-in στο SWRLTab του Protégé-OWL που υποστηρίζει την εκτέλεση SQWRL ερωτημάτων. Παρέχει έναν εύχρηστο τρόπο απεικόνισης των αποτελεσμάτων των ερωτημάτων σε μια OWL οντολογία. Οι χρήστες που επιθυμούν να ασχοληθούν με τα αποτελέσματα των SQWRL ερωτημάτων σε επίπεδο API μπορούν να χρησιμοποιήσουν την SQWRLQueryAPI.

Το SQWRL Query Tab αποτελεί μέρος του Protégé-OWL 3.4 και δεν χρειάζεται να “κατέβει” και να εγκατασταθεί ξεχωριστά. Παρόλα αυτά, απαιτεί την εγκατάσταση της Jess rule engine, που η άδεια χρήσης της είναι δωρεάν μόνο για ακαδημαϊκή χρήση. Περισσότερες πληροφορίες σχετικά με την εγκατάσταση της Jess βρίσκονται στην ενότητα 3.7.2. Πάντως, σε περίπτωση που δεν υπάρχει η Jess, τότε το SQWRLQueryTab δεν μπορεί να ενεργοποιηθεί. Αν ήδη χρησιμοποιείται επιτυχώς το SWRLJessTab, τότε δεν χρειάζεται καμία επιπλέον εγκατάσταση για την λειτουργία του SQWRLQueryTab. Για την ενεργοποίησή του αρκεί να “πατήσουμε” το εικονίδιο "SQ" που βρίσκεται στην πάνω δεξιά γωνία του SWRL Editor. Τότε, το SQWRLQueryTab θα εμφανιστεί μέσα στο SWRLTab με τη μορφή παραθύρου κάτω από τον SWRL Editor. Σε αυτό το σημείο θα εμφανίζονται τα αποτελέσματα των SQWRL ερωτημάτων. “Πατώντας” άλλη μια φορά το εικονίδιο "SQ", θα κρυφτεί η καρτέλα και θα απενεργοποιηθεί το query tab, κάτι που θα οδηγήσει σε απώλεια όλης της πληροφορίας που έδειχνε.



Εικόνα 5. Ενεργοποίηση του Query Tab

Το SWRLQueryTab έχει μια προκαθορισμένη σειρά εκτέλεσης των SQWRL ερωτημάτων. Αυτό επιτυγχάνεται επιλέγοντας τον κανόνα (εφόσον τα ερωτήματα είναι μια επέκταση των κανόνων) που θέλουμε να εκτελεστεί από τον SWRL Editor και “πατώντας” το κουμπί "Run". Με αυτόν τον τρόπο εκτελείται το SQWRL ερώτημα, και αν παράγει κάποιο αποτέλεσμα, θα εμφανιστεί σε μια νέα καρτέλα που θα ανοίξει δίπλα από την ήδη υπάρχουσα. Οι χρήστες μπορούν να πλοηγηθούν στη νέα καρτέλα για την ανάγνωση των αποτελεσμάτων που παρουσιάζονται σε στήλες και γραμμές. Κάθε νέα καρτέλα φέρει τρία κουμπιά που αντίστοιχα: σώζουν τα αποτελέσματα του κανόνα σε CSV

format (αρχείο που διαχωρίζει τις τιμές του με κόμματα), εκτελούν ξανά τον κανόνα με αποτέλεσα την αναθεώρηση του αποτελέσματος, και κλείνουν την καρτέλα των αποτελεσμάτων. Οποιαδήποτε εκτέλεση κανόνων στο SQWRLQueryTab δεν αλλάζει την OWL οντολογία που έχει φορτωθεί.

3.7 SWRL Jess Bridge

3.7.1 Εισαγωγή

Η SWRL Jess Bridge είναι ένα plug-in στο SWRLTab του Protégé-OWL που υποστηρίζει την εκτέλεση SWRL κανόνων χρησιμοποιώντας την Jess rule engine. Παράλληλα, παρέχεται και το SWRLJessTab που αποτελεί το γραφικό περιβάλλον διάδρασης με αυτούς τους κανόνες. Η SWRL Jess Bridge αποτελεί μέρος του Protégé-OWL 3.4 και δεν χρειάζεται να "κατέβει" και να εγκατασταθεί ξεχωριστά. Όμως, όπως και τα υπόλοιπα μέρη του λογισμικού που απαρτίζουν το SWRLTab, κρίνεται απαραίτητο να χρησιμοποιείται πάντα η πιο πρόσφατη υποέκδοση (η εργασία εκπονήθηκε πάνω στο Protégé-OWL 3.4.4 Build 579). Η SWRL Jess Bridge απαιτεί την εγκατάσταση της Jess rule engine. Σε περίπτωση που η Jess δεν είναι εγκατεστημένη, τότε η SWRL Jess Bridge θα εμφανίσει ένα μήνυμα λάθους όταν ενεργοποιηθεί.

Η SWRL Jess Bridge είναι η μοναδική γέφυρα που υλοποιείται μέχρι στιγμής στο Protégé-OWL, και διαθέτει κάποιους περιορισμούς όσον αφορά την διάδρασή της με την OWL. Ο σημαντικότερος ίσως περιορισμός της, είναι ότι δεν μπορεί να αναπαραστήσει όλα τα OWL Axioms κατά την ανταλλαγή γνώσης από την OWL οντολογία στη γέφυρα. Τα axioms που τώρα υποστηρίζονται είναι μόνο τα ακόλουθα:

- OWL declaration axiom
- OWL class assertion axiom
- OWL property assertion axiom
- OWL subclass axiom
- OWL sub property axiom
- OWL equivalent class axiom
- OWL equivalent property axiom
- OWL different individuals axiom
- OWL same individuals axiom

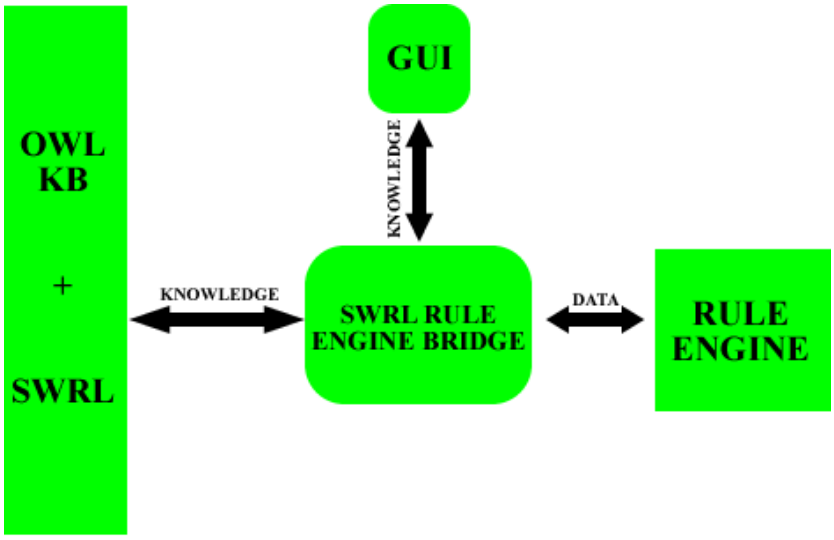
Τα axioms χρησιμοποιούνται για να συσχετίσουν κλάσεις και ιδιότητες με μερικές ή ολόκληρες περιγραφές των χαρακτηριστικών τους, και για να δώσουν άλλη πληροφορία για κλάσεις και ιδιότητες. Τα axioms συνηθίζονταν να αποκαλούνται definitions, αλλά δεν είναι όλα τα definitions και axioms βάση κοινής λογικής του όρου αυτού, οπότε επιλέχθηκε ένα πιο ουδέτερο όνομα.

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

Πίνακας 4. OWL Axioms
Πηγή: http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/

Αυτή η αδυναμία της Jess, οδηγεί τους συμπερασματικούς μηχανισμούς να μην γνωρίζουν για τα εναπομείναντα OWL axioms, με αποτέλεσμα να μην εξάγονται όλα τα πιθανά αποτελέσματα. Επίσης, υπάρχει η πιθανότητα διενέξεων στην OWL οντολογία, ανάμεσα στην πληροφορία που εισήχθη από μια μηχανή κανόνων και στα OWL axioms που δεν λήφθηκαν υπόψιν. Παραδείγματος χάριν, μια ιδιότητα ονόματι hasMother μπορεί να επιβληθεί σε ένα συγκεκριμένο άτομο σε ένα SWRL κανόνα, αλλά ένα OWL axiom ίσως να απαγορέψει τον συσχετισμό αυτής της ιδιότητας με αυτό το άτομο. Προς το παρών, τέτοιες διενέξεις δεν εντοπίζονται αυτόματα, και η επίλυση μιας τέτοιας διένεξης – που στην πραγματικότητα αφορά ένα SWRL κανόνα και ένα ή περισσότερα OWL axioms, και δεν έχει να κάνει με μια συγκεκριμένη rule engine – αφήνεται στον χρήστη. Τυχόν διενέξεις μπορούν να εντοπιστούν εκτελώντας έναν OWL reasoner στην οντολογία που έχει εμπλουτιστεί από συμπερασματική γνώση (inferred knowledge).

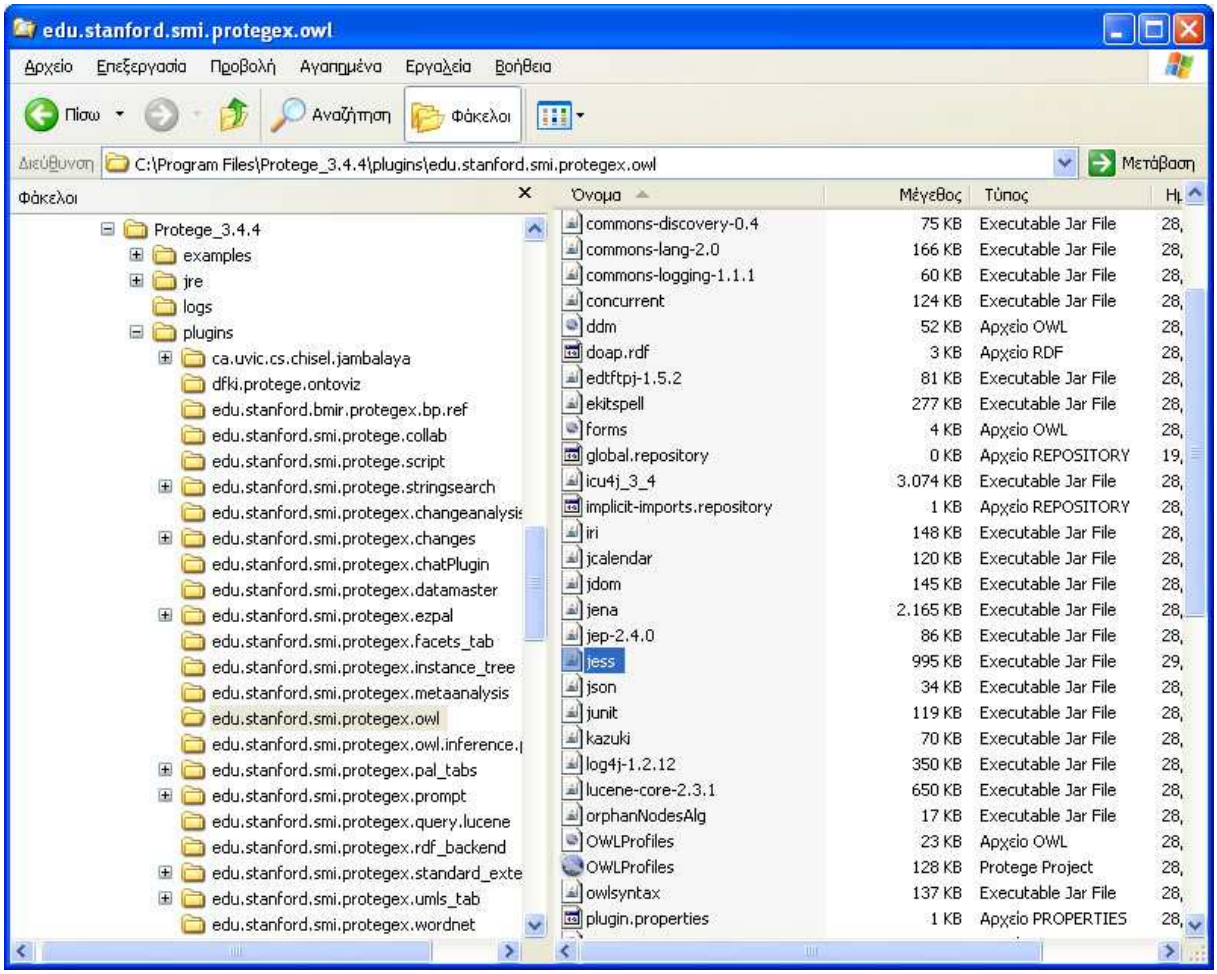
Ένα επιπλέον ζήτημα είναι ότι μια χωρίς διενέξεις εκτέλεση ενός classifier (description logic reasoner που βοηθά στον έλεγχο και στην οργάνωση μιας οντολογίας) μπορεί επίσης να συμπεράνει νέα γνώση, που μπορεί με τη σειρά της να παράγει πληροφορία που μπορεί να επωφεληθεί από περαιτέρω διάδραση με την SWRL, μια διαδικασία που μπορεί να χρειαστεί πολλές επαναλήψεις πριν τη δημιουργία μη επιθυμητής νέας γνώσης. Ο Pellet reasoner λαμβάνει υπόψιν του όλα τα σχετικά OWL axioms όταν πραγματοποιεί συμπερασματική γνώση με SWRL κανόνες, αλλά δεν μπορεί να δουλέψει με τις SWRL built-ins βιβλιοθήκες.



Σχήμα 4. Μια τυπική SWRL Rule Engine Bridge

3.7.2 Εγκατάσταση της Jess Rule Engine

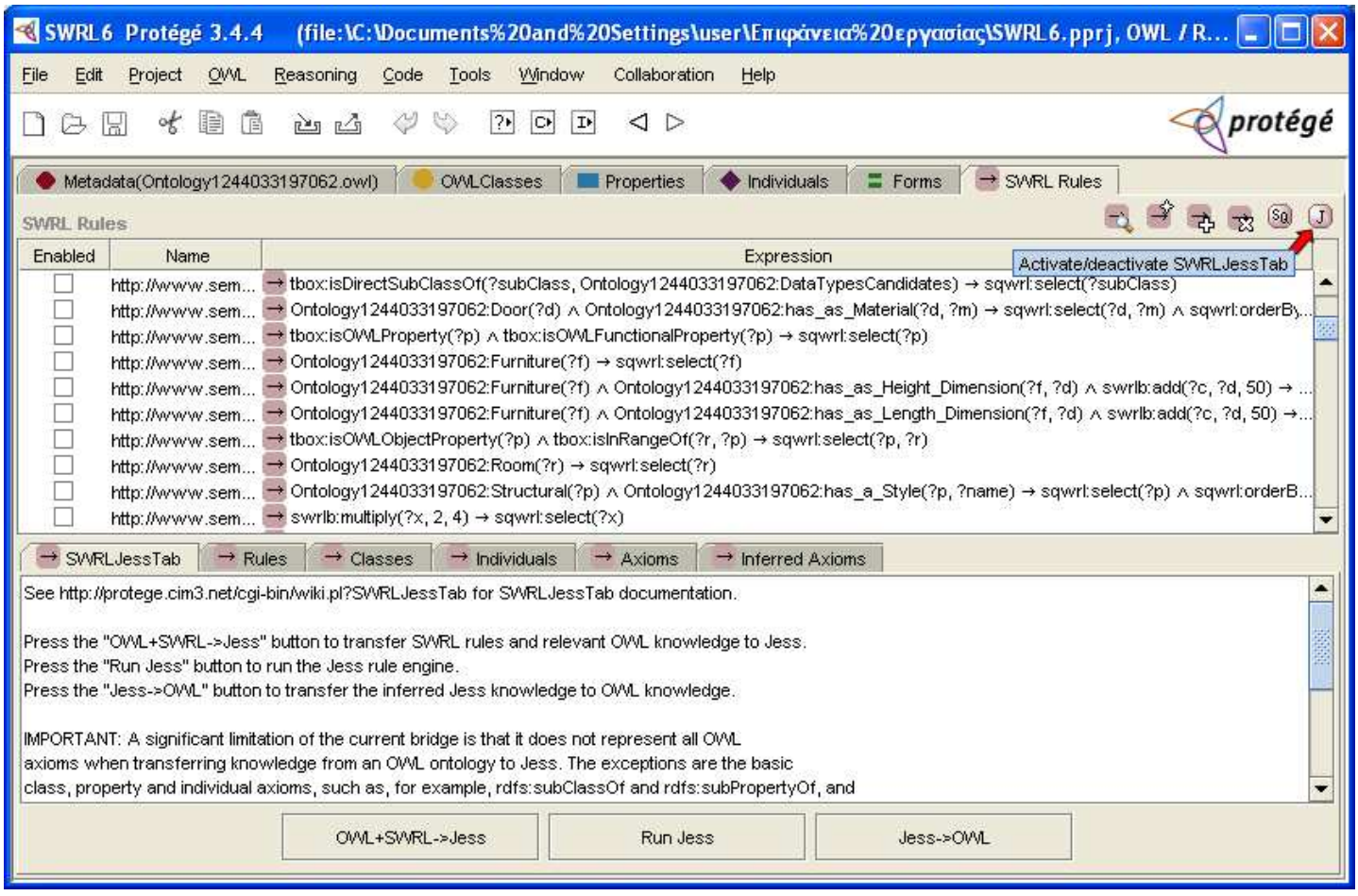
Το SWRLTab απαιτεί την εγκατάσταση της Jess rule engine για την εκτέλεση των SWRL κανόνων και των SQWRL ερωτημάτων. Η Jess rule engine δεν αποτελεί ελεύθερο λογισμικό οπότε πρέπει να "κατέβει" ξεχωριστά. Αυτή η μηχανή περιέχεται σε ένα Java JAR που ονομάζεται jess.jar, το οποίο περιλαμβάνεται στην τυπική διανομή της Jess. Η συγκεκριμένη διανομή μπορεί να βρεθεί στο http://herzberg.ca.sandia.gov/jess/. Έπειτα, αυτό το JAR αρχείο πρέπει να αντιγραφεί μέσα στον υποκατάλογο edu.stanford.smi.protegex.owl που βρίσκεται μέσα στον υποκατάλογο plugins του φακέλου εγκατάστασης του Protégé-OWL (π.χ. ./plugins/edu.stanford.smi.protegex.owl/). Κάθε φορά που το Protégé-OWL θα αρχίζει, τότε θα φορτώνεται αυτόματα και το JAR αρχείο αν είναι παρών σε αυτόν τον υποκατάλογο. Απαιτούμενη κρίνεται η χρήση της έκδοσης Jess 7.1p1 ή κάποιας νεότερης εφόσον υπάρχει. Ακολουθεί μια εικόνα της διαδρομής εγκατάστασης του Jess JAR:



Εικόνα 6. Διαδρομή εγκατάστασης του Jess JAR

3.7.3 SWRL Jess Tab

Το SWRL Jess Tab είναι ένα plugin του SWRLTab μέσα στο Protégé-OWL που υποστηρίζει την εκτέλεση των SWRL κανόνων χρησιμοποιώντας την Jess rule engine. Κύριο χαρακτηριστικό του είναι η παροχή ενός γραφικού περιβάλλοντος για την διάδραση με την SWRLJess Bridge. Το SWRL Jess Tab αποτελεί μέρος του Protégé-OWL 3.4 και δεν χρειάζεται να "κατέβει" και να εγκατασταθεί ξεχωριστά. Όμως, όπως και τα υπόλοιπα μέρη του λογισμικού που απαρτίζουν το SWRLTab, κρίνεται απαραίτητο να χρησιμοποιείται πάντα η πιο πρόσφατη υποέκδοση (η εργασία εκπονήθηκε πάνω στο Protégé-OWL 3.4.4 Build 579). Το SWRL Jess Tab απαιτεί την εγκατάσταση της Jess rule engine. Σε περίπτωση που η Jess δεν είναι εγκατεστημένη, τότε το SWRL Jess Tab θα εμφανίσει ένα μήνυμα λάθους όταν ενεργοποιηθεί. Όταν ενεργοποιηθεί επιτυχώς το SWRLTab, θα εμφανιστεί μια λίστα από εικονίδια στην πάνω δεξιά γωνία του πίνακα των SWRL κανόνων. Το Jess Tab ενεργοποιείται "πατώντας" το εικονίδιο "J". Όταν πατηθεί, το SWRLJessTab θα εμφανιστεί μέσα στο SWRLTab με τη μορφή παραθύρου κάτω από τον SWRL Editor.



Εικόνα 7. Ενεργοποίηση του Jess Tab

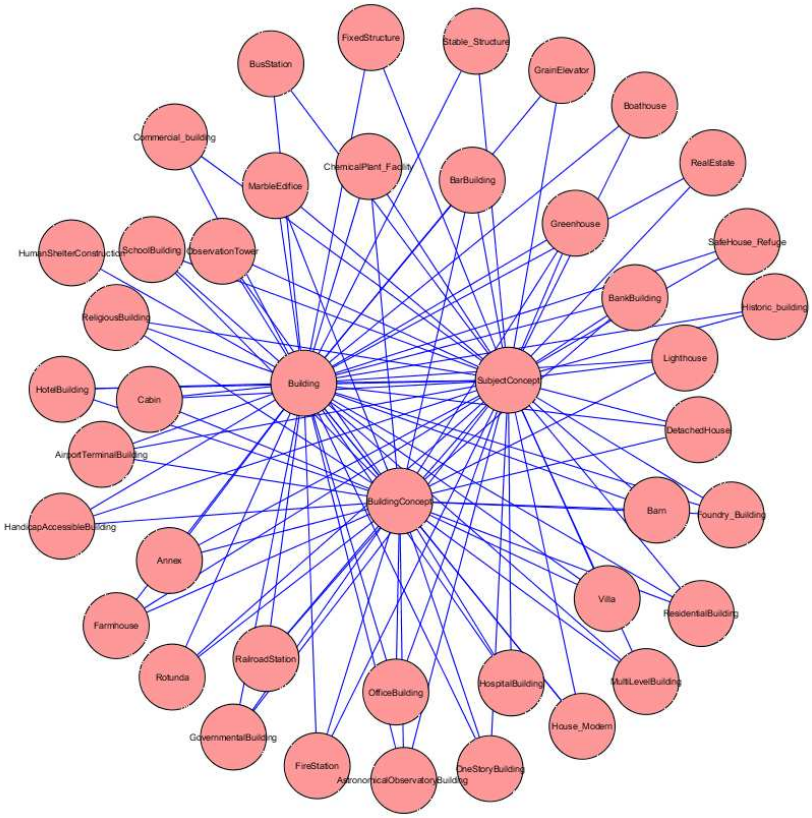
Αν το Jess Tab φορτώσει επιτυχώς την Jess rule engine, θα εμφανιστούν μια σειρά από καρτέλες στο κάτω μέρος του SWRL Editor, που μπορούν να επικοινωνήσουν με την Jess. Σε περίπτωση αποτυχίας, ένα μήνυμα λάθους εμφανίζεται σε αυτό το σημείο υποδεικνύοντας ότι η Jess rule engine δεν βρέθηκε. Αν "πατήσουμε" το εικονίδιο "J" ξανά, θα χαθούν αυτές οι καρτέλες και θα απενεργοποιηθεί η Jess rule engine. Με την απενεργοποίηση του Jess Tab όλες οι πληροφορίες και καταστάσεις των κανόνων που τρέχουν θα χαθούν.

Η επικοινωνία μεταξύ της OWL και της Jess rule engine είναι πλήρως καθοδηγούμενη από τον χρήστη. Ο χρήστης ελέγχει πότε η OWL γνώση και οι SWRL κανόνες θα μεταφερθούν στην Jess,

πότε συμπεραίνεται η νέα γνώση χρησιμοποιώντας την υπάρχουσα και τους κανόνες, και πότε τα αποτελέσματα που προκύπτουν από την Jess μεταφέρονται πίσω στο Protégé-OWL ως OWL γνώση. Το SWRLTab έχει έξι παράθυρα με τη μορφή καρτέλας που μπορούν να χρησιμοποιηθούν για τον έλεγχο και την ανασκόπηση αυτής της επικοινωνίας. Η πρώτη καρτέλα είναι η SWRLJessTab, η οποία παρέχει τον λειτουργικό έλεγχο. Διαθέτει ένα παράθυρο καταγραφής που απεικονίζει την κατάσταση της επικοινωνίας που λαμβάνει μέρος ανάμεσα στον χρήστη και την Jess. Παράλληλα διαθέτει και τρία κουμπιά για τον έλεγχο της διαδικασίας ανταλλαγής γνώσης και συμπεράσματος:

- `OWL + SWRL → Jess`, είναι το κουμπί το οποίο θα μεταφέρει όλους τους SWRL κανόνες και την κατάλληλη OWL γνώση στην Jess rule engine. Το παράθυρο καταγραφής θα δείξει τον αριθμό των κανόνων, των OWL κλάσεων, των OWL ατόμων, και των OWL axioms που έχουν μεταφερθεί. "Πατώντας" άλλη μια φορά το κουμπί, θα σβηστεί όλη η προηγούμενη γνώση που μεταφέρθηκε στην Jess και θα εκκινηθεί εκ νέου η διαδικασία μεταφοράς. Η ύπαρξη των επόμενων τεσσάρων καρτελών στην SWRLJessTab αποσκοπεί στην εξέταση την γνώσης που έχει μεταφερθεί στην Jess. Αυτές είναι οι καρτέλες "Rules", "Classes", "Individuals" και "Axioms" που διαθέτουν μια λίστα με τους SWRL κανόνες, τις OWL κλάσεις, τα OWL άτομα και τα OWL axioms, αντίστοιχα, όπως αναπαριστώνται από την Jess
- `Run Jess`, μετά "πατιέται" αυτό το κουμπί για να ξεκινήσει η διαδικασία με την οποία εξάγονται τα συμπεράσματα. Η Jess θα τρέξει την μηχανή εξαγωγής συμπερασμάτων και πιθανότατα θα δημιουργήσει νέα γνώση, που αναπαρίσταται από τα γεγονότα της Jess. Αυτή η συμπερασματική γνώση μπορεί να εξεταστεί στην έκτη και τελευταία καρτέλα "Inferred Axioms"
- `Jess → OWL`, αυτή η συμπερασματική γνώση μπορεί να μεταβιβαστεί πίσω στην OWL "πατώντας" αυτό το κουμπί. Εκείνη τη στιγμή, η νέα γνώση δεν μπορεί να διακριθεί από την ήδη υπάρχον, αφού αυτό αποτελεί κοινή τακτική της διάδρασης των reasoner με την τεκμηριωμένη γνώση στο Protégé-OWL. Σε μελλοντική έκδοση αναμένεται η επισήμανση αυτής της διαφοράς

4. ΠΕΡΙΓΡΑΦΗ ΟΝΤΟΛΟΓΙΑΣ



4.1 Εισαγωγή

Η εργασία εκπονήθηκε βασισμένη σε μια σημασιολογική οντολογία σχεδίασης και διακόσμησης εσωτερικών χώρων που εξετάζει κάθε άποψη δημιουργίας ενός εσωτερικού χώρου. Η συγκεκριμένη οντολογία εκτός από ποσοτικές και ογκομετρικές πληροφορίες, μπορεί και παρέχει αφηρημένα ποσοτικά χαρακτηριστικά ενός φυσικού αντικειμένου του πραγματικού κόσμου, όπως παραδείγματος χάριν, τυχόν λειτουργικότητα χώρου, επικρατούσα ατμόσφαιρα, ύφος δωματίου, κτλ. Η οντολογία αναπτύχθηκε με τέτοιο τρόπο ώστε να είναι δυνατή η αναλυτική περιγραφή ενός εσωτερικού χώρου, ενώ ταυτόχρονα η γενικευμένη δομή της, επιτρέπει την περιγραφή οποιουδήποτε τύπου δωματίου ενός εσωτερικού χώρου. Η οντολογία αναπτύχθηκε εξ' ολοκλήρου πάνω στην OWL (Web Ontology Language), χρησιμοποιώντας το Protégé-OWL 3.4.4, που επιτρέπει την εκμετάλλευση της οντολογίας μέσω της διακίνησης και της ανταλλαγής της στο διαδίκτυο, και της εφαρμογής προσιτών και κατανοητών αλγορίθμων, όπως αλγόριθμοι γράφων και βάσεων δεδομένων.

Γιατί OWL ;

Το Semantic Web αποτελεί ένα όραμα για το μέλλον του Web κατά το οποίο δίνεται στην πληροφορία λεπτομερής σημασιολογία, κάνοντας για τις μηχανές ευκολότερη την αυτόματη επεξεργασία και την ενσωμάτωση της πληροφορίας που υπάρχει διαθέσιμη στο Web. Το Semantic Web θα χτιστεί στην ικανότητα του XML να καθορίζει tagging schemes και στην ευελιξία του RDF για την αναπαράσταση των δεδομένων. Το πρώτο επίπεδο πάνω από το RDF που απαιτείται για το Semantic Web είναι μια γλώσσα οντολογίας που μπορεί να περιγράψει επίσημα την έννοια της ονοματολογίας που χρησιμοποιείται στα έγγραφα του διαδικτύου. Σε περίπτωση που οι μηχανές προβλέπεται να εκτελέσουν χρήσιμες συλλογιστικές εργασίες σε αυτά τα έγγραφα, η γλώσσα αυτή θα πρέπει να ξεπεράσει τα βασικά σημασιολογικά στοιχεία του RDF Schema.

Η OWL σχεδιάστηκε για να καλύψει την ανάγκη μιας τέτοιας γλώσσας οντολογίας διαδικτύου (Web Ontology Language) και αποτελεί μέρος της συνεχώς αναπτυσσόμενης στοίβας της σύστασης W3C που σχετίζεται με το Semantic Web. Σε αυτή περιέχονται:

- `XML`, παρέχει μια επιφανειακή σύνταξη για την δόμηση εγγράφων, αλλά δεν επιβάλλεται κανένας σημασιολογικός περιορισμός στην έννοια αυτών των εγγραφών
- `XML Schema`, είναι μια γλώσσα για τον περιορισμό στην δομή των XML εγγράφων και επίσης επεκτείνει την XML με datatypes
- `RDF`, είναι ένα μοντέλο δεδομένων για αντικείμενα (resources) και σχέσεις μεταξύ αυτών, το οποίο παρέχει ένα απλό σημασιολογικό αποτέλεσμα, με αυτά τα datamodels να μπορούν να αναπαρασταθούν σε σύνταξη XML
- `RDF Schema`, είναι ένα λεξιλόγιο για την περιγραφή των ιδιοτήτων και των κλάσεων των αντικειμένων του RDF, με σημασιολογία για την γενίκευση ιεραρχιών τέτοιων ιδιοτήτων και κλάσεων

Η OWL προσθέτει επιπλέον λεξιλόγιο για τη περιγραφή ιδιοτήτων και κλάσεων, συσχετίσεων μεταξύ κλάσεων (π.χ. disjointness), αριθμούς στοιχείων συνόλου (π.χ. ακριβώς ένα), ισότητες, πλουσιότερη γραφή ιδιοτήτων, χαρακτηριστικά γνωρίσματα ιδιοτήτων (π.χ. συμμετρία) και enumerated κλάσεις.

Η συγκεκριμένη OWL οντολογία περιέχει ποσοτικές και ποιοτικές πληροφορίες, που καλύπτουν όλο το φάσμα εκείνων των συντελεστών, που κρίνονται απαραίτητοι για την ακριβής περιγραφή ενός εσωτερικού χώρου. Αυτό γίνεται ευκολότερα κατανοητό και εφικτό, μέσω των απαντήσεων που θα δοθούν στα ακόλουθα τέσσερα ερωτήματα:

- Ποια είναι τα αντικείμενα που υπάρχουν στον εσωτερικό χώρο
- Ποια είναι η μορφή αυτών των αντικειμένων
- Που τοποθετούνται αυτά τα αντικείμενα μέσα στον χώρο σε σχέση με άλλα αντικείμενα
- Ποια είναι η “προσωπικότητα” του χώρου

Σε πρώτη φάση, καταγράφηκαν σε ένα XML όλα τα πιθανά σενάρια που μπορούν να λάβουν μέρος σε ένα εσωτερικό χώρο, και καθορίστηκαν οι σχέσεις μεταξύ αντικειμένων και δωματίου. Με αυτόν τον τρόπο αποφεύγουμε τον πληθωρισμό αντικειμένων και επικεντρωνόμαστε στην θέση των αντικειμένων μέσα στον χώρο βάση μιας ιεραρχίας και των ιδιοτήτων των αντικειμένων.

Ο καλύτερος τρόπος για την περιγραφή και την συσχέτιση γνώσης, παρέχεται μέσω της δημιουργίας μιας οντολογίας που μπορεί να συνεργαστεί με προγράμματα και υπηρεσίες διαδικτύου, ώστε να είναι δυνατή η διανομή και η ανταλλαγή της εγγραφείσας γνώσης. Ακολουθώντας αυτήν την διαδικασία μπορεί οποιοσδήποτε εξουσιοδοτημένος (ή απλά ενδιαφερόμενος) χρήστης να συνεισφέρει στην ένταξη εννοιών του πραγματικού κόσμου στην βιομηχανία της επιστήμης των υπολογιστών.

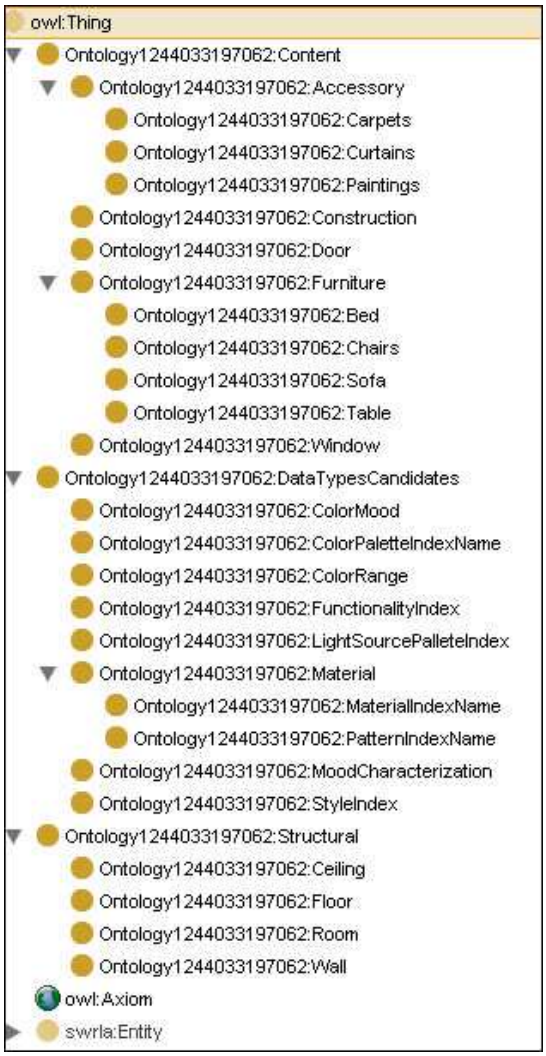
4.2 Δομή Οντολογίας

Η οντολογία έχει ως σημείο αναφοράς ένα δωμάτιο κάθε φορά. Για την εκτενέστερη περιγραφή ενός εσωτερικού χώρου, αναλύσαμε όλα τα φυσικά αντικείμενα που μπορεί να περιέχει (έτσι ώστε κάποιος να μπορεί να καταλάβει τι ακριβώς περιέχει ένα δωμάτιο), τα χαρακτηριστικά κάθε φυσικού αντικειμένου (έτσι ώστε να υπάρχει πληροφορία για το στυλ, τις διαστάσεις, το σχήμα, κτλ), και τελικώς τις σχέσεις μεταξύ αυτών των αντικειμένων (έτσι ώστε κάποιος να μπορεί να καταλάβει την δομή του εσωτερικού χώρου).

4.2.1 Κλάσεις οντολογίας

Η κορυφή της οντολογίας είναι η ρίζα κάθε OWL οντολογίας, που δεν είναι άλλη από την owl:Thing, στην οποία στηρίζονται όλες οι μετέπειτα κλάσεις. Ως απευθείας και μοναδικά παιδιά αυτής της ρίζας, ορίζουμε τρεις υποκλάσεις που συνθέτουν όλα τα πιθανά αντικείμενα που μπορεί να περιέχει ένα δωμάτιο. Έτσι προκύπτει ότι στο ίδιο ιεραρχικά επίπεδο θα βρίσκονται οι υποκλάσεις Content, DataTypesCandidates και Structural. Κάθε μια από αυτές τις υποκλάσεις που απαρτίζουν την κλάση owl:Thing, έχει ένα σύνολο από περαιτέρω υποκλάσεις:

- `Content`, πρόκειται για την κλάση που περιέχει όλα τα φυσικά αντικείμενα που μπορούν να βρεθούν σε ένα εσωτερικό χώρο
  - **Accessory**, πρόκειται για την υποκλάση του Content που περιέχει δικές της υποκλάσεις, τα άτομα των οποίων καθορίζουν ένα πλήθος πιθανών αξεσουάρ ενός δωματίου, όπως χαλιά, κουρτίνες, πίνακες, κτλ
  - **Construction**, πρόκειται για την υποκλάση του Content που περιέχει άτομα που καθορίζουν κατασκευές και όχι δομικά αντικείμενα του χώρου
  - **Door**, πρόκειται για την υποκλάση του Content που περιέχει άτομα που καθορίζουν τις πόρτες ενός δωματίου
  - **Furniture**, πρόκειται για την υποκλάση του Content που περιέχει δικές της υποκλάσεις, τα άτομα των οποίων καθορίζουν ένα πλήθος πιθανών επίπλων ενός δωματίου, όπως τραπέζια, καρέκλες, καναπέδες, κτλ
  - **Window**, πρόκειται για την υποκλάση του Content που περιέχει άτομα που καθορίζουν τα παράθυρα ενός δωματίου
- `DataTypesCandidates`, πρόκειται για την κλάση που απαριθμεί όλα τα ποιοτικά χαρακτηριστικά ενός δωματίου και των αντικειμένων που περιέχονται σε αυτό
  - **ColorMood**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που κατατάσσουν ένα σύνολο από ColorRange άτομα, σε μια αυθαίρετη επικρατούσα ατμόσφαιρα
  - **ColorPaletteIndexName**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν ένα χρώμα
  - **ColorRange**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που κατατάσσουν ένα σύνολο από ColorPaletteIndexName άτομα, σε μια γενικευμένη κατηγορία χρώματος
  - **LightSourcePalleteIndex,** πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζονται ως πηγές φωτός, όπως λάμπες, φωτιστικά, κτλ
  - **FunctionalityIndex,** πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν την λειτουργικότητα του χώρου, όπως υπνοδωμάτιο, κουζίνα, κτλ
  - **Material**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει τις δύο υποκλάσεις MaterialIndexName και PatternIndexName, τα άτομα των οποίων καθορίζουν αντίστοιχα τα υλικά και το σχέδιο κατασκευής ενός αντικειμένου
  - **MoodCharacterization**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν την προσωπικότητα του δωματίου, όπως fresh, relaxing, dynamic, κτλ
  - **StyleIndex**, πρόκειται για την υποκλάση του DataTypesCandidates που περιέχει άτομα που καθορίζουν το στυλ των ατόμων, όπως black, exotic, κτλ
- `Structural`, πρόκειται για την κλάση που περιέχει υποκλάσεις που υποδεικνύουν απαραίτητα άτομα για την δομή και τον σχεδιασμό ενός δωματίου, τα οποία δύσκολα επιδέχονται αλλαγές
  - **Ceiling**, πρόκειται για την υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στην οροφή ενός δωματίου
  - **Floor**, πρόκειται για την υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στο πάτωμα ενός δωματίου
  - **Wall**, πρόκειται για την υποκλάση του Structural που περιέχει άτομα που αντιστοιχούν στους τοίχους ενός δωματίου
  - **Room**, πρόκειται για την υποκλάση του Structural που κάθε φορά περιέχει ένα άτομο, το οποίο αναφέρεται στο δωμάτιο που περιγράφεται στην συγκεκριμένη οντολογία



Εικόνα 8. Η ιεραρχία των κλάσεων της OWL οντολογίας

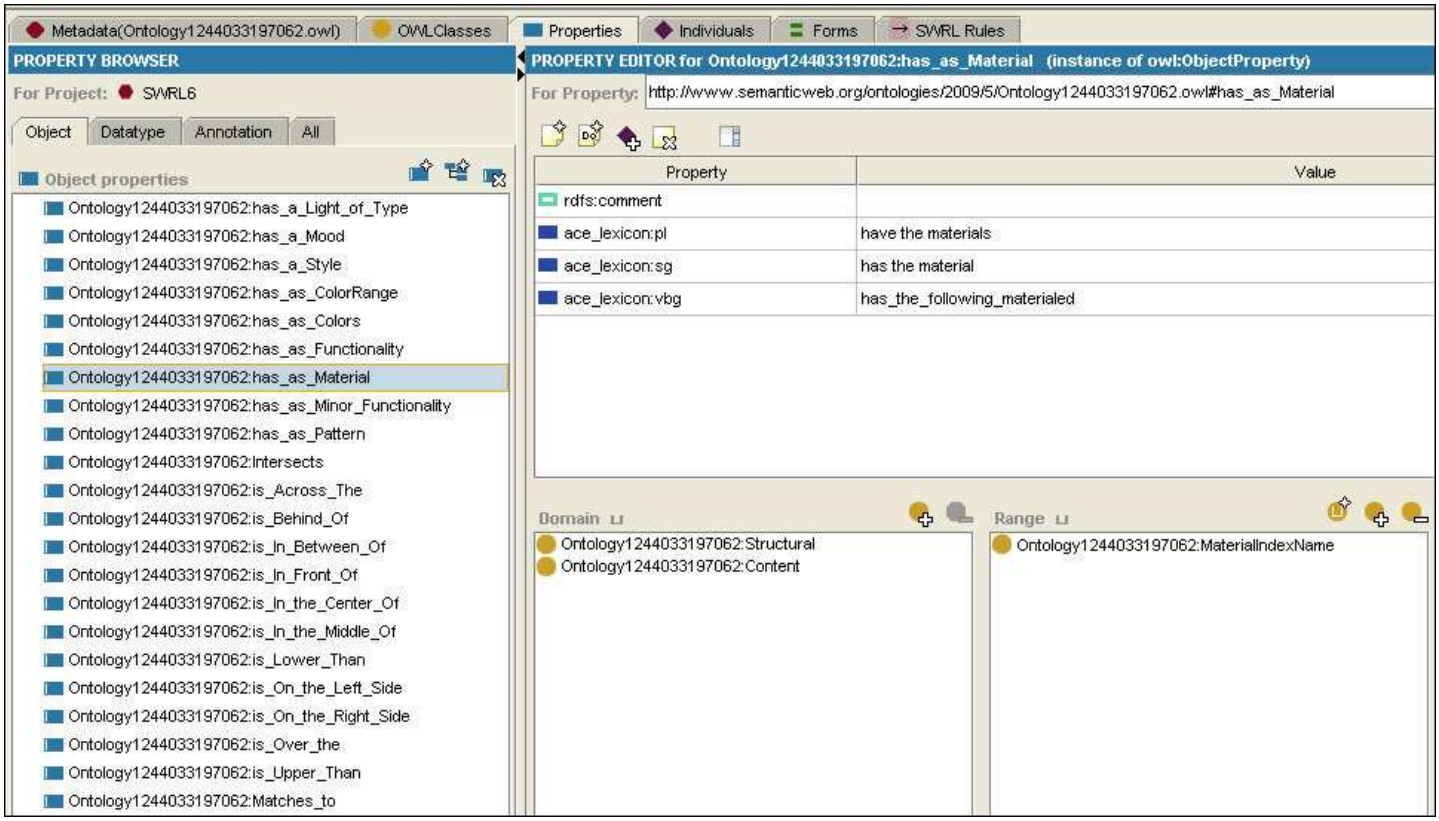
4.2.2 Ιδιότητες οντολογίας

Η πλήρης περιγραφή των φυσικών αντικειμένων μέσα στον χώρο και μεταξύ αυτού, είναι δυνατή μόνο με την αξιοποίηση των ιδιοτήτων. Οι ιδιότητες αποτελούν ένα από τα βασικά στοιχεία της OWL και χωρίζονται σε δύο μεγάλες κατηγορίες, τις object properties και τις datatype properties. Οι object properties μιας OWL οντολογίας συσχετίζουν τα άτομα των κλάσεων που δηλώνονται στο Domain πεδίο τους, με τα άτομα των κλάσεων που δηλώνονται στο Range πεδίο τους. Αντίθετα, οι datatype properties συσχετίζουν τα άτομα των κλάσεων που δηλώνονται στο Domain πεδίο τους, με ένα datatype που δηλώνεται στο Range πεδίο τους. Τα διαθέσιμα datatypes που επιτρέπεται να δηλωθούν είναι οποιουδήποτε τύπου από τα boolean, float, int, string, date, dateTime και time. Η απλή λογική που διέπει την κατασκευή των OWL προτάσεων στηρίζεται στα στοιχεία της γλωσσικής σύνταξης κατά τα οποία, το Domain πεδίο αντιστοιχεί στο υποκείμενο, η ιδιότητα αντιστοιχεί στο ρήμα και το Range πεδίο αντιστοιχεί στο αντικείμενο. Οι σχέσεις που λαμβάνουν μέρος μέσω των ιδιοτήτων, μαζί με τα περιεχόμενα άτομα των Domain και Range πεδίων τους, χρησιμοποιούνται και ως axioms για τον συλλογισμό. Ακολουθεί μια περιγραφική λίστα των ιδιοτήτων που υπάρχουν στην οντολογία, χωρισμένες ανάλογα με την κατηγορία στις οποίες ανήκουν:

- `Object properties`, όλες οι ιδιότητες αυτού του τύπου που έχουν υλοποιηθεί μέσα στην OWL οντολογία είναι οι εξής:
  - **has_a_Light_of_Type**, πρόκειται για την ιδιότητα που καθορίζει ότι μια οντότητα έχει κάποια πηγή φωτός, η οποία είναι άτομο της κλάσης LightSourcePalleteIndex

- **has_a_Mood**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια προσωπικότητα, το οποίο είναι άτομο της κλάσης MoodCharacterization
- **has_a_Style**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει κάποιο στυλ, το οποίο είναι άτομο της κλάσης StyleIndex
- **has_as_ColorRange**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorMood, περιέχει κάποια γενικευμένη κατηγορία, η οποία είναι άτομο της κλάσης ColorRange
- **has_as_Colors**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorRange ή Structural ή Content, έχει κάποιο χρώμα, το οποίο είναι άτομο της κλάσης ColorPaletteIndexName
- **has_as_Functionality**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια βασική λειτουργικότητα, η οποία είναι άτομο της κλάσης FunctionalityIndex
- **has_as_Material**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, προέρχεται από κάποιο υλικό, το οποίο είναι άτομο της κλάσης MaterialIndexName
- **has_as_Minor_Functionality**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει κάποια δευτερεύουσα λειτουργικότητα, η οποία είναι άτομο της κλάσης FunctionalityIndex
- **has_as_Pattern**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει κάποιο σχέδιο, το οποίο είναι άτομο της κλάσης PatternIndexName
- **Intersects**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, διασταυρώνεται με κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_Across_The**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται απέναντι από κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_Behind_Of**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται πίσω από κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_In_Between_Of**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται ανάμεσα σε κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_In_Front_Of**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται μπροστά από κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_In_the_Center_Of**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο κέντρο κάποιου άλλου ατόμου της κλάσης Structural ή Content
- **is_In_the_Middle_Of**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο μέσο σημείο κάποιου άλλου ατόμου της κλάσης Structural ή Content
- **is_Lower_Than**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται χαμηλότερα σε σχέση με κάποιο άλλο άτομο της κλάσης Structural ή Content
- **is_On_the_Left_Side**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο αριστερό μέρος κάποιου άλλου ατόμου της κλάσης Structural ή Content
- **is_On_the_Right_Side**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται στο δεξί μέρος κάποιου άλλου ατόμου της κλάσης Structural ή Content
- **is_Over_the**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται πάνω από κάποιο άλλο άτομο της κλάσης Structural ή Content

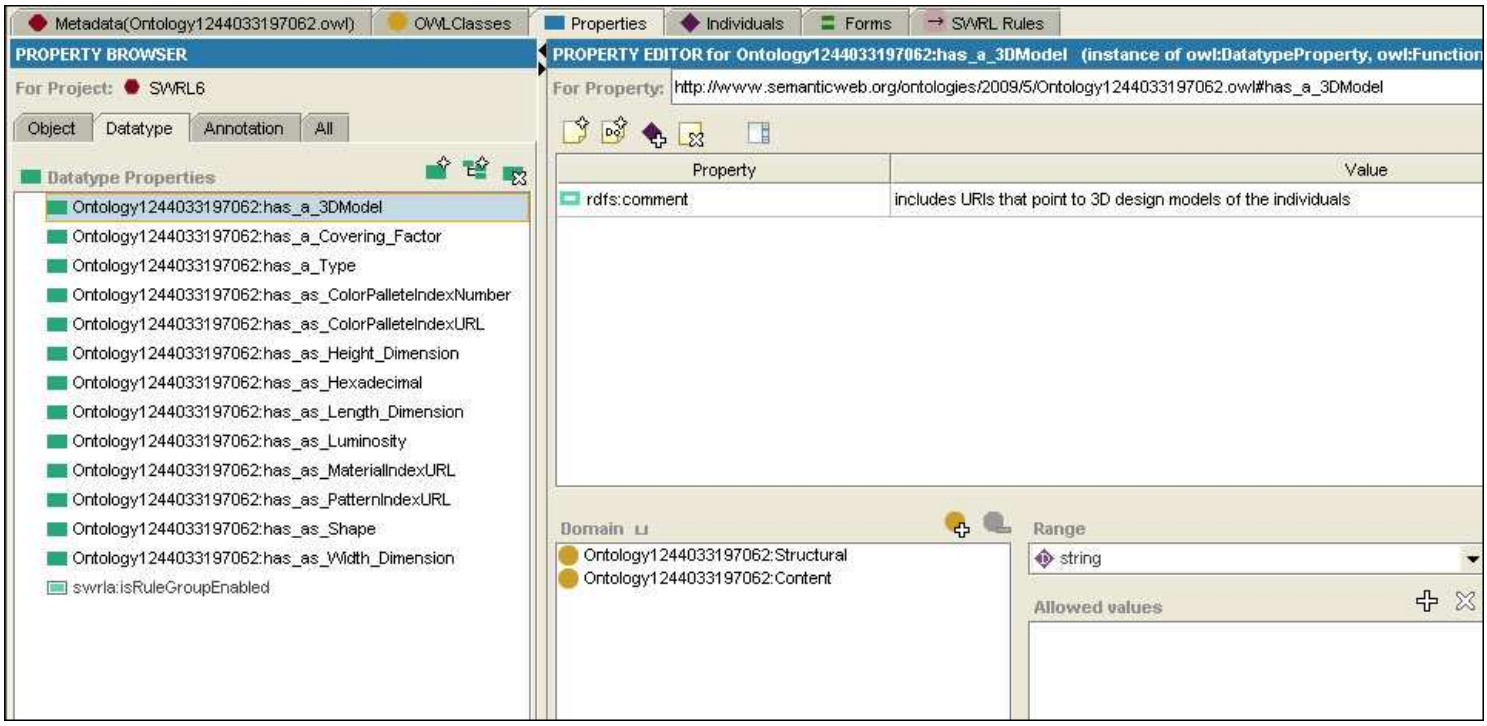
- **is_Upper_Than**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, βρίσκεται υψηλότερα σε σχέση με κάποιο άλλο άτομο της κλάσης Structural ή Content
- **Matches_to**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, ταιριάζει στο χώρο απόλυτα με κάποιο άλλο άτομο της κλάσης Structural ή Content



Εικόνα 9. Οι object properties της OWL οντολογίας

- `Datatype properties`, όλες οι ιδιότητες αυτού του τύπου που έχουν υλοποιηθεί μέσα στην OWL οντολογία είναι οι εξής:
  - **has_a_3D_Model**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα αλφαριθμητικό (string), το οποίο είναι ένα URI που δείχνει ένα τρισδιάστατο μοντέλο αυτού του ατόμου
  - **has_a_Covering_Factor**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει την εκατοστιαία αναλογία του δωματίου που καλύπτεται από αντικείμενα
  - **has_a_Type,** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα αλφαριθμητικό (string), το οποίο προσδίδει επιπλέον πληροφορίες για ένα αντικείμενο
  - **has_as_ColorPalleteIndexNumber,** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίο καθορίζει έναν μοναδικό αριθμό που αντιπροσωπεύει το χρώμα
  - **has_as_ColorPalleteIndexURL**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του χρώματος
  - **has_as_Height_Dimension**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του ύψους του αντικειμένου

- **has_as_Hexadecimal**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης ColorPaletteIndexName, έχει ένα αλφαριθμητικό (string), το οποίο καθορίζει τον δεκαεξαδικό αριθμό που αντιστοιχεί στο συγκεκριμένο χρώμα
- **has_as_ Length_Dimension,** πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του μήκους του αντικειμένου
- **has_as_Luminosity**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Room, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιπροσωπεύει την φωτεινότητα του εσωτερικού χώρου
- **has_as_MaterialIndexURL**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης MaterialIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του υλικού
- **has_as_PatternIndexURL**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης PatternIndexName, έχει ένα αλφαριθμητικό (string), το οποίο αναφέρει το URL ή απλώς περιέχει πληροφορία για την διαθεσιμότητα του σχεδίου
- **has_as_Shape**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα αλφαριθμητικό (string), το οποίο περιγράφει τα σχηματικά χαρακτηριστικά ενός φυσικού αντικειμένου
- **has_as_Width_Dimension**, πρόκειται για την ιδιότητα που καθορίζει ότι ένα άτομο της κλάσης Structural ή Content, έχει ένα πραγματικό αριθμό (float), ο οποίος αντιστοιχεί στην διάσταση του πλάτους του αντικειμένου



Εικόνα 10. Οι datatype properties της OWL οντολογίας

Η OWL, εκτός από τα κλασσικά χαρακτηριστικά γνωρίσματα ιδιοτήτων που διαθέτει (functional, symmetric, transitive, κτλ) μπορεί να αποδώσει περιορισμούς στις κλάσεις, πράγμα που οδηγεί σε περιορισμό του εύρους των αποτελεσμάτων που μπορεί να λάβει μια ιδιότητα. Ωστόσο, κάτι τέτοιο περιορίζει σημαντικά τον συνδυασμό των εσωτερικών χώρων και των αντικειμένων, με αποτέλεσμα να μειώνεται δραματικά η πολυμορφικότητα ενός εσωτερικού χώρου. Για αυτούς τους λόγους δεν έχει προστεθεί κανένας περιορισμός στις κλάσεις της OWL οντολογίας.

4.3 Εξαγωγή Οντολογίας

Το Protégé-OWL επιτρέπει την εξαγωγή μιας οντολογίας σε διάφορα πρότυπα, ανάμεσα στα οποία είναι και ο κώδικας Java. Το αποτέλεσμα αυτής της διαδικασίας εξαγωγής, παρέχει σε διάφορα προγράμματα ανάπτυξης εφαρμογών (όπως το NetBeans) την δημιουργία λογισμικού βασισμένου στις ανάγκες μιας OWL οντολογίας, μέσω της εξέτασης των κλάσεων και των συσχετίσεων που υπάρχουν. Επίσης γίνεται δυνατή και η δημιουργία ενός UML (Unified Modeling Language) Diagram. Για την εξαγωγή της OWL οντολογίας σε κώδικα Java, αρκεί να ακολουθήσουμε την διαδρομή Code → Generate Protege-OWL Java Code. Μετά αναγράφουμε την διαδρομή αποθήκευσης, το όνομα του Java package και το όνομα της κλάσης Factory.

Για την δημιουργία ενός UML Diagram μέσα από το NetBeans IDE 6.9.1, πρέπει να δημιουργήσουμε ένα καινούργιο κενό project και να προσθέσουμε σε ένα package, που θα έχει το ίδιο όνομα με αυτό που επιλέξαμε στο πεδίο εξαγωγής, όλες τις κλάσεις που δημιουργήθηκαν από το Protégé-OWL code. Όποια μηνύματα λάθους προκύψουν στις κλάσεις του πακέτου, οφείλονται στο ότι δεν περιλαμβάνονται οι βιβλιοθήκες που είναι απαραίτητες για την ορθή εκτέλεση του project. Για την επίλυση αυτού του προβλήματος θα πρέπει να εισάγουμε όλα τα .jar αρχεία του φακέλου edu.stanford.smi.protegex.owl, που βρίσκεται στον φάκελο εγκατάστασης του Protégé-OWL, στις βιβλιοθήκες του project που δημιουργήσαμε. Έπειτα διαλέγοντας την επιλογή “Reverse Engineer…” δημιουργούμε ένα Java-Model project. Στο καινούργιο project ακολουθώντας την δενδρική διαδρομή Model → Ontology, επιλέγουμε τα στοιχεία που θέλουμε να συμπεριληφθούν στο διάγραμμά μας, και με δεξί κλικ διαλέγουμε την επιλογή “Create Diagram From Selected Elements…”. Το μόνο που απομένει είναι η επιλογή του τύπου του UML διαγράμματος που επιθυμούμε να εξάγουμε.

4.4 Γράφος Οντολογίας

Το OWLViz είναι ένα plugin που σχεδιάστηκε για αλληλεπίδραση με τον Protégé-OWL editor. Κύριο χαρακτηριστικό του, είναι η υποστήριξη αυξητικής απεικόνισης και περιήγησης της ιεραρχίας των κλάσεων μιας OWL οντολογίας. Με αυτόν τον τρόπο επιτρέπει την σύγκριση της δυναμικής ιεραρχίας κλάσεων και της συμπερασματικής ιεραρχίας κλάσεων. Καθώς το OWLViz εντάσσεται με τον Protégé-OWL editor, χρησιμοποιεί τα τυπικά είδη χρώματος για τον διαχωρισμό των πρώτιστων κλάσεων και των καθορισμένων από τον χρήστη κλάσεων. Οποιαδήποτε αλλαγή γίνεται άμεσα εμφανής, ενώ πιθανές ασταθείς αντιλήψεις τονίζονται με έντονο κόκκινο χρώμα. Επίσης, το OWLViz παρέχει την δυνατότητα αποθήκευσης είτε της asserted ή της inferred απεικόνισης της ιεραρχίας των κλάσεων της οντολογίας, σε διάφορα πρότυπα ανάμεσα στα οποία περιλαμβάνονται τα PNG, JPEG, SVG και DOT.

Όπως προαναφέρθηκε, το OWLViz περιέχεται στην διανομή του Protégé-OWL, αλλά αυτό δεν είναι αρκετό για την επιτυχής ενεργοποίηση του. Το πρώτο βήμα που θα πρέπει να εκτελεστεί είναι η εγκατάσταση της πιο πρόσφατης έκδοσης του Graphviz (στην περίπτωσή μας χρησιμοποιήθηκε η έκδοση Graphviz 2.26.3), το οποίο αποτελεί ένα ελεύθερο λογισμικό για την απεικόνιση γράφων και διατίθεται από την AT&T Research μέσω του ιστότοπου http://www.graphviz.org/Download..php. Εφόσον έχουμε ήδη τελειώσει με την εγκατάσταση του Graphviz, προχωρούμε στην εκκίνηση του Protégé-OWL, το οποίο θα προσπαθήσει να εντοπίσει την διαδρομή εγκατάστασής του Graphviz. Η διαδικασία καθορισμού του μονοπατιού εγκατάστασης γίνεται βάση του λειτουργικού συστήματος που διαθέτει ο χρήστης, και τις περισσότερες φορές αποτυγχάνει. Η ενεργοποίηση του OWLViz γίνεται μέσω του μενού Project → Configure → Tab Widgets → OWLVizTab. Εάν έχει αποτύχει να εντοπίσει επιτυχώς το μονοπάτι εγκατάστασης θα εμφανιστεί ένα μήνυμα λάθους (DOT Error). Τότε, επιλέγοντας το εικονίδιο “Options” της μπάρας εικονιδίων του OWLViz, και ακολούθως την καρτέλα “Layout Options”, τοποθετούμε την σωστή διαδρομή εγκατάστασης του Graphviz 2.26.3\bin\dot.exe στην περιοχή κειμένου της “Dot Application Path” καρτέλας.

Εικόνα 11. Γράφος με την ιεραρχία των κλάσεων της οντολογίας

5. CASE STUDIES

Στο παρών κεφάλαιο περιγράφονται λεπτομερώς όλοι οι SWRL κανόνες και τα SQWRL ερωτήματα που έχουν υλοποιηθεί στην OWL οντολογία. Η περιγραφή περιλαμβάνει την παρουσίαση του κανόνα όπως συντάσσεται στο SWRLTab, την ανάλυση των στοιχείων από τα οποία αποτελείται, και τέλος, τα αποτελέσματα που επιστρέφει. Για να είναι εφικτή η διάδραση των SWRL κανόνων και των SQWRL ερωτημάτων με τη βάση γνώσης, ήταν απαραίτητο η οντολογία μας να εμπλουτιστεί με άτομα κλάσεων και να συσχετίσει οντότητες μέσω της απόδοσης τιμών στα πεδία domain και range των ιδιοτήτων.

Τα SQWRL ερωτήματα, βασισμένα στην υπάρχουσα γνώση βάσης που περιέχεται στην OWL οντολογία, θα επιστρέψουν κάποιο αποτέλεσμα ή την ένδειξη ότι δεν δημιουργήθηκε κάποιο αποτέλεσμα αν δεν ικανοποιούνται οι συνθήκες του antecedent μέρους του κανόνα. Τα αποτελέσματα που παράγονται από ένα SQWRL ερώτημα δεν μπορούν να χρησιμοποιηθούν από τον χρήστη, άρα ούτε και να εισαχθούν ως νέα γνώση στην OWL οντολογία. Κάτι τέτοιο θα παραβίαζε την open world assumption και θα οδηγούσε σε non-monotonicity, μέθοδοι λογικής που δεν υποστηρίζονται από την OWL.

Σε αντίθεση με τα SQWRL ερωτήματα, οι SWRL κανόνες προσθέτουν την γνώση που προκύπτει από το consequent μέρος του κανόνα (εφόσον ικανοποιούνται πλήρως οι συνθήκες του antecedent μέρους), πίσω στην οντολογία. Ωστόσο, εξακολουθούν να δεσμεύονται από τα SWRL semantics, με αποτέλεσμα να μην υποστηρίζουν non-monotonicity, μορφές άρνησης ως διάψευση (negation as failure) και disjunction (λογικός τελεστής που επιστρέφει true εάν ένας από τους τελεστές του είναι αληθής). Επιπλέον, οι SWRL κανόνες είναι αδύνατο να ανακαλέσουν ή να αφαιρέσουν γνώση που υπάρχει στην βάση ή συμπεράθηκε από κάποιους SWRL κανόνες. Αυτό συνεπάγεται πως στην περίπτωση που πρέπει να εισάγουν τιμή σε κάποιο πεδίο που φέρει ήδη μία ή περισσότερες τιμές, η νέα τιμή θα προστεθεί ως επιπλέον πληροφορία. Εάν πρόκειται για functional ιδιότητα, ένας OWL reasoner (π.χ. Pellet 1.5.2) θα εντοπίσει την αντίφαση όταν πρόκειται να προστεθεί η τιμή πίσω στην ιδιότητα. Τέλος, για την αποσφαλμάτωση των SWRL κανόνων μπορούμε να χρησιμοποιήσουμε τα SQWRL ερωτήματα, κρατώντας σταθερό το antecedent μέρος του κανόνα και μεταβάλλοντας το consequent μέρος στο αντίστοιχο sqwrl:select build-in.

5.1 SQWRL Ερωτήματα

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/AddingStringToContentHavingModel3D

SWRL Expression:

```
Ontology1244033197062:Content(?c) ^
Ontology1244033197062:has_a_3DModel(?c, ?m) ^
swrlb:stringConcat(?result, "This is a URI:  ", ?m) →
sqwrl:select(?result)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Content και των υποκλάσεων της, που διαθέτουν μια τιμή στην ιδιότητα "has_a_3DModel" και μέσω του core build-in stringConcat, προσθέτει το αλφαριθμητικό "This is a URI:" στην αρχή της τιμής αυτής. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει το νέο αλφαριθμητικό (result) που δημιουργήθηκε.

Αποτέλεσμα:

This is a URI: http://www.loewen.com/windows/windowStyles/awning/3DModel/Window46
This is a URI: http://www.loewen.com/windows/windowStyles/awning/3DModel/Window24
This is a URI: http://www.greeklandscapes.com/greece/athens_museum_classica

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/AllContentWithMaterialTypeWood

SWRL Expression:

```
Ontology1244033197062:Content(?c) ^
Ontology1244033197062:has_as_Material(?c, ?m) ˚
sqwrl:makeSet(?set, ?m) ^ sqwrl:groupBy(?set, ?c) ^
sqwrl:makeSet(?test, Ontology1244033197062:Wood) ˚
sqwrl:intersects(?set, ?test) →
sqwrl:select(?c)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Content και των υποκλάσεων της, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Material". Έπειτα, δημιουργείται ένα σύνολο που περιέχει όλα τα υλικά των ατόμων της κλάσης Content και μέσω του χειριστή groupBy, κάθε υλικό παίρνει το αντίστοιχο άτομο της κλάσης Content που το προσδιορίζει. Μετά, ένα δεύτερο σύνολο περιέχει αποκλειστικά και μόνο το υλικό ξύλο (Ontology1244033197062:Wood), και χρησιμοποιώντας τον χειριστή intersects παίρνουμε την τομή των δύο συνόλων. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τα άτομα της κλάσης Content που ικανοποιούν τις προηγούμενες συνθήκες.

Αποτέλεσμα:

Ontology1244033197062:Construction_4
Ontology1244033197062:Desk_17
Ontology1244033197062:Door_3
Ontology1244033197062:Table_13
Ontology1244033197062:Window_2
Ontology1244033197062:Window_3
Ontology1244033197062:Window_4

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/AllIndividualsAndTheirValuesHaving_has_a_Mood_Property

SWRL Expression:

```
abox:isIndividual(?i) ^ abox:hasValue(?i,
 Ontology1244033197062:has_a_Mood, ?v) →
sqwrl:select(?i, Ontology1244033197062:has_a_Mood, ?v)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της οντολογίας που διαθέτουν μια τιμή στην ιδιότητα “has_as_Mood”. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τρεις στήλες που αντίστοιχα περιέχουν: τα άτομα, την ιδιότητα που συσχετίζει τα άτομα με μια προσωπικότητα, και την προσωπικότητα που αντιστοιχεί στο άτομο αυτό.

Αποτέλεσμα:

Ontology1244033197062:Room_1	Ontology1244033197062:has_a_Mood	Ontology1244033197062:Relaxing

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/AllIndividualsWithValueFor_has_as_Colors

SWRL Expression:

```
abox:isIndividual(?i) ^
abox:hasProperty(?i, Ontology1244033197062:has_as_Colors) →
sqwrl:select(?i)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της οντολογίας που διαθέτουν μια τιμή στην ιδιότητα “has_as_Colors”. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τα άτομα που ικανοποιούν την συνθήκη του antecedent μέρους.

Αποτέλεσμα:

Ontology1244033197062:Beige_Colors
Ontology1244033197062:Blue_Colors
Ontology1244033197062:Light_Brown_Colors

Ontology1244033197062:Red_Colors
Ontology1244033197062:Turquoise_Colors
Ontology1244033197062:Dark_Brown_Colors
Ontology1244033197062:Black_Colors
Ontology1244033197062:Terracotta_Colors
Ontology1244033197062:Stone_Colors
Ontology1244033197062:Earth_Tones_Colors
Ontology1244033197062:Pink_Colors
Ontology1244033197062:Green_Colors
Ontology1244033197062:Purple_Colors
Ontology1244033197062:Yellow_Colors
Ontology1244033197062:White_Colors
Ontology1244033197062:Cream_Colors
Ontology1244033197062:Ochre_Colors
Ontology1244033197062:Gray_Colors
Ontology1244033197062:Brown_Colors
Ontology1244033197062:Orange_Colors
Ontology1244033197062:Indigo_Colors
Ontology1244033197062:Mid_Brown_Colors

****

SWRL Editor Name:
`http://www.semanticweb.org/ontologies/2009/5/AllOWLClasses`

SWRL Expression:

```
tbox:isOWLClass(?c) →
sqwrl:select(?c) ^ sqwrl:orderBy(?c)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος απλώς θα ανακτήσει όλες τις κλάσεις της οντολογίας. Στο consequent μέρος το build-in sqwrl:select θα επιστρέψει αυτές τις κλάσεις και με τη βοήθεια του χειριστή sqwrl:orderBy θα τις ταξινομήσει με αύξουσα σειρά.

Αποτέλεσμα:

swrla:Entity
swrla:RuleGroup
Ontology1244033197062:Accessory
Ontology1244033197062:ArmChair
Ontology1244033197062:Bed
Ontology1244033197062:BedSide_Table
Ontology1244033197062:Bookcase
Ontology1244033197062:Carpets
Ontology1244033197062:Ceiling
Ontology1244033197062:Chairs
Ontology1244033197062:Closet
Ontology1244033197062:ColorMood

Ontology1244033197062:ColorPaletteIndexName
Ontology1244033197062:ColorRange
Ontology1244033197062:Construction
Ontology1244033197062:Content
Ontology1244033197062:Curtains
Ontology1244033197062:Cushions
Ontology1244033197062:DataTypesCandidates
Ontology1244033197062:Desk
Ontology1244033197062:Door
Ontology1244033197062:Floor
Ontology1244033197062:FunctionalityIndex
Ontology1244033197062:Furniture
Ontology1244033197062:LightSourcePalleteIndex
Ontology1244033197062:Material
Ontology1244033197062:MaterialIndexName
Ontology1244033197062:Mirrors
Ontology1244033197062:MoodCharacterization
Ontology1244033197062:Paintings
Ontology1244033197062:PatternIndexName
Ontology1244033197062:Room
Ontology1244033197062:Slipcovers
Ontology1244033197062:Sofa
Ontology1244033197062:Structural
Ontology1244033197062:StyleIndex
Ontology1244033197062:Table
Ontology1244033197062:Throw
Ontology1244033197062:Upholstery
Ontology1244033197062:Wall
Ontology1244033197062:Window

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/AlphabeticallyFirstThreeRoomWithStyleIndex

SWRL Expression:

```
Ontology1244033197062:Room(?r) ^
Ontology1244033197062:has_a_Style(?r, ?s) →
sqwrl:select(?r) ^ sqwrl:orderBy(?r) ^ sqwrl:firstN(3)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Room που διαθέτουν μια τιμή στην ιδιότητα “has_a_Style”. Στο consequent μέρος το build-in sqwrl:select θα επιστρέψει αυτά τα άτομα και με τη βοήθεια του χειριστή sqwrl:orderBy θα τα ταξινομήσει με αύξουσα σειρά. Χρησιμοποιώντας τον χειριστή sqwrl:firstN επιβάλλαμε την εμφάνιση μόνο των τριών πρώτων ταξινομημένων αποτελεσμάτων (παραβλέποντας το γεγονός ότι κάθε οντολογία περιγράφει μόνο ένα δωμάτιο κάθε συγκεκριμένη στιγμή).

Αποτέλεσμα:

Ontology1244033197062:Room_1
Ontology1244033197062:Room_2
Ontology1244033197062:Room_3

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ColorPaletteIndexCounter

SWRL Expression:

```
Ontology1244033197062:ColorPaletteIndexName(?c) → sqwrl:count(?c)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος απλώς θα ανακτήσει όλα τα άτομα της κλάσης ColorPaletteIndexName. Στο consequent μέρος το build-in sqwrl:count θα καταμετρήσει και θα επιστρέψει τον αριθμό των ατόμων αυτών (στην πραγματικότητα πόσα χρώματα υπάρχουν στην οντολογία).

Αποτέλεσμα:

181

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/UnboundMathExample

SWRL Expression:

```
swrlb:multiply(?x, 2, 4) → sqwrl:select(?x)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα πολλαπλασιάσει το δεύτερο με το τρίτο argument και εφόσον το πρώτο argument είναι unbound, θα του αποδώσει το αποτέλεσμα του γινομένου. Στο consequent μέρος το build-in sqwrl:select θα επιστρέψει τον αριθμό αυτό. Αν το πρώτο argument ήταν bound, δηλαδή έδειχνε σε κάποια μεταβλητή ή ήταν κάποιος συγκεκριμένος αριθμός, τότε θα προσδιόριζε αν το αποτέλεσμα του γινομένου είναι ίσο (true) ή όχι (false) με αυτό το argument.

Αποτέλεσμα:

8

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ConstructionWithMaterialTypeAndShapePallete

SWRL Expression:

```
Ontology1244033197062:Construction(?c) ^
Ontology1244033197062:has_as_Material(?c, ?m) ˚
sqwrl:makeSet(?set, ?m) ^
Ontology1244033197062:has_as_Shape(?c, ?s) ^
sqwrl:makeSet(?set, ?s) →
sqwrl:select(?c)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Construction, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Material" και μια τιμή στην ιδιότητα "has_as_Shape". Παράλληλα, δημιουργούνται δύο σύνολα που περιέχουν αντίστοιχα, τα υλικά και τα σχήματα που αντιστοιχούν στα άτομα της κλάσης Construction. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τα άτομα που ικανοποιούν τις δύο συνθήκες που καθορίζονται στο antecedent μέρους.

Αποτέλεσμα:

Ontology1244033197062:Construction_2
Ontology1244033197062:Construction_4

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ContentWithColorPaletteIndex

SWRL Expression:

```
Ontology1244033197062:Content(?c) ^
Ontology1244033197062:ColorPaletteIndexName(?p) ^
Ontology1244033197062:has_as_Colors(?c, ?p) →
sqwrl:select(?c, ?p)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Content, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Colors", καθώς και τα άτομα της κλάσης ColorPaletteIndexName. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει δύο στήλες που αντίστοιχα περιέχουν: τα άτομα της κλάσης Content που ικανοποιούν την συνθήκη, και το χρώμα που αντιστοιχεί στα άτομα αυτά.

Αποτέλεσμα:

Ontology1244033197062:Cushions_4	Ontology1244033197062:Beige_Soft
Ontology1244033197062:Construction_3	Ontology1244033197062:Beige_Fresh
Ontology1244033197062:Window_2	Ontology1244033197062:Black
Ontology1244033197062:Curtains_5	Ontology1244033197062:Taupe
Ontology1244033197062:Chairs_20	Ontology1244033197062:Brown
Ontology1244033197062:Desk_17	Ontology1244033197062:Brown

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ContentWithMoreThan2MaterialType

SWRL Expression:

```
Ontology1244033197062:Content(?c) ^
Ontology1244033197062:has_as_Material(?c, ?m) ˚
sqwrl:makeSet(?set, ?m) ^ sqwrl:groupBy(?set, ?c) ˚
sqwrl:size(?size, ?set) ^ swrlb:greaterThan(?size, 1) →
sqwrl:select(?c, ?m)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Content, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Material". Έπειτα, δημιουργείται ένα σύνολο που περιέχει τα υλικά των ατόμων της κλάσης Content και μέσω του χειριστή groupBy, κάθε υλικό παίρνει το αντίστοιχο άτομο της κλάσης Content που το προσδιορίζει. Στη συνέχεια χρησιμοποιείται ο χειριστής sqwrl:size για την εύρεση του μεγέθους κάθε στοιχείου του συνόλου, ενώ ο swrlb:greaterThan αναδεικνύει μόνο τα στοιχεία που έχουν μέγεθος μεγαλύτερο του δύο. Αυτό μεταφράζεται στην επιλογή των ατόμων που αποτελούνται από δύο υλικά. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει δύο στήλες που αντίστοιχα περιέχουν: τα άτομα της κλάσης Content που ικανοποιούν τις συνθήκες του antecedent μέρους, και τα υλικό που αντιστοιχεί στα άτομα αυτά.

Αποτέλεσμα:

Ontology1244033197062:Door_1	Ontology1244033197062:Metal
Ontology1244033197062:Door_1	Ontology1244033197062:Steel
Ontology1244033197062:Window_1	Ontology1244033197062:Glass
Ontology1244033197062:Window_1	Ontology1244033197062:Metal
Ontology1244033197062:Window_2	Ontology1244033197062:Glass
Ontology1244033197062:Window_2	Ontology1244033197062:Wood
Ontology1244033197062:Window_3	Ontology1244033197062:Glass
Ontology1244033197062:Window_3	Ontology1244033197062:Wood
Ontology1244033197062:Window_4	Ontology1244033197062:Glass
Ontology1244033197062:Window_4	Ontology1244033197062:Wood

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/RoomsWithNonOfficeFunctionality

SWRL Expression:

```
Ontology1244033197062:Room(?r) ^
Ontology1244033197062:has_as_Functionality(?r, ?m) ˚
sqwrl:makeSet(?set, ?m) ^ sqwrl:groupBy(?set, ?r) ˚
sqwrl:notElement(Ontology1244033197062:Office, ?set) →
sqwrl:select(?r, ?m)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Room, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Functionality". Έπειτα, δημιουργείται ένα σύνολο που περιέχει όλα τα είδη λειτουργικότητας των ατόμων της κλάσης Room και μέσω του χειριστή groupBy, κάθε λειτουργικότητα παίρνει το αντίστοιχο άτομο της κλάσης Room που το προσδιορίζει. Μετά, ο χειριστής sqwrl:notElement απορρίπτει αποκλειστικά και μόνο την λειτουργικότητα του γραφείου (Ontology1244033197062:Office) από το σύνολο. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει δύο στήλες που αντίστοιχα περιέχουν: τα άτομα της κλάσης Room που ικανοποιούν τις συνθήκες του antecedent μέρους, και την λειτουργικότητα του δωματίου που αντιστοιχεί στα άτομα αυτά.

Αποτέλεσμα:

Ontology1244033197062:Room_1	Ontology1244033197062:Family_Room
Ontology1244033197062:Room_5	Ontology1244033197062:Playroom

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/DifferenceExample

SWRL Expression:

```
    ˚ sqwrl:makeSet(?s1, Ontology1244033197062:Bed_10) ^
     sqwrl:makeSet(?s2, Ontology1244033197062:Sofa_14) ˚
  sqwrl:difference(?d, ?s1, ?s2) ^ sqwrl:size(?size, ?d) →
                    sqwrl:select(?size)
```

Επεξήγηση:
Στο antecedent μέρος του ερωτήματος δημιουργείται ένα σύνολο μέσα στο οποίο τοποθετείται ένα συγκεκριμένο άτομο (Ontology1244033197062:Bed_10), που αναπαριστά ένα κοινό κρεβάτι. Έπειτα, δημιουργείται ένα δεύτερο σύνολο μέσα στο οποίο τοποθετείται ένα συγκεκριμένο άτομο (Ontology1244033197062:Sofa_14), που αναπαριστά ένα κοινό καναπέ. Μετά, εφαρμόζουμε τον χειριστή sqwrl:difference για να διακρίνουμε αν πρόκειται για διαφορετικά άτομα. Το αποτέλεσμά του είναι προσπελάσιμο μέσω του χειριστή sqwrl:size, και θα έχει τιμή μηδέν αν πρόκειται για ίδια άτομα (false) και τιμή 1 αν πρόκειται για διαφορετικά άτομα (true). Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει το αποτέλεσμα του χειριστή sqwrl:size.

Αποτέλεσμα:

1

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/DirectSubClassesOfDataType_CandidatesClass

SWRL Expression:

```
        tbox:isDirectSubClassOf(?subClass,
    Ontology1244033197062:DataTypesCandidates) →
            sqwrl:select(?subClass)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει μόνο τις απευθείας υποκλάσεις της κλάσης DataTypesCandidates, αγνοώντας τυχόν υποκλάσεις σε βάθος μεγαλύτερο του 1. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τις κλάσεις αυτές.

Αποτέλεσμα:

Ontology1244033197062:FunctionalityIndex
Ontology1244033197062:ColorMood
Ontology1244033197062:StyleIndex
Ontology1244033197062:Material
Ontology1244033197062:ColorPaletteIndexName
Ontology1244033197062:ColorRange
Ontology1244033197062:LightSourcePalleteIndex
Ontology1244033197062:MoodCharacterization

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/DoorWithMaterialTypeInDescendingOrder

SWRL Expression:

```
Ontology1244033197062:Door(?d) ^
Ontology1244033197062:has_as_Material(?d, ?m) →
sqwrl:select(?d, ?m) ^ sqwrl:orderByDescending(?m)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Door, που διαθέτουν μια τιμή στην ιδιότητα "has_as_Material". Στο consequent μέρος υπάρχει το build-in sqwrl:select, το οποίο επιστρέφει δύο στήλες που αντίστοιχα περιέχουν: τα άτομα της κλάσης Door που ικανοποιούν την συνθήκη του antecedent μέρους, και το υλικό που αντιστοιχεί στα άτομα αυτά. Χρησιμοποιώντας τον χειριστή sqwrl:orderByDescending, η ταξινόμηση των αποτελεσμάτων γίνεται κατά φθίνουσα σειρά βάση της ονομασίας των υλικών.

Αποτέλεσμα:

Ontology1244033197062:Door_3	Ontology1244033197062:Wood
Ontology1244033197062:Door_1	Ontology1244033197062:Steel
Ontology1244033197062:Door_1	Ontology1244033197062:Metal
Ontology1244033197062:Door_2	Ontology1244033197062:Glass

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/Furniture

SWRL Expression:

```
Ontology1244033197062:Furniture(?f) →
sqwrl:select(?f)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Furniture. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τα άτομα αυτά.

Αποτέλεσμα:

Ontology1244033197062:Closet_12
Ontology1244033197062:Bed_10
Ontology1244033197062:Desk_17
Ontology1244033197062:Sofa_14
Ontology1244033197062:Desk_18
Ontology1244033197062:Chairs_20
Ontology1244033197062:ArmChair_27
Ontology1244033197062:BedSide_Table_21
Ontology1244033197062:Table_13
Ontology1244033197062:Bookcase_11

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/FunctionalPropertiesOfOntology

SWRL Expression:

```
tbox:isOWLProperty(?p) ^ tbox:isOWLFunctionalProperty(?p) →
                        sqwrl:select(?p)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει τις ιδιότητες της OWL οντολογίας που φέρουν το χαρακτηριστικό functional. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει τις ιδιότητες αυτές.

Αποτέλεσμα:

Ontology1244033197062:has_a_Covering_Factor
swrla:hasBuiltInPhrase
swrla:isRuleGroupEnabled
swrla:hasClassPhrase
swrla:isRuleEnabled
swrla:hasPropertyPhrase
Ontology1244033197062:has_as_Hexadecimal
Ontology1244033197062:has_as_Shape
Ontology1244033197062:has_a_3DModel

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ObjectPropertiesWithTheirRanges

SWRL Expression:

```
tbox:isOWLObjectProperty(?p) ^ tbox:isInRangeOf(?r, ?p) →
                        sqwrl:select(?p, ?r)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει εκείνες τις ιδιότητες της OWL οντολογίας που αποδίδουν σχέσεις ανάμεσα στα στιγμιότυπα των κλάσεων, δηλαδή όλες τις object properties. Επίσης, μέσω του build-in tbox:isInRangeOf προσδιορίζονται οι τιμές του πεδίου Range κάθε τέτοιας ιδιότητας. Στο consequent μέρος υπάρχει μόνο το build-in sqwrl:select, το οποίο επιστρέφει δύο στήλες που περιέχουν αντίστοιχα: τις ιδιότητες αυτές, και το αντίστοιχο Range τους.

Αποτέλεσμα:

Ontology1244033197062:is_Behind_Of	Ontology1244033197062:Content
Ontology1244033197062:is_Behind_Of	Ontology1244033197062:Structural
Ontology1244033197062:has_as_Minor_Functionality	Ontology1244033197062:FunctionalityIndex
Ontology1244033197062:Intersects	Ontology1244033197062:Content

Ontology1244033197062:Intersects	Ontology1244033197062:Structural
Ontology1244033197062:is_In_Between_Of	Ontology1244033197062:Structural
Ontology1244033197062:is_In_Between_Of	Ontology1244033197062:Content
Ontology1244033197062:has_as_Material	Ontology1244033197062:MaterialIndexName
Ontology1244033197062:is_On_the_Right_Side	Ontology1244033197062:Structural
Ontology1244033197062:is_On_the_Right_Side	Ontology1244033197062:Content
swrla:hasRuleGroup	swrla:RuleGroup
Ontology1244033197062:is_Over_the	Ontology1244033197062:Structural
Ontology1244033197062:is_Over_the	Ontology1244033197062:Content
Ontology1244033197062:is_On_the_Left_Side	Ontology1244033197062:Content
Ontology1244033197062:is_On_the_Left_Side	Ontology1244033197062:Structural
Ontology1244033197062:has_a_Style	Ontology1244033197062:StyleIndex
Ontology1244033197062:is_In_the_Center_Of	Ontology1244033197062:Content
Ontology1244033197062:is_In_the_Center_Of	Ontology1244033197062:Structural
Ontology1244033197062:is_Upper_Than	Ontology1244033197062:Structural
Ontology1244033197062:is_Upper_Than	Ontology1244033197062:Content
Ontology1244033197062:is_Lower_Than	Ontology1244033197062:Content
Ontology1244033197062:is_Lower_Than	Ontology1244033197062:Structural
Ontology1244033197062:is_In_the_Middle_Of	Ontology1244033197062:Content
Ontology1244033197062:is_In_the_Middle_Of	Ontology1244033197062:Structural
Ontology1244033197062:is_In_Front_Of	Ontology1244033197062:Content
Ontology1244033197062:is_In_Front_Of	Ontology1244033197062:Structural
Ontology1244033197062:has_as_Pattern	Ontology1244033197062:PatternIndexName
Ontology1244033197062:has_a_Light_of_Type	Ontology1244033197062:LightSourcePalleteIndex
Ontology1244033197062:is_Across_The	Ontology1244033197062:Structural
Ontology1244033197062:is_Across_The	Ontology1244033197062:Content
Ontology1244033197062:has_as_Colors	Ontology1244033197062:ColorPaletteIndexName
Ontology1244033197062:Matches_to	Ontology1244033197062:Content
Ontology1244033197062:Matches_to	Ontology1244033197062:Structural
Ontology1244033197062:has_as_ColorRange	Ontology1244033197062:ColorRange
Ontology1244033197062:has_a_Mood	Ontology1244033197062:MoodCharacterization
Ontology1244033197062:has_as_Functionality	Ontology1244033197062:FunctionalityIndex

****

SWRL Editor Name:
http://www.semanticweb.org/ontologies/2009/5/ThirdToSixthStructuralWithStylePallete

SWRL Expression:

```
Ontology1244033197062:Structural(?p) ^
Ontology1244033197062:has_a_Style(?p, ?name) →
sqwrl:select(?p) ^ sqwrl:orderBy(?p) ^ sqwrl:nthLastSlice(3, 6)
```

Επεξήγηση:
Το antecedent μέρος του ερωτήματος θα ανακτήσει όλα τα άτομα της κλάσης Structural, που διαθέτουν μια τιμή στην ιδιότητα "has_a_Style". Στο consequent μέρος υπάρχει το build-in sqwrl:select, το οποίο επιστρέφει τα άτομα της κλάσης Structural που ικανοποιούν την συνθήκη του antecedent μέρους, ταξινομημένα κατά αύξουσα σειρά μέσω του χειριστή sqwrl:select. Όμως, αξιοποιώντας τον χειριστή sqwrl:nthLastSlice αρχίζουμε απ' το άτομο της 4η θέσης και απεικονίζουμε μόνο τα επόμενα έξι στην αύξουσα σειρά.

Αποτέλεσμα:

Ontology1244033197062:Floor_12
Ontology1244033197062:Floor_3
Ontology1244033197062:Room_1
Ontology1244033197062:Room_5
Ontology1244033197062:Wall_1
Ontology1244033197062:Wall_7

5.2 SWRL Κανόνες

Η OWL οντολογία περιγράφει κάθε φορά ένα διαφορετικό δωμάτιο που περιέχει σχεδόν οτιδήποτε μπορεί να έχει και ένα τυπικό καθημερινό δωμάτιο. Σε αυτά τα αντικείμενα του χώρου μπορούν να αποδοθούν συγκεκριμένα υλικά κατασκευής, κάποιο σχήμα δόμησης, και χρώματα. Η πραγμάτωση αυτών των χαρακτηριστικών είναι δυνατή μέσω των ιδιοτήτων της OWL οντολογίας και οι τιμές τους καθορίζονται βάση του χρώματος που φέρει το πάτωμα αυτού του χώρου. Όμως, η αυτόματη απόδοση αυτών των τιμών είναι εφικτή μόνο μέσω της συγγραφής και της εκτέλεσης SWRL κανόνων, που οδηγούν στην επιθυμητή από τον χρήστη συμπερασματολογία.

SWRL Editor Name:
RULE1:_IF_WhiteFloors_THEN_CircularMotion_WITH_Ceramic_AND_Sisal

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:ArmChair(?armchair) ^
                Ontology1244033197062:Curtains(?curtains) ^
                Ontology1244033197062:Cushions(?cushions) ^
               Ontology1244033197062:Accessory(?accessory) ^
              Ontology1244033197062:Slipcovers(?slipcovers) ^
               Ontology1244033197062:Furniture(?furniture) ^
  Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
  Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:White_Colors, ?colors) →
  Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Circular_Motion)
   ^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Retro_1960) ^
     Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Fawn) ^
   Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Taupe) ^
  Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Dove_Grey)
   ^ Ontology1244033197062:has_as_Colors(?accessory, Ontology1244033197062:Taupe)
^ Ontology1244033197062:has_as_Colors(?accessory, Ontology1244033197062:Dove_Grey)
^ Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Bone_White)
  ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
             Ontology1244033197062:Retro_Circle_Motif_for_Curtains)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Linen)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
             Ontology1244033197062:Retro_Circle_Motif_for_Cushions)
^ Ontology1244033197062:has_as_Material(?slipcovers, Ontology1244033197062:Cotton)
               ^ Ontology1244033197062:has_as_Pattern(?slipcovers,
                   Ontology1244033197062:Grey_Taupe_Brushed)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                  Ontology1244033197062:Textured_Circle_Motif)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Wool)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                       Ontology1244033197062:Grey_Sateen)
^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Ceramic_Tile)
                ^ Ontology1244033197062:has_as_Pattern(?floor,
                    Ontology1244033197062:Leather_Effect)
  ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Sisal)
  ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Vinyl)
                ^ Ontology1244033197062:has_as_Pattern(?carpets,
                 Ontology1244033197062:Ivory_Toned_Faux_Maple)
```

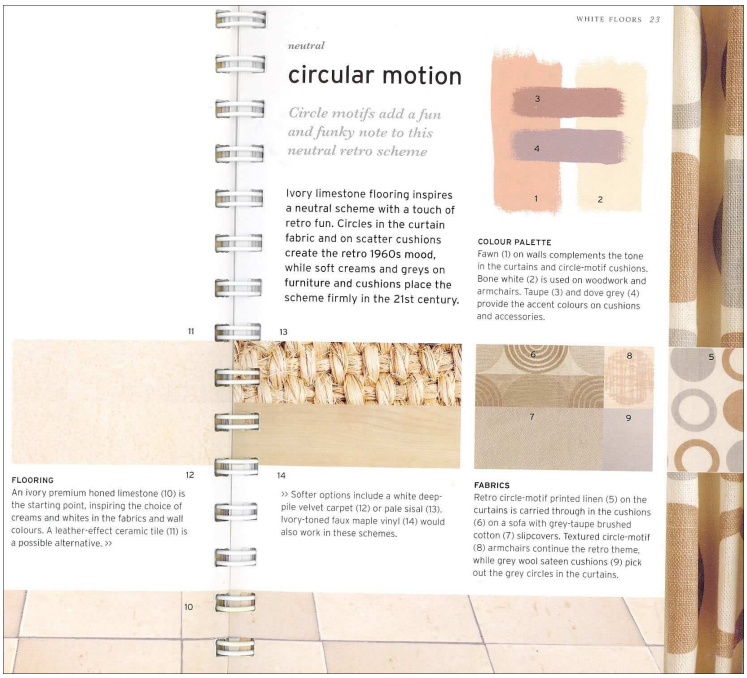
Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα "has_as_Material" για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε άσπρο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων White_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Circular_Motion ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Retro_1960 ως προσωπικότητας του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Fawn ως χρώμα για τους τοίχους, του Taupe και Dove_Grey για τα μαξιλάρια και τα αξεσουάρ, και του Bone_White για τα έπιπλα
- `has_as_Material`, για τον προσδιορισμό του Linen ως υλικό για τις κουρτίνες, του Linen και Wool για τα μαξιλάρια, του Cotton για τα καλύμματα, του Ceramic_Tile για το πάτωμα, και του Sisal και Vinyl για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Retro_Circle_Motif_for_Curtains ως σχέδιο για τις κουρτίνες, του Retro_Circle_Motif_for_Cushions και Grey_Sateen για τα μαξιλάρια, του Grey_Taupe_Brushed για τα καλύμματα, του Textured_Circle_Motif για τις πολυθρόνες, του Leather_Effect για το πάτωμα, και του Ivory_Toned_Faux_Maple για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE1:_IF_WhiteFloors_THEN_CircularMotion_WITH_Ceramic_AND_Velvet:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε τα Velvet και Vinyl ως υλικά για τα χαλιά, και μέσω του property atom has_as_Pattern, προσδιορίζουμε το Deep_Pile ως επιπλέον σχέδιο για τα χαλιά



Εικόνα 12. Συμπερασματολογία βάση άσπρων πατωμάτων

- `RULE1:_IF_WhiteFloors_THEN_CircularMotion_WITH_Limestone_AND_Sisal:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα. Χρησιμοποιώντας το property atom

has_as_Material, προσδιορίζουμε το Limestone ως υλικό για το πάτωμα, και μέσω του property atom has_as_Pattern, προσδιορίζουμε το Ivory_Premium_Honed ως σχέδιο για το πάτωμα

- RULE1:_IF_WhiteFloors_THEN_CircularMotion_WITH_Limestone_AND_Velvet:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα και τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Limestone ως υλικό για το πάτωμα, και μέσω του property atom has_as_Pattern, προσδιορίζουμε το Ivory_Premium_Honed ως σχέδιο για το πάτωμα. Στην περίπτωση των χαλιών, το property atom has_as_Material προσδιορίζει τα Velvet και Vinyl ως υλικά για τα χαλιά, και το property atom has_as_Pattern προσδιορίζει το Deep_Pile ως επιπλέον σχέδιο για τα χαλιά

SWRL Editor Name:
RULE2:_IF_BeigeFloors_THEN_ExoticAubergine_WITH_FauxTravertineCeramicTile_FOR_GlamorousMood

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Curtains(?curtains) ^
                Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Wall(?wall) ^ Ontology1244033197062:Cushions(?cushions) ^
  Ontology1244033197062:Chairs(?chairs) ^ Ontology1244033197062:Sofa(?sofa) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Beige_Colors, ?colors) →
 Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Exotic_Aubergine)
   ^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Glamorous) ^
                Ontology1244033197062:has_as_Functionality(?room,
                        Ontology1244033197062:Living_Room)
               ^ Ontology1244033197062:has_as_Functionality(?room,
                        Ontology1244033197062:Dining_Room)
 ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Aubergine)
^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Aubergine)
   ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Warm_Taupe)
                ^ Ontology1244033197062:has_as_Colors(?curtains,
                        Ontology1244033197062:Rich_Crimson)
  ^Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Fuchsia)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                     Ontology1244033197062:Stylized_Leaves)
                  ^ Ontology1244033197062:has_as_Pattern(?chairs,
                 Ontology1244033197062:Aubergine_and_Taupe_Circles)
                    ^ Ontology1244033197062:has_as_Pattern(?sofa,
               Ontology1244033197062:Pale_Taupe_with_Aubergine_Dots)
                  ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Fuchsia_and_Taupe_Stripes)
                  ^ Ontology1244033197062:has_as_Pattern(?cushions,
                 Ontology1244033197062:Aubergine_and_Fuchsia_Checks)
                    ^ Ontology1244033197062:has_as_Material(?floor,
                 Ontology1244033197062:Faux_Travertine_Ceramic_Tile)
                  ^ Ontology1244033197062:has_as_Pattern(?carpets,
                       Ontology1244033197062:Densely_Tufted)
  ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Vinyl)
                ^ Ontology1244033197062:has_as_Pattern(?carpets,
                     Ontology1244033197062:Limestone_Effect)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχει και ένα property atom, το "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε μπεζ χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Beige_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Exotic_Aubergine ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Glamorous ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Living_Room και Dining_Room ως λειτουργικές ικανότητες του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Aubergine ως χρώμα για τις κουρτίνες και την ταπετσαρία, του Warm_Taupe για τους τοίχους, του Fuchsia για τα μαξιλάρια, και επιπλέον του Rich_Crimson για τις κουρτίνες
- `has_as_Material`, για τον προσδιορισμό του Faux_Travertine_Ceramic_Tile ως υλικό για το πάτωμα, και του Vinyl για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Stylized_Leaves ως σχέδιο για τις κουρτίνες, του Aubergine_and_Taupe_Circles για τις καρέκλες, του Pale_Taupe_with_Aubergine_Dots για τους καναπέδες, του Fuchsia_and_Taupe_Stripes και Aubergine_and_Fuchsia_Circles για τα μαξιλάρια, και του Densely_Tufted και Limestone_Effect για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE2:_IF_BeigeFloors_THEN_ExoticAubergine_WITH_JerusalemGreyGoldGlossLimestone_FOR_GlamorousMood:`
  Η μόνη διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Jerusalem_Grey_and_Gold_Gloss_Limestone ως υλικό για το πάτωμα

Εικόνα 13. Συμπερασματολογία βάση μπεζ πατωμάτων

- `RULE2:_IF_BeigeFloors_THEN_ExoticAubergine_WITH_TravertineMarbleFloor_FOR_StrongMood:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα, προσδίδοντας μια πιο έντονη προσωπικότητα στο χώρο. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Travertine_Marble ως υλικό για το πάτωμα, ενώ με το has_as_Mood προσδιορίζουμε Strong προσωπικότητα
- `RULE2:_IF_CreamFloors_THEN_ExoticAubergine_WITH_FauxTravertineCeramicTile_FOR_GlamorousMood:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του property atom "has_as_Colors" στο antecedent μέρος του κανόνα, το οποίο θα προσδιορίσει πάτωμα που έχει ένα οποιοδήποτε ελαφρά κιτρινωπό χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Cream_Colors). Το consequent μέρος του κανόνα παραμένει απαράλλακτο, άρα θα συμπεράνει ακριβώς τα ίδια αποτελέσματα
- `RULE2:_IF_CreamFloors_THEN_ExoticAubergine_WITH_JerusalemGreyGoldGlossLimestone_FOR_GlamorousMood:`
  Το antecedent μέρος του συγκεκριμένου κανόνα παραμένει ίδιο με το antecedent μέρος του RULE2:_IF_CreamFloors_THEN_ExoticAubergine_WITH_FauxTravertineCeramicTile_FOR_GlamorousMood κανόνα, ενώ το consequent μέρος είναι όμοιο με το consequent μέρος του RULE2:_IF_BeigeFloors_THEN_ExoticAubergine_WITH_JerusalemGreyGoldGlossLimestone_FOR_GlamorousMood κανόνα. Με απλά λόγια θα τρέξει μόνο αν ισχύουν οι συνθήκες του 1ου κανόνα, και τότε θα συμπεράνει ίδια αποτελέσματα με τον 2ο κανόνα
- `RULE2:_IF_CreamFloors_THEN_ExoticAubergine_WITH_TravertineMarbleFloor_FOR_StrongMood:`
  Το antecedent μέρος του συγκεκριμένου κανόνα παραμένει ίδιο με το antecedent μέρος του RULE2:_IF_CreamFloors_THEN_ExoticAubergine_WITH_FauxTravertineCeramicTile_FOR_GlamorousMood κανόνα, ενώ το consequent μέρος είναι όμοιο με το consequent μέρος του RULE2:_IF_BeigeFloors_THEN_ExoticAubergine_WITH_TravertineMarbleFloor_FOR_StrongMood κανόνα. Με απλά λόγια θα τρέξει μόνο αν ισχύουν οι συνθήκες του 1ου κανόνα, και τότε θα συμπεράνει ίδια αποτελέσματα με τον 2ο κανόνα

SWRL Editor Name:
RULE3:_IF_OchreFloors_THEN_ChocolateToffees_WITH_Cork_AND_Linoleum

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^
Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Wall(?wall) ^
Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Ochre_Colors, ?colors) →
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Chocolate_Toffees)
^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Robust) ^
Ontology1244033197062:has_as_Functionality(?room,
Ontology1244033197062:Family_Room)
^ Ontology1244033197062:has_as_Colors(?upholstery,
Ontology1244033197062:Cocoa_Brown)
^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Toffee)
^ Ontology1244033197062:has_as_Colors(?curtains,
Ontology1244033197062:Butterscotch)
^ Ontology1244033197062:has_as_Colors(?cushions,
Ontology1244033197062:Butterscotch)
^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Warm_White)
^ Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Warm_White)
^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Cotton)
^ Ontology1244033197062:has_as_Pattern(?curtains,
Ontology1244033197062:Striped_in_tones_of_Cocoa_Toffee_WarmWhite_and_Butterscotch)
^ Ontology1244033197062:has_as_Material(?upholstery,
Ontology1244033197062:Faux_Leather)
^ Ontology1244033197062:has_as_Material(?upholstery,
Ontology1244033197062:Chenille)
^ Ontology1244033197062:has_as_Pattern(?upholstery,
Ontology1244033197062:Luscious_Toffee_and_Butterscotch)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Linen)
^ Ontology1244033197062:has_as_Pattern(?cushions,
Ontology1244033197062:Cocoa_and_Toffee_Ottoman_Textural_Accents)
^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Cork) ^
Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Linoleum)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε άσπρο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Ochre_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Chocolate_Toffees ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Robust ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Family_Room ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Butterscotch ως χρώμα για τις κουρτίνες και τα μαξιλάρια, του Cocoa_Brown Toffee για την ταπετσαρία, και του Warm_White για τους τοίχους και τα έπιπλα
- `has_as_Material`, για τον προσδιορισμό του Cotton ως υλικό για τις κουρτίνες, των Faux_Leather και Chenille για την ταπετσαρία, του Linen για τα μαξιλάρια, του Cork για το πάτωμα, και του Linoleum για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Luscious_Toffee_and_Butterscotch ως σχέδιο για την ταπετσαρία, του Cocoa_and_Toffee_Ottoman_Textural_Accents για τα μαξιλάρια, και του Striped_in_tones_of_Cocoa_Toffee_WarmWhite_and_Butterscotch για τις κουρτίνες

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE3:_IF_OchreFloors_THEN_ChocolateToffees_WITH_Cork_AND_Wool:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και ενός σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Wool ως υλικό για τα χαλιά και μέσω ενός καινούργιου property atom has_as_Pattern προσθέτουμε το Densely_Tufted ως σχέδιο για τα χαλιά

Εικόνα 14. Συμπερασματολογία βάση ωχρών πατωμάτων

- `RULE3:_IF_OchreFloors_THEN_ChocolateToffees_WITH_OakPlanks_AND_Linoleum:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Oak_Planks ως υλικό για το πάτωμα, ενώ όλα τα υπόλοιπα μέρη παραμένουν τα ίδια
- `RULE3:_IF_OchreFloors_THEN_ChocolateToffees_WITH_OakPlanks_AND_Wool:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα και τα χαλιά. Χρησιμοποιώντας τα property atoms has_as_Material, προσδιορίζουμε το Oak_Planks ως υλικό για το πάτωμα, και το Wool ως υλικό για τα χαλιά

- RULE3:_IF_OchreFloors_THEN_ChocolateToffees_WITH_VeinedMarble:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Veined_Marble ως υλικό για το πάτωμα, και παράλληλα αφαιρούμε class και property atoms που προσδιορίζουν χαλιά

SWRL Editor Name:
RULE4:_IF_TerracottaFloors_THEN_AutumnalGarden_WITH_Leather

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:ArmChair(?armchair) ^
                Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Sofa(?sofa) ^
                Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
  Ontology1244033197062:has_as_Colors(Ontology1244033197062:Terracotta_Colors,
                                      ?colors) →
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Autumnal_Garden)
  ^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Relaxing) ^
                Ontology1244033197062:has_as_Functionality(?room,
                        Ontology1244033197062:Living_Room)
              ^ Ontology1244033197062:has_as_Functionality(?room,
                        Ontology1244033197062:Family_Room)
 ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Ice_Blue) ^
                Ontology1244033197062:has_as_Colors(?curtains,
                Ontology1244033197062:Deep_Blue_Green_of_Ocean)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Brown_Sugar)
  ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Toffee)
                ^ Ontology1244033197062:has_as_Colors(?upholstery,
                        Ontology1244033197062:Brown_Sugar)
 ^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Toffee)
 ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
    Ontology1244033197062:Variety_of_Floral_Print_of_Blues_and_Spicy_Browns)
 ^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Suede)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                  Ontology1244033197062:Softly_Textural_Faux)
 ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille)
                  ^ Ontology1244033197062:has_as_Pattern(?sofa,
                    Ontology1244033197062:Inviting_Chenille)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Exotic_Blue_Pailsey)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Brown_Sugar_Faux)
 ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Leather)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχει και ένα property atom, το "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε πήλινο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Terracotta_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Autumnal_Garden ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Relaxing ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Living_Room και Family_Room ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Brown_Sugar και Toffee ως χρώματα για τις κουρτίνες και την ταπετσαρία, του Ice_Blue για τους τοίχους, και ως ένα επιπλέον χρώμα για τις κουρτίνες το Deep_Blue_Green_of_Ocean
- `has_as_Material`, για τον προσδιορισμό του Linen ως υλικό για τις κουρτίνες, του Suede για τις πολυθρόνες, του Chenille για τους καναπέδες, του Silk για τα μαξιλάρια, και του Leather για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Softly_Textural_Faux ως σχέδιο για τις πολυθρόνες, του Variety_of_Floral_Print_of_Blues_and_Spicy_Browns για τις κουρτίνες, του Inviting_Chenille για τους καναπέδες, και των Exotic_Blue_Pailsey και Brown_Sugar_Faux για τα μαξιλάρια

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE4:_IF_TerracottaFloors_THEN_AutumnalGarden_WITH_Persian:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και ενός σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Wool ως υλικό για τα χαλιά και μέσω ενός καινούργιου property atom has_as_Pattern προσθέτουμε το Rich_Ginger_with_Persian_Rug_Motifs ως σχέδιο για τα χαλιά

Εικόνα 15. Συμπερασματολογία βάση πήλινων πατωμάτων

- RULE4:_IF_TerracottaFloors_THEN_AutumnalGarden_WITH_Tile:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Terracotta_Tile ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Rustic_Distressed ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα
- RULE4:_IF_TerracottaFloors_THEN_AutumnalGarden_WITH_Wood:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Wood ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Oiled_Merbau ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα
- RULE4:_IF_TerracottaFloors_THEN_AutumnalGarden_WITH_Wool:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και ενός σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Wool ως υλικό για τα χαλιά και μέσω ενός καινούργιου property atom has_as_Pattern προσθέτουμε το Tufted_Chequerboard_Motif ως σχέδιο για τα χαλιά

SWRL Editor Name:
RULE5:_IF_GreenFloors_THEN_RelaxedAndRestful_WITH_Limestone_AND_Sisal

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Wall(?wall) ^
Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Curtains(?curtains) ^ Ontology1244033197062:Throw(?throw)
^ Ontology1244033197062:Cushions(?cushions) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Green_Colors, ?colors) →
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Relaxed_and_Restful)
^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Restful) ^
Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Mist) ^
Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Seagrass)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Seagrass)
^ Ontology1244033197062:has_as_Colors(?upholstery,
Ontology1244033197062:Dark_Brown)
^ Ontology1244033197062:has_as_Pattern(?curtains,
Ontology1244033197062:Seagrass_Linen)
^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen)
^Ontology1244033197062:has_as_Pattern(?upholstery,
Ontology1244033197062:Nubby_in_Brown_Taupe_Blue)
^ Ontology1244033197062:has_as_Material(?upholstery, Ontology1244033197062:Cotton)
^ Ontology1244033197062:has_as_Colors(?upholstery,
Ontology1244033197062:Powder_Blue)
^ Ontology1244033197062:has_as_Pattern(?upholstery,
Ontology1244033197062:Dark_Brown_Leather)
^ Ontology1244033197062:has_as_Material(?upholstery, Ontology1244033197062:Leather)
^ Ontology1244033197062:has_as_Pattern(?throw, Ontology1244033197062:Mohair_Throw)
^ Ontology1244033197062:has_as_Colors(?throw, Ontology1244033197062:Powder_Blue)
^ Ontology1244033197062:has_as_Material(?throw, Ontology1244033197062:Mohair) ^
Ontology1244033197062:has_as_Pattern(?cushions, Ontology1244033197062:Plaid) ^
Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Powder_Blue)
^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Limestone)
^ Ontology1244033197062:has_as_Pattern(?floor,
Ontology1244033197062:Limestone_Effect)
^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Sisal)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε πράσινο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Green_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων

ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Relaxed_and_Restful ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Restful ως προσωπικότητας του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Mist ως χρώμα για τους τοίχους, του Seagrass για τα έπιπλα και τις κουρτίνες, του Powder_Blue για την ταπετσαρία, τα ριχτάρια και τα μαξιλάρια, και ως επιπλέον χρώμα για την ταπετσαρία το Dark_Brown
- `has_as_Material`, για τον προσδιορισμό του Linen ως υλικό για τις κουρτίνες, των Cotton και Leather για την ταπετσαρία, του Mohair για τα ριχτάρια, του Slate για το πάτωμα, και των Vinyl και Wool για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Seagrass_Linen ως σχέδιο για τις κουρτίνες, των Nubby_in_Brown_Taupe_Blue και Dark_Brown_Leather για την ταπετσαρία, του Mohair_Throw για τα ριχτάρια, του Plaid για τα μαξιλάρια, του Grey_Green_Slate για το πάτωμα, και των Slate_Effect και Grey_Woollen για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE5:_IF_GreenFloors_THEN_RelaxedAndRestful_WITH_Slate_AND_Vinyl_Wool:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα και τα χαλιά. Επίσης, δεν υπάρχει προσδιορισμός σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Limestone ως υλικό για το πάτωμα, και το Sisal για τα χαλιά. Παράλληλα, εφόσον δεν ζητείται σχέδιο για τα χαλιά, αφαιρούμε εκείνο το property atom has_as_Pattern, που προσδιορίζει το σχέδιο για τα χαλιά

Εικόνα 16. Συμπερασματολογία βάση πράσινων πατωμάτων

SWRL Editor Name:
RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Limestone

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:Furniture(?furniture) ^
  Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
                 Ontology1244033197062:Curtains(?curtains) ^
                 Ontology1244033197062:Cushions(?cushions) ^
               Ontology1244033197062:Upholstery(?upholstery) ^
    Ontology1244033197062:Chairs(?chairs) ^ Ontology1244033197062:Sofa(?sofa) ^
  Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
   Ontology1244033197062:has_as_Colors(Ontology1244033197062:Light_Brown_Colors,
                                       ?colors) →
  Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Moorish_Sands) ^
     Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Dynamic) ^
                  Ontology1244033197062:has_as_Colors(?wall,
                     Ontology1244033197062:Rich_Clotted_Cream)
                ^ Ontology1244033197062:has_as_Colors(?furniture,
                     Ontology1244033197062:Rich_Clotted_Cream)
                 ^ Ontology1244033197062:has_as_Colors(?curtains,
                     Ontology1244033197062:Rich_Clotted_Cream)
                ^ Ontology1244033197062:has_as_Colors(?cushions,
                     Ontology1244033197062:Rich_Clotted_Cream)
  ^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Honey)
                ^ Ontology1244033197062:has_as_Colors(?upholstery,
                       Ontology1244033197062:Warm_Taupe)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Pale_Wheat)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                      Ontology1244033197062:Printed_Scrim)
  ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen)
               ^ Ontology1244033197062:has_as_Pattern(?upholstery,
                    Ontology1244033197062:Horizontally_Striped)
   ^ Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Honey) ^
     Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Honey) ^
  Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Warm_Taupe)
  ^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Warm_Taupe)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                     Ontology1244033197062:Coarse_Woven_Raw)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Lustrous_Cream)
 ^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Limestone)
   ^ Ontology1244033197062:has_as_Pattern(?floor, Ontology1244033197062:Mosaic)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα "has_as_Material" για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε ανοικτό-καφέ χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Light_Brown_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Moorish_Sands ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Dynamic ως προσωπικότητας του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Rich_Clotted_Cream ως χρώμα για τους τοίχους, τα έπιπλα, τις κουρτίνες και τα μαξιλάρια, ενώ προσδιορίζεται και το επιπλέον χρώμα Pale_Wheat για τα μαξιλάρια. Επίσης, προσδιορίζονται τα χρώματα Warm_Taupe και Honey για την ταπετσαρία, τις καρέκλες και τους καναπέδες
- `has_as_Material`, για τον προσδιορισμό του Linen ως υλικό για τις κουρτίνες, του Silk για τα μαξιλάρια, και του Limestone για το πάτωμα
- `has_as_Pattern`, για τον προσδιορισμό του Printed_Scrim ως σχέδιο για τις κουρτίνες, του Horizontally_Striped για την ταπετσαρία, των Coarse_Woven_Raw και Lustrous_Cream για τα μαξιλάρια, και του Mosaic για το πάτωμα

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Quartz:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Quartz ως το υλικό για το πάτωμα και μέσω του property atom has_as_Pattern προσδιορίζουμε το Polished_ManMade ως το σχέδιο για το πάτωμα

Εικόνα 17. Συμπερασματολογία βάση ανοικτών-καφέ πατωμάτων

- RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Sisal_FOR_Bedroom:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στην προσθήκη μιας ενδεικτικής λειτουργικότητας χώρου, ενός υλικού και σχεδίου για τα χαλιά, και στην απουσία αναφοράς υλικού και σχεδίου για το πάτωμα. Προσθέτοντας τρία καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Sisal ως υλικό για τα χαλιά, μέσω του has_as_Pattern, προσδιορίζουμε το Striped ως σχέδιο για τα χαλιά, και μέσω του has_as_Functionality, προσδιορίζουμε το Bedroom ως πιθανή λειτουργικότητα του δωματίου. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν πάτωμα, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για τα χαλιά
- RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Wool_FOR_Bedroom:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Bedroom που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Sisal_FOR_Bedroom. Οι μόνες διαφορές τους, εντοπίζονται στους προσδιορισμούς του υλικού Wool και του σχεδίου Tufted_in_Warm_Almond για τα χαλιά
- RULE6:_IF_LightBrownFloors_THEN_MoorishSands_WITH_Woven:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στην προσθήκη ενός σχεδίου για τα χαλιά και στην απουσία αναφοράς υλικού και σχεδίου για το πάτωμα. Προσθέτοντας το property atom has_as_Pattern, προσδιορίζουμε το Striped_Woven ως σχέδιο για τα χαλιά, αφού έχουμε προσθέσει το απαραίτητο class atom στο antecedent μέρος του κανόνα. Παράλληλα, αφαιρούμε από το ίδιο μέρος την αναφορά για το πάτωμα και διαγράφουμε τα σχετικά properties atoms

SWRL Editor Name:
RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Limestone_AND_Flatweave

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Wall(?wall) ^
Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:ArmChair(?armchair) ^
                Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
  Ontology1244033197062:has_as_Colors(Ontology1244033197062:Mid_Brown_Colors,
                                      ?colors) →
 Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Josephines_Roses)
   ^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Romantic) ^
Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Ocean_Blue)
^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Lavender)
^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Pale_Lime)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Ocean_Blue)
 ^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Lavender)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Pale_Lime)
  ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Pale_Yellow)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Ocean_Blue)
 ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Lavender)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Pale_Lime)
                ^ Ontology1244033197062:has_as_Colors(?curtains,
                      Ontology1244033197062:Butterscotch)
  ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                      Ontology1244033197062:French_Roses)
  ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Cotton) ^
                  Ontology1244033197062:has_as_Pattern(?sofa,
        Ontology1244033197062:Woven_in_tones_of_Ocean_Butterscotch_Cream)
  ^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Ocean_Blue)
 ^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Butterscotch)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Chenille)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Lavender_Chenille)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Linen)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                     Ontology1244033197062:Pale_Lime_Linen)
  ^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Linen)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                    Ontology1244033197062:Butterscotch_Linen)
                 ^ Ontology1244033197062:has_as_Colors(?armchair,
                      Ontology1244033197062:Butterscotch)
 ^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Limestone)
                  ^ Ontology1244033197062:has_as_Pattern(?floor,
               Ontology1244033197062:Honed_with_Large_Squared_Tiles)
  ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Wool) ^
 Ontology1244033197062:has_as_Pattern(?carpets, Ontology1244033197062:Flatweave)
```

Επεξήγηση και Αποτέλεσμα:

Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχει και ένα property atom, το "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε καφέ χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Mid_Brown_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Josephines_Roses ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Romantic ως προσωπικότητας του δωματίου
- `has_as_Colors`, για τον προσδιορισμό των Ocean_Blue, Lavender, και Pale_Lime ως χρώματα για την ταπετσαρία, τα μαξιλάρια, και τις κουρτίνες, ενώ το χρώμα Pale_Yellow προσδιορίζει τους τοίχους. Το Butterscotch χρησιμοποιείται στις πολυθρόνες, ενώ μαζί με το Ocean_Blue αποδίδεται και στους καναπέδες
- `has_as_Material`, για τον προσδιορισμό του Silk ως υλικό για τις κουρτίνες, του Cotton για τους καναπέδες, των Chenille και Linen για τα μαξιλάρια, του Linen για τις πολυθρόνες, του Limestone για το πάτωμα, και του Wool για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του French_Roses ως σχέδιο για τις κουρτίνες, του Woven_in_tones_of_Ocean_Butterscotch_Cream για τους καναπέδες, των Lavender_Chenille και Pale_Lime_Linen για τα μαξιλάρια, του Butterscotch_Linen για τις πολυθρόνες, του Honed_with_Large_Squared_Tiles για το πάτωμα, και του Flatweave για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Limestone_AND_Tufted:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του σχεδίου και ενός χρώματος για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Pattern προσδιορίζουμε το Tufted_Wilton ως σχέδιο για τα χαλιά, και μέσω ενός καινούργιου property atom has_as_Colors αποδίδουμε ως χρώμα το Warm_Taupe για τα χαλιά

Εικόνα 18. Συμπερασματολογία βάση καφέ πατωμάτων

- `RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Vinyl_AND_Flatweave:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material προσδιορίζουμε το Vinyl ως υλικό για το πάτωμα, και μέσω του has_as_Pattern προσδιορίζουμε το High_Quality ως σχέδιο για το πάτωμα

- RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Vinyl_AND_Tufted:
  Ο συγκεκριμένος κανόνας αποτελεί συνδυασμό των δύο προηγουμένων καταστάσεων. Μέσω του RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Limestone_AND_Tufted κανόνα προσδιορίζουμε το προαναφερθέν ζητούμενο σχέδιο και χρώμα για τα χαλιά, ενώ μέσω του κανόνα RULE7:_IF_MidBrownFloors_THEN_JosephinesRoses_WITH_Vinyl_AND_Flatweave προσδιορίζουμε το υλικό και το σχέδιο για το πάτωμα

SWRL Editor Name:
RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_BlackOak_AND_Sisal

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Furniture(?furniture)
^ Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood)
                ^ Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Wall(?wall) ^ Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:ArmChair(?armchair) ^ Ontology1244033197062:Sofa(?sofa) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Floor(?floor)
            ^ Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
  Ontology1244033197062:has_as_Colors(Ontology1244033197062:Dark_Brown_Colors,
                                      ?colors)
                  ^ Ontology1244033197062:Carpets(?carpets) →
   Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Glamorous) ^
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Taupe_Baroque) ^
                Ontology1244033197062:has_as_Functionality(?room,
                    Ontology1244033197062:Living_Room)
 ^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Fawn)
 ^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Sand)
                ^ Ontology1244033197062:has_as_Colors(?upholstery,
                  Ontology1244033197062:Bittersweet_Chocolate)
^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:French_Vanilla)
                ^ Ontology1244033197062:has_as_Colors(?furniture,
                    Ontology1244033197062:French_Vanilla)
 ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Fawn) ^
  Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Sand) ^
                Ontology1244033197062:has_as_Pattern(?curtains,
                    Ontology1244033197062:Floral_Baroque)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                    Ontology1244033197062:Deep_Chocolate)
^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Chenille)
   ^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Fawn) ^
    Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Sand) ^
  Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Cotton) ^
   Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Silk) ^
                  Ontology1244033197062:has_as_Pattern(?sofa,
                 Ontology1244033197062:Taupe_Cotton_Silk_Blend)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Striped_Faux)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Linen)
^ Ontology1244033197062:has_as_Pattern(?cushions, Ontology1244033197062:Viscose)
 ^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Sand) ^
Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Black_Oak)
                ^ Ontology1244033197062:has_as_Pattern(?floor,
                  Ontology1244033197062:Ultra_Matt_Lacquered)
 ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Sisal)
                ^ Ontology1244033197062:has_as_Colors(?carpets,
                  Ontology1244033197062:Chocolate_Brown)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα "has_as_Material" για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε σκούρο-καφέ χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Dark_Brown_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Taupe_Baroque ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Glamorous ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Living_Room ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του French_Vanilla ως χρώμα για τους τοίχους και τα έπιπλα. Για τα μαξιλάρια προσδίδεται το χρώμα Sand, ενώ το Chocolate_Brown για τα χαλιά. Μετά, ακολουθούν τα Fawn και Sand για τον χρωματισμό της ταπετσαρίας, των κουρτινών και των καναπέδων, ενώ στην περίπτωση της ταπετσαρίας χρησιμοποιείται και το Bittersweet_Chocolate
- `has_as_Material`, για τον προσδιορισμό του Chenille ως υλικό για τις πολυθρόνες, των Cotton και Silk για τους καναπέδες, των Silk και Linen για τα μαξιλάρια, του Black_Oak για το πάτωμα, και του Sisal για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Floral_Baroque ως σχέδιο για τις κουρτίνες, του Deep_Chocolate για τις πολυθρόνες, του Taupe_Cotton_Silk_Blend για τους καναπέδες, των Striped_Faux, Viscose για τα μαξιλάρια, και του Ultra_Matt_Lacquered για το πάτωμα

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_BlackOak_AND_Wool:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και ενός σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Wool ως το υλικό για τα χαλιά, και δημιουργώντας ένα καινούργιο property atom has_as_Pattern προσδιορίζουμε το Deep_Pile ως σχέδιο για τα χαλιά

Εικόνα 19. Συμπερασματολογία βάση σκούρο-καφέ πατωμάτων

- RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_CeramicTile_AND_Sisal:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Ceramic_Tile ως υλικό για το πάτωμα και μέσω του property atom has_as_Pattern προσδιορίζουμε το Leather_Effect ως σχέδιο για το πάτωμα
- RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_CeramicTile_AND_Wool:
  Ο συγκεκριμένος κανόνας αποτελεί ένα συνδυασμό των δύο προηγούμενων, κρατώντας τους προσδιορισμούς του υλικού και του σχεδίου του χαλιού που καθορίζονται στον κανόνα RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_BlackOak_AND_Wool, ενώ από τον RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_CeramicTile_AND_Sisal κρατάει τους προσδιορισμούς του υλικού και του σχεδίου για το πάτωμα
- RULE8:_IF_DarkBrownFloors_THEN_TaupeBaroque_WITH_Leather:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για το πάτωμα, ενώ δεν ακολουθεί προσδιορισμός σχεδίου για αυτό. Άρα, χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Leather ως υλικό για το πάτωμα, ενώ αφαιρούμε το αντίστοιχο has_as_Pattern που προσδιορίζει το σχέδιο για το πάτωμα. Παράλληλα, εφόσον δεν χρησιμοποιούνται χαλιά, αφαιρούμε το class atom που αφορά τα χαλιά, και το αντίστοιχο property atom has_as_Material, που προσδιορίζει το υλικό για τα χαλιά

SWRL Editor Name:
RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Cork_FOR_FamilyRoom

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Furniture(?furniture)
^ Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood)
^ Ontology1244033197062:Wall(?wall) ^ Ontology1244033197062:Curtains(?curtains)
 ^ Ontology1244033197062:Chairs(?chairs) ^ Ontology1244033197062:Sofa(?sofa) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Floor(?floor)
            ^ Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Black_Colors, ?colors) →
               Ontology1244033197062:has_as_Functionality(?room,
                    Ontology1244033197062:Family_Room)
 ^ Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Black_Magic) ^
 Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Ivory_White) ^
Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Ivory_White)
 ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Black) ^
Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Ivory_White)
 ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Felt)
               ^ Ontology1244033197062:has_as_Pattern(?curtains,
                 Ontology1244033197062:Inky_with_Ribbon_Spirals)
  ^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Black) ^
 Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Ivory_White) ^
  Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille) ^
Ontology1244033197062:has_as_Pattern(?sofa, Ontology1244033197062:Circle_Motif_Cut)
  ^ Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Rose) ^
 Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Ivory_White)
  ^ Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Taupe) ^
  Ontology1244033197062:has_as_Material(?chairs, Ontology1244033197062:Linen) ^
                 Ontology1244033197062:has_as_Pattern(?chairs,
           Ontology1244033197062:Accent_in_Red_White_Taupe_Printed)
 ^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Shadow)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Mohair)
               ^ Ontology1244033197062:has_as_Pattern(?cushions,
                 Ontology1244033197062:Mohair_in_Shadow_Grey)
 ^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Rose) ^
Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Velvet)
               ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Distressed_Red)
 ^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Cork) ^
 Ontology1244033197062:has_as_Pattern(?floor, Ontology1244033197062:Robust_Tile)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε μαύρο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Black_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Black_Magic ως στυλ δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Family_Room ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Ivory_White ως χρώμα για τους τοίχους, τα έπιπλα, τις κουρτίνες, τους καναπέδες, και τις καρέκλες. Επίσης, κουρτίνες και καναπέδες χρωματίζονται και με Black, ενώ οι καρέκλες και με Rose. Τα χρώματα Rose και Shadow αποδίδονται στα μαξιλάρια
- `has_as_Material`, για τον προσδιορισμό του Felt ως υλικό για τις κουρτίνες, του Chenille για τους καναπέδες, του Linen για τις καρέκλες, των Mohair και Velvet για τα μαξιλάρια, και του Cork για το πάτωμα
- `has_as_Pattern`, για τον προσδιορισμό του Inky_with_Ribbon_Spirals ως σχέδιο για τις κουρτίνες, του Accent_in_Red_White_Taupe_Printed για τις καρέκλες, του Circle_Motif_Cut για τους καναπέδες, των Mohair_in_Shadow_Grey και Distressed_Red για τα μαξιλάρια, και του Robust_Tile για το πάτωμα

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Limestone:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Limestone ως το υλικό για το πάτωμα και μέσω του property atom has_as_Pattern προσδιορίζουμε το Honed_with_Grey_Taupe ως το σχέδιο για το πάτωμα

Εικόνα 20. Συμπερασματολογία βάση μαύρων ή γκρι πατωμάτων

- `RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Marble_FOR_LivingRoom:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού, του σχεδίου και ενός χρωματισμού για το πάτωμα. Χρησιμοποιώντας το property atom has_as_Material, προσδιορίζουμε το Marble ως το υλικό για το πάτωμα και μέσω του property atom has_as_Pattern προσδιορίζουμε το Marble_in_Shadow_Grey_and_Brown ως το σχέδιο για το πάτωμα. Επίσης, προσθέτοντας ένα καινούργιο property atom has_as_Colors προσδιορίζουμε τα Shadow και Brown ως χρώματα για το πάτωμα και μέσω του property atom has_as_Functionality τονίζουμε ότι το δωμάτιο αφορά λειτουργικότητα Living_Room
- `RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Striped:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στην προσθήκη ενός υλικού και σχεδίου για τα χαλιά, και παράλληλα αφαιρούμε τις εξαρτήσεις που αφορούν πάτωμα. Προσθέτοντας το property atom has_as_Material, προσδιορίζουμε το Wool ως υλικό για τα χαλιά και μέσω ενός καινούργιου property atom has_as_Pattern προσδιορίζουμε το

Striped_Flatweave ως σχέδιο για τα χαλιά. Όπως προαναφέραμε ήδη, διαγράφουμε τα class και property atoms που αφορούν το πάτωμα

- RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Wool_FOR_BedRoom:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στην προσθήκη ενός υλικού και σχεδίου για τα χαλιά, και παράλληλα αφαιρούμε τις εξαρτήσεις που αφορούν πάτωμα. Προσθέτοντας το property atom has_as_Material, προσδιορίζουμε το Wool ως υλικό για τα χαλιά και μέσω ενός καινούργιου property atom has_as_Pattern προσδιορίζουμε το Loop_Pile ως σχέδιο για τα χαλιά. Όπως προαναφέραμε ήδη, διαγράφουμε τα class και property atoms που αφορούν το πάτωμα και μέσω του property atom has_as_Functionality τονίζουμε ότι το δωμάτιο αφορά λειτουργικότητα Bedroom
- RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Wool_FOR_LivingRoom:
  Πρόκειται για ακριβώς τον ίδιο κανόνα με τον προηγούμενο στην ιεραρχία παραδειγμάτων μας (RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Wool_FOR_BedRoom), με την διαφορά ότι η λειτουργικότητα του δωματίου μπορεί να είναι και Living_Room. Στην περίπτωση που η ιδιότητα has_as_Functionality καθοριστεί ως functional μέσα από την καρτέλα ιδιοτήτων του Protégé-OWL, τότε κρίνεται απαραίτητη αυτή η διαδικασία κλωνοποίησης των κανόνων
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Cork_FOR_FamilyRoom:
  Ο συγκεκριμένος κανόνας διαφοροποιείται σε σχέση με τον κανόνα του παραδείγματος στο antecedent μέρος, το οποίο επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος του είναι ίδιο με αυτό του RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Cork_FOR_FamilyRoom, οπότε θα έχει και την ίδια συμπερασματολογία
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Limestone:
  Ο συγκεκριμένος κανόνας διαφοροποιείται στο antecedent μέρος του σε σχέση με τον κανόνα RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Limestone, αφού τώρα επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος και των δύο κανόνων είναι ίδιο, οπότε θα έχει την ίδια συμπερασματολογία
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Marble_FOR_LivingRoom:
  Ο συγκεκριμένος κανόνας διαφοροποιείται στο antecedent μέρος του σε σχέση με τον κανόνα RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Marble_FOR_LivingRoom, αφού τώρα επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος και των δύο κανόνων είναι ίδιο, οπότε θα έχει την ίδια συμπερασματολογία
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Striped:
  Ο συγκεκριμένος κανόνας διαφοροποιείται στο antecedent μέρος του σε σχέση με τον κανόνα RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Striped, αφού τώρα επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος και των δύο κανόνων είναι ίδιο, οπότε θα έχει την ίδια συμπερασματολογία
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Wool_FOR_BedRoom:
  Ο συγκεκριμένος κανόνας διαφοροποιείται στο antecedent μέρος του σε σχέση με τον κανόνα RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Wool_FOR_BedRoom, αφού τώρα επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος και των δύο κανόνων είναι ίδιο, οπότε θα έχει την ίδια συμπερασματολογία
- RULE9:_IF_GreyFloors_THEN_BlackMagic_WITH_Wool_FOR_LivingRoom:
  Ο συγκεκριμένος κανόνας διαφοροποιείται στο antecedent μέρος του σε σχέση με τον κανόνα RULE9:_IF_BlackFloors_THEN_BlackMagic_WITH_Wool_FOR_LivingRoom, αφού τώρα επαληθεύεται για πατώματα που φέρουν κάποιο γκρι χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Gray_Colors). Ωστόσο, το consequent μέρος και των δύο κανόνων είναι ίδιο, οπότε θα έχει την ίδια συμπερασματολογία

SWRL Editor Name:
RULE10:_IF_RedFloors_THEN_MagicLanterns_FOR_Marble_WITH_ShagPile

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Chairs(?chairs) ^
Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Wall(?wall) ^
Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Red_Colors, ?colors) →
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Magic_Lanterns)
^ Ontology1244033197062:has_as_Functionality(?room,
Ontology1244033197062:Living_Room)
^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Sand) ^
Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Off_White)
^ Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Honey) ^
Ontology1244033197062:has_as_Pattern(?curtains,
Ontology1244033197062:Oriental_Floral_Print_on_Ramie)
^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille)
^ Ontology1244033197062:has_as_Pattern(?sofa,
Ontology1244033197062:Striped_Chenille)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Felt)
^ Ontology1244033197062:has_as_Pattern(?cushions,
Ontology1244033197062:Swimming_Pool_Blue_Patterned)
^ Ontology1244033197062:has_as_Colors(?cushions,
Ontology1244033197062:Swimming_Pool_Blue)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Velvet)
^ Ontology1244033197062:has_as_Pattern(?cushions,
Ontology1244033197062:Rose_Patterned)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Rose) ^
Ontology1244033197062:has_as_Material(?chairs, Ontology1244033197062:Chenille)
^ Ontology1244033197062:has_as_Pattern(?chairs,
Ontology1244033197062:Cut_Pile_in_Oval_Motif)
^ Ontology1244033197062:has_as_Colors(?chairs, Ontology1244033197062:Rose) ^
Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Marble) ^
Ontology1244033197062:has_as_Pattern(?floor, Ontology1244033197062:Red_High_Gloss)
^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Shag) ^
Ontology1244033197062:has_as_Pattern(?carpets,
Ontology1244033197062:Shag_Pile_in_Cherry_and_Honey)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε κόκκινο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Red_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Magic_Lanterns ως στυλ δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Living_Room ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Sand ως χρώμα για τους τοίχους, του Off_White για τα έπιπλα, του Honey για τους καναπέδες, του Rose για τις καρέκλες, και των Swimming_Pool_Blue και Rose για τα μαξιλάρια
- `has_as_Material`, για τον προσδιορισμό του Chenille ως υλικό για τους καναπέδες και τις καρέκλες, των Felt και Velvet για τα μαξιλάρια, του Marble για το πάτωμα, και του Shag για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Oriental_Floral_Print_on_Ramie ως σχέδιο για τις κουρτίνες, του Striped_Chenille για τους καναπέδες, των Rose_Patterned και Swimming_Pool_Blue_Patterned για τα μαξιλάρια, του Cut_Pile_in_Oval_Motif για τις καρέκλες, του Red_High_Gloss για το πάτωμα, και του Shag_Pile_in_Cherry_and_Honey για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE10:_IF_RedFloors_THEN_MagicLanterns_FOR_Marble_WITH_Wool:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για τα χαλιά. Χρησιμοποιώντας δύο property atom has_as_Material, προσδιορίζουμε τα Wool και Velvet ως υλικά για τα χαλιά και μέσω του property atom has_as_Pattern προσδιορίζουμε το Plush_in_Rich_Cherry ως σχέδιο για τα χαλιά

Εικόνα 21. Συμπερασματολογία βάση κόκκινων πατωμάτων

- `RULE10:_IF_RedFloors_THEN_MagicLanterns_FOR_Resin_WITH_Linoleum:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού και του σχεδίου για χαλιά και πάτωμα. Χρησιμοποιώντας δύο property atom has_as_Material, προσδιορίζουμε το Resin ως υλικό για το πάτωμα και το Linoleum ως υλικό για τα χαλιά. Παράλληλα μέσω άλλων δύο property atom has_as_Pattern προσδιορίζουμε το Textural_Poured ως σχέδιο για το πάτωμα και το Multitude_of_Hues ως σχέδιο για τα χαλιά

SWRL Editor Name:
RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_CoralWool

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Curtains(?curtains) ^
                Ontology1244033197062:Cushions(?cushions) ^
              Ontology1244033197062:Upholstery(?upholstery) ^
  Ontology1244033197062:ArmChair(?armchair) ^ Ontology1244033197062:Sofa(?sofa) ^
  Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Orange_Colors, ?colors) →
  Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Indian_Garden) ^
      Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Joyful) ^
 Ontology1244033197062:has_as_Functionality(?room, Ontology1244033197062:Bedroom)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Soft_Peach)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Soft_Peach)
  ^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Khaki)
^ Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Wisteria)
                ^ Ontology1244033197062:has_as_Colors(?curtains,
                      Ontology1244033197062:Lipstick_Red)
 ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Chintz)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                   Ontology1244033197062:Indian_Floral_Print)
  ^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Linen)
 ^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
              Ontology1244033197062:Large_Striped_in_Wisteria_Tones)
   ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Linen) ^
    Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Cotton) ^
      Ontology1244033197062:has_as_Colors(?sofa, Ontology1244033197062:Khaki) ^
  Ontology1244033197062:has_as_Pattern(?sofa, Ontology1244033197062:Green_Woven)
^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Chenille)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Peach_Chenille)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Velvet)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                     Ontology1244033197062:Cut_Stripes)
 ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Wool) ^
                  Ontology1244033197062:has_as_Pattern(?carpets,
                  Ontology1244033197062:Coral_Striped_Flatweave)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχει και ένα property atom, το "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε πορτοκαλί χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Orange_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Indian_Garden ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Joyful ως προσωπικότητας του δωματίου

- `has_as_Functionality`, για τον προσδιορισμό της Bedroom ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Soft_Peach ως χρώμα για τις κουρτίνες και τα μαξιλάρια, ενώ προσδιορίζεται και το επιπλέον χρώμα Lipstick_Red για τις κουρτίνες. Επίσης, προσδιορίζεται το χρώμα Khaki για την ταπετσαρία, και τους καναπέδες, με το Wisteria να αποτελεί συνοδευτικό χρώμα της ταπετσαρίας
- `has_as_Material`, για τον προσδιορισμό των Linen και Cotton ως υλικά για τις πολυθρόνες και τις καρέκλες, του Chintz για τις κουρτίνες,, των Chenille και Velvet για τα μαξιλάρια, και του Wool για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Indian_Floral_Print ως σχέδιο για τις κουρτίνες, του Large_Striped_in_Wisteria_Tones για τις πολυθρόνες, του Green_Woven για τους καναπέδες, των Peach_Chenille και Cut_Stripes για τα μαξιλάρια, και του Coral_Striped_Flatweave για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_Leather:`

Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Leather ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Natural_Toned ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα

Εικόνα 22. Συμπερασματολογία βάση πορτοκαλί πατωμάτων

- `RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_PeachWool:`

Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Pattern, προσδιορίζουμε το Peach_Twist ως το σχέδιο για τα χαλιά, με το υπόλοιπο μέρος του κανόνα να μένει απαράλλακτο, οπότε να έχει και την ίδια συμπερασματολογία

- `RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_TrellisPatterned:`

Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του σχεδίου για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Pattern,

προσδιορίζουμε το Trellis_Patterned_Loop ως το σχέδιο για τα χαλιά, με το υπόλοιπο μέρος του κανόνα να μένει απαράλλακτο, οπότε να έχει και την ίδια συμπερασματολογία

- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Hallway_WITH_Stone:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Hallway. Παράλληλα, προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Ceramic_Tile ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Stone_Effect ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα
- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Kitchen_WITH_Stone:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον προηγούμενο σε αυτήν την ακολουθία (RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Hallway_WITH_Stone), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_ Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Kitchen, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος

- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_LivingRoom_WITH_CoralWool:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον κανόνα του παραδείγματός μας (RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_CoralWool), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_LivingRoom_WITH_Leather:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον δεύτερο κανόνα σε αυτήν την ιεραρχία (RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_Leather), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_LivingRoom_WITH_PeachWool:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον τρίτο κανόνα σε αυτήν την ιεραρχία (RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_PeachWool), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας ίδιο το antecedent και το υπόλοιπο consequent μέρος
- RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_LivingRoom_WITH_TrellisPatterned:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον τέταρτο κανόνα σε αυτήν την ιεραρχία (RULE11:_IF_OrangeFloors_THEN_IndianGarden_FOR_Bedroom_WITH_TrellisPatterned), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας ίδιο το antecedent και το υπόλοιπο consequent μέρος

SWRL Editor Name:
RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Bedroom_WITH_Axminster

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Wall(?wall) ^
  Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:Chairs(?chairs) ^
                  Ontology1244033197062:Carpets(?carpets) ^
                Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
                    Ontology1244033197062:Floor(?floor) ^
              Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Yellow_Colors, ?colors) →
   Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Mellow_Mood) ^
     Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Relaxing) ^
 Ontology1244033197062:has_as_Functionality(?room, Ontology1244033197062:Bedroom)
 ^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Honey) ^
Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Butterscotch)
 ^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Honey) ^
Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Butterscotch)
^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Ecru) ^
   Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Ecru) ^
  Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen) ^
   Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Silk) ^
                Ontology1244033197062:has_as_Pattern(?curtains,
                Ontology1244033197062:Weave_of_Golden_Leaves)
  ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille)
                ^ Ontology1244033197062:has_as_Pattern(?sofa,
      Ontology1244033197062:Woven_in_tones_of_Honey_Butterscotch_BrownSugar)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
^ Ontology1244033197062:has_as_Pattern(?cushions, Ontology1244033197062:Textural)
  ^ Ontology1244033197062:has_as_Material(?chairs, Ontology1244033197062:Wool) ^
Ontology1244033197062:has_as_Pattern(?chairs, Ontology1244033197062:Woven_Daisy)
 ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Wool) ^
                Ontology1244033197062:has_as_Pattern(?carpets,
        Ontology1244033197062:Woven_Axminster_with_Subtle_Diamond_Motif)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε κίτρινο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Yellow_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Mellow_Mood ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Relaxing ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Bedroom ως λειτουργική ικανότητα του δωματίου

- `has_as_Colors`, για τον προσδιορισμό του Ecru ως χρώμα για τους τοίχους και τα έπιπλα. Για τον χρωματισμό των κουρτινών και των μαξιλαριών προσδιορίζονται τα χρώματα Honey και Butterscotch
- `has_as_Material`, για τον προσδιορισμό του Silk ως υλικό για κουρτίνες και μαξιλάρια, με το Linen να αποτελεί μια επιπλέον λύση για τις κουρτίνες. Επίσης, προσδιορίζονται το Chenille για τους καναπέδες, και το Wool για τις καρέκλες και τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Weave_of_Golden_Leaves ως σχέδιο για τις κουρτίνες, του Woven_in_tones_of_Honey_Butterscotch_BrownSugar για τους καναπέδες, του Textural για τα μαξιλάρια, του Woven_Axminster_with_Subtle_Diamond_Motif για τα χαλιά, και του Woven_Daisy για τις καρέκλες

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Bedroom_WITH_Chunky:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό του υλικού για τα χαλιά. Χρησιμοποιώντας το property atom has_as_Pattern προσδιορίζουμε το Chunky_Flatweave ως το σχέδιο για τα χαλιά, κρατώντας ίδιο το antecedent και το υπόλοιπο consequent μέρος

Εικόνα 23. Συμπερασματολογία βάση κίτρινων πατωμάτων

- `RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Bedroom_WITH_Speckled:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός επιπλέον υλικού και διαφορετικού σχεδίου για τα χαλιά. Χρησιμοποιώντας ένα καινούργιο property atom has_as_Material, προσδιορίζουμε και το Velvet ως υλικό για τα χαλιά και μέσω του property atom has_as_Pattern προσδιορίζουμε το Speckled ως σχέδιο για τα χαλιά
- `RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_DiningRoom:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Dining_Room. Παράλληλα, προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Marble ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Antique_Gold ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα

- RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_EntranceHallway:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον προηγούμενο σε αυτήν την ακολουθία (RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_DiningRoom), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Entrance_Hallway, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Kitchen:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Kitchen. Παράλληλα, προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Linoleum ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Warm_Toned ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα
- RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_LivingRoom:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον τέταρτο σε αυτήν την ιεραρχία κανόνων (RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_DiningRoom), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Utility:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον έκτο σε αυτήν την ιεραρχία κανόνων (RULE12:_IF_YellowFloors_THEN_MellowMood_FOR_Kitchen), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Utility_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος

SWRL Editor Name:
RULE13:_IF_GreenFloors_THEN_AprilParis_FOR_Glamour

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Curtains(?curtains) ^
                    Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:Upholstery(?upholstery) ^
Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:ArmChair(?armchair) ^
                  Ontology1244033197062:Cushions(?cushions) ^
                   Ontology1244033197062:Carpets(?carpets) ^
                 Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
                   Ontology1244033197062:Floor(?floor) ^
             Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Green_Colors, ?colors) →
  Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:April_in_Paris)
   ^ Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Glamorous) ^
   Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Lime) ^
  Ontology1244033197062:has_as_Colors(?upholstery, Ontology1244033197062:Lime) ^
 Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Delphinium)
  ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Warm_White)
^ Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Warm_White)
 ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                  Ontology1244033197062:Stylized_Leaf_Design)
  ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille)
                  ^ Ontology1244033197062:has_as_Pattern(?sofa,
                  Ontology1244033197062:Woven_in_tones_of_Lime)
^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Chenille)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                 Ontology1244033197062:Woven_with_Circle_Weave)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Striped_Luscious_Toned)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Woven_Luscious_Toned)
  ^ Ontology1244033197062:has_as_Material(?floor, Ontology1244033197062:Onyx) ^
 Ontology1244033197062:has_as_Pattern(?floor, Ontology1244033197062:Green_Onyx)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα “has_as_Material” για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και “has_as_Colors” για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε πράσινο χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Green_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ’ ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του April_In_Paris ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Glamorous ως προσωπικότητας του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Lime ως χρώμα για τις κουρτίνες και την ταπετσαρία, του Delphinium για τα μαξιλάρια, και του Warm_White για τους τοίχους και τα έπιπλα
- `has_as_Material`, για τον προσδιορισμό του Cotton ως το υλικό για τις κουρτίνες και τα μαξιλάρια, του Chenille για τους καναπέδες και τις πολυθρόνες, και του Onyx για το πάτωμα
- `has_as_Pattern`, για τον προσδιορισμό του Stylized_Leaf_Design ως σχέδιο για τις κουρτίνες, του Woven_in_tones_of_Lime για τους καναπέδες, των Striped_Luscious_Toned και Woven_Luscious_Toned για τα μαξιλάρια, και του Green_Onyx για το πάτωμα

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE13:_IF_GreenFloors_THEN_AprilParis_FOR_Kitchen:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στην προσθήκη υλικού, σχεδίου και χρώματος για χαλιά, ενώ παράλληλα αφαιρούνται οι εξαρτήσεις που αναφέρονται στο πάτωμα. Προσθέτοντας το property atom has_as_Material, προσδιορίζουμε το Rubber ως υλικό για τα χαλιά, μέσω ενός καινούργιου property atom has_as_Pattern προσδιορίζουμε το Vibrant_Textural_Circled ως σχέδιο για τα χαλιά, και μέσω ενός ακόμα has_as_Colors καθορίζουμε το Lime ως χρώμα για τα χαλιά. Όπως προαναφέραμε ήδη, διαγράφουμε τα class και property atoms που αφορούν το πάτωμα και μέσω του property atom has_as_Functionality τονίζουμε ότι το δωμάτιο αφορά λειτουργικότητα Kitchen

Εικόνα 24. Συμπερασματολογία βάση πράσινων πατωμάτων

- `RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Linoleum_FOR_Children Bedroom:`
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Children_Bedroom που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα RULE13:_IF_GreenFloors_THEN_AprilParis_FOR_Kitchen. Οι μόνες διαφορές τους εντοπίζονται στους προσδιορισμούς του υλικού Linoleum και του σχεδίου Natural_Robust για τα χαλιά
- `RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Linoleum_FOR_Playroom :`
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον αμέσως προηγούμενο κανόνα (RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Linoleum_FOR_ChildrenBedroom), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Children_Bedroom, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος

- RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_TrellisMotif:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση που δεν διαθέτει λειτουργικότητα (άρα αφαιρείται το property atom has_as_Functionality), και ακολουθεί τα βήματα υλοποίησης του κανόνα RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Linoleum_FOR_Playroom. Οι περαιτέρω διαφορές τους εντοπίζονται στους προσδιορισμούς του υλικού Wool και του σχεδίου Natural_Looking_Loop_in_Trellis_Motif για τα χαλιά
- RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Vinyl_FOR_ChildrenBedroom:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Children_Bedroom που μπορεί να έχει ένα δωμάτιο, και ακολουθεί τα βήματα υλοποίησης του RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Linoleum_FOR_ChildrenBedroom, ενώ οι μόνες διαφορές τους, εντοπίζονται στους προσδιορισμούς του υλικού Vinyl και του σχεδίου Yellow_Green_Toned για τα χαλιά
- RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Vinyl_FOR_Playroom:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον αμέσως προηγούμενο κανόνα (RULE13:_IF_GreenFloors_THEN_AprilParis_WITH_Vinyl_FOR_ChildrenBedroom), με τη μόνη διαφορά να εντοπίζεται στο γεγονός ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Playroom, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος

SWRL Editor Name:
RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_Bedroom

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:Curtains(?curtains) ^
                Ontology1244033197062:Cushions(?cushions) ^
              Ontology1244033197062:Upholstery(?upholstery) ^
 Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:ArmChair(?armchair) ^
                 Ontology1244033197062:Carpets(?carpets) ^
               Ontology1244033197062:Furniture(?furniture) ^
 Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
                  Ontology1244033197062:Floor(?floor) ^
           Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
 Ontology1244033197062:has_as_Colors(Ontology1244033197062:Blue_Colors, ?colors) →
    Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Blue_Rose) ^
     Ontology1244033197062:has_a_Mood(?room, Ontology1244033197062:Relaxing) ^
 Ontology1244033197062:has_as_Functionality(?room, Ontology1244033197062:Bedroom)
    ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Ecru) ^
Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Ivory_White)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Sailor_Blue)
^ Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Cornflower)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Sailor_Blue)
^ Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Cornflower)
                ^ Ontology1244033197062:has_as_Colors(?upholstery,
                        Ontology1244033197062:Sailor_Blue)
                ^ Ontology1244033197062:has_as_Colors(?upholstery,
                        Ontology1244033197062:Cornflower)
  ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Linen)
  ^ Ontology1244033197062:has_as_Material(?curtains, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?curtains,
                    Ontology1244033197062:Blue_Rose_Print)
  ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Chenille)
  ^ Ontology1244033197062:has_as_Pattern(?sofa, Ontology1244033197062:Plush_Navy)
  ^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Wool)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                    Ontology1244033197062:Woven_Neutral_Plaid)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                      Ontology1244033197062:Ivory_White_Silk)
  ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Cotton)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                    Ontology1244033197062:Blue_and_White_Checked)
  ^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Wool) ^
                  Ontology1244033197062:has_as_Pattern(?carpets,
                Ontology1244033197062:Navy_Diamond_Motif_Axminster)
```

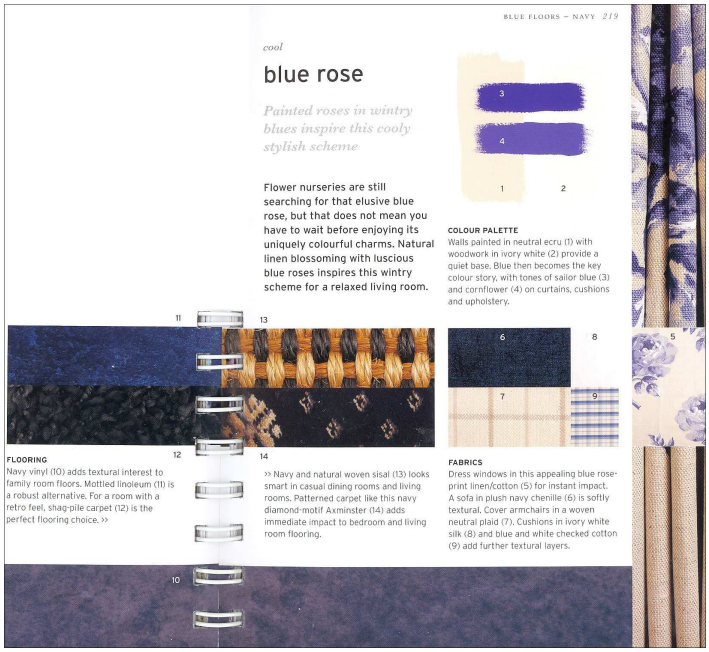
Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα "has_as_Material" για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε μπλε χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Blue_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Blue_Rose ως στυλ δωματίου
- `has_a_Mood`, για τον προσδιορισμό της Relaxing ως προσωπικότητας του δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Bedroom ως λειτουργική ικανότητα του δωματίου
- `has_as_Colors`, για τον προσδιορισμό του Ecru ως χρώμα για τοίχους, του Ivory_White για τα έπιπλα, και των Sailor_Blue και Cornflower για τα μαξιλάρια και την ταπετσαρία
- `has_as_Material`, για τον προσδιορισμό του Linen και Cotton ως υλικά για κουρτίνες, του Chenille για τους καναπέδες, των Silk και Cotton για τα μαξιλάρια, και του Wool για τις πολυθρόνες και τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Blue_Rose_Print ως σχέδιο για τις κουρτίνες, του Plush_Navy για τους καναπέδες, του Navy_Diamond_Motif_Axminster για τα χαλιά, των Ivory_White_Silk και Blue_and_White_Checked για τα μαξιλάρια, και Woven_Neutral_Plaid για τις πολυθρόνες

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_DiningRoom:`
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Dining_Room που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα του παραδείγματος. Η μόνη διαφορά εντοπίζεται στον προσδιορισμό του σχεδίου Navy_and_Natural_Woven για τα χαλιά



Εικόνα 25. Συμπερασματολογία βάση μπλε πατωμάτων

- RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_FamilyRoom:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Family_Room που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα του παραδείγματος. Οι μόνες διαφορές εντοπίζονται στον προσδιορισμό του υλικού Vinyl και του σχεδίου Navy για τα χαλιά
- RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_LivingRoom1:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Living_Room που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα του παραδείγματος. Η μόνη διαφορά, εντοπίζεται στον προσδιορισμό του σχεδίου Navy_and_Natural_Woven για τα χαλιά
- RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_LivingRoom2:
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον κανόνα του παραδείγματος μας (RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_Bedroom), με μοναδική διαφορά ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_Retro:
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Family_Room που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα (RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_Bedroom). Οι μόνες διαφορές τους εντοπίζονται στους προσδιορισμούς του υλικού Shag για τα χαλιά, του σχεδίου Shag_Pile για τα χαλιά, και της προσωπικότητας του δωματίου σε Retro_1960
- RULE14:_IF_BlueFloors_THEN_BlueRose_FOR_Robust:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού και σχεδίου για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Family_Room, και η προσωπικότητα του δωματίου ορίζεται η Robust. Επίσης, προσθέτοντας δύο καινούργια properties atoms όπως το has_as_Material, προσδιορίζουμε το Linoleum ως υλικό για το πάτωμα, και μέσω του has_as_Pattern, προσδιορίζουμε το Mottled ως σχέδιο για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα

SWRL Editor Name:
RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_Bedroom

SWRL Expression:

```
Ontology1244033197062:Room(?room) ^ Ontology1244033197062:ArmChair(?armchair) ^
Ontology1244033197062:Sofa(?sofa) ^ Ontology1244033197062:Curtains(?curtains) ^
Ontology1244033197062:Cushions(?cushions) ^ Ontology1244033197062:Wall(?wall) ^
                Ontology1244033197062:Furniture(?furniture) ^
Ontology1244033197062:has_as_Material(?furniture, Ontology1244033197062:Wood) ^
Ontology1244033197062:Carpets(?carpets) ^ Ontology1244033197062:Floor(?floor) ^
            Ontology1244033197062:has_as_Colors(?floor, ?colors) ^
Ontology1244033197062:has_as_Colors(Ontology1244033197062:Purple_Colors, ?colors) →
Ontology1244033197062:has_a_Style(?room, Ontology1244033197062:Pansy_Panache) ^
Ontology1244033197062:has_as_Functionality(?room, Ontology1244033197062:Bedroom)
   ^ Ontology1244033197062:has_as_Colors(?wall, Ontology1244033197062:Ecru) ^
 Ontology1244033197062:has_as_Colors(?furniture, Ontology1244033197062:Ecru) ^
Ontology1244033197062:has_as_Colors(?curtains, Ontology1244033197062:Thistle) ^
Ontology1244033197062:has_as_Pattern(?curtains, Ontology1244033197062:Circle_Motif)
^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Nugget_Gold)
 ^ Ontology1244033197062:has_as_Material(?sofa, Ontology1244033197062:Thistle) ^
  Ontology1244033197062:has_as_Pattern(?sofa, Ontology1244033197062:Pansies) ^
Ontology1244033197062:has_as_Colors(?cushions, Ontology1244033197062:Nugget_Gold)
 ^ Ontology1244033197062:has_as_Material(?cushions, Ontology1244033197062:Silk)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Nugget_Gold_Patterned)
                ^ Ontology1244033197062:has_as_Colors(?cushions,
                    Ontology1244033197062:Antique_Gold_Color)
                ^ Ontology1244033197062:has_as_Pattern(?cushions,
                  Ontology1244033197062:Antique_Gold_Patterned)
^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Cotton)
^ Ontology1244033197062:has_as_Material(?armchair, Ontology1244033197062:Velvet)
^ Ontology1244033197062:has_as_Colors(?armchair, Ontology1244033197062:Nugget_Gold)
                ^ Ontology1244033197062:has_as_Pattern(?armchair,
                  Ontology1244033197062:Distressed_Nugget_Gold)
^ Ontology1244033197062:has_as_Colors(?carpets, Ontology1244033197062:Lavender)
^ Ontology1244033197062:has_as_Material(?carpets, Ontology1244033197062:Wool) ^
                Ontology1244033197062:has_as_Pattern(?carpets,
                    Ontology1244033197062:Textural_Twist)
```

Επεξήγηση και Αποτέλεσμα:
Το antecedent μέρος του κανόνα περιλαμβάνει τα class atoms που αντιπροσωπεύουν τις οντότητες στις οποίες χρειάζεται να εφαρμοστούν οι ζητούμενες συσχετίσεις στιγμιότυπων ανάμεσα στις κλάσεις. Επίσης, υπάρχουν και properties atoms όπως τα "has_as_Material" για την επιλογή επίπλων που είναι αποκλειστικά κατασκευασμένα από ξύλο, και "has_as_Colors" για την επιλογή ενός πατώματος που έχει ένα οποιοδήποτε μωβ χρώμα (το οποίο περιέχεται στην κατηγορία χρωμάτων Purple_Colors). Το consequent μέρος του κανόνα θα εκτελεστεί εφόσον στην οντολογία εντοπιστεί τουλάχιστον ένα άτομο από κάθε class atom που αναφέρεται στο antecedent μέρος, και ταυτόχρονα εκπληρώνει οποιεσδήποτε property atom συνθήκες υπάρχουν σε αυτό το μέρος. Όταν επαληθευτεί εξ' ολοκλήρου το antecedent μέρος, ξεκινάει η συμπερασματολογία τιμών για τα πεδία των κατάλληλων ιδιοτήτων συσχετίζοντας τις κλάσεις με property atoms. Τα property atoms που λαμβάνουν μέρος στο συγκεκριμένο κανόνα είναι:

- `has_a_Style`, για τον προσδιορισμό του Pansy_Panache ως στυλ δωματίου
- `has_as_Functionality`, για τον προσδιορισμό της Bedroom ως λειτουργική ικανότητα του δωματίου

- `has_as_Colors`, για τον προσδιορισμό του Ecru ως χρώμα για τους τοίχους και τα έπιπλα, του Thistle για τις κουρτίνες και τους καναπέδες, με τους τελευταίους να χρωματίζονται και με Nugget_Gold, χρώμα που αποδίδεται στις πολυθρόνες και στα μαξιλάρια. Ωστόσο τα μαξιλάρια μπορούν να έχουν και Antique_Gold_Color, ενώ τα χαλιά προσδιορίζονται με το χρώμα Lavender
- `has_as_Material`, για τον προσδιορισμό του Silk ως υλικό για τις κουρτίνες, των Cotton και Velvet για τις πολυθρόνες, και του Wool για τα χαλιά
- `has_as_Pattern`, για τον προσδιορισμό του Circle_Motif ως σχέδιο για τις κουρτίνες, του Pansies για τους καναπέδες, του Distressed_Nugget_Gold για τις πολυθρόνες, του Nugget_Gold_Patterned για τα μαξιλάρια, και του Textural_Twist για τα χαλιά

Τα axioms που προκύπτουν από την συμπερασματολογία των κανόνων μπορούν να εισαχθούν ως νέα γνώση πίσω στην OWL οντολογία μέσω του κουμπιού Jess → OWL του SWRLJessTab.

Παρεμφερείς κανόνες:

- `RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_Kitchen:`
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού, σχεδίου, και χρώματος για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Kitchen. Επίσης, προσθέτοντας τρία καινούργια properties atoms όπως το has_as_Material προσδιορίζουμε το Ceramic_Tile ως υλικό για το πάτωμα, μέσω του has_as_Pattern προσδιορίζουμε το Deep_Lavender ως σχέδιο για το πάτωμα, και μέσω του has_as_Colors προσδιορίζουμε το Lavender ως χρώμα για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα

Εικόνα 26. Συμπερασματολογία βάση μωβ πατωμάτων

- `RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_LivingRoom:`
  Ο συγκεκριμένος κανόνας είναι πανομοιότυπος με τον κανόνα του παραδείγματος μας (RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_Bedroom), με μοναδική διαφορά ότι το property atom has_as_Functionality προσδιορίζει την λειτουργικότητα του δωματίου ως Living_Room, κρατώντας όμοιο το antecedent και το υπόλοιπο consequent μέρος
- `RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_Playroom_WITH_Linoleum:`
  Ο συγκεκριμένος κανόνας αποτελεί μια εναλλακτική πρόταση βασισμένη στην λειτουργικότητα Playroom που μπορεί να έχει ένα δωμάτιο και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα του παραδείγματος. Οι μόνες διαφορές εντοπίζονται στον προσδιορισμό του υλικού Linoleum και του σχεδίου Mottled_Paint_Effect για τα χαλιά

- RULE15:_IF_PurpleFloors_THEN_PansyPanache_FOR_Playroom_WITH_Rubber:
  Η διαφοροποίηση του συγκεκριμένου κανόνα με αυτόν του παραδείγματος εντοπίζεται στον προσδιορισμό ενός υλικού, σχεδίου, και χρώματος για κάποιο πάτωμα και στην μη ύπαρξη χαλιών για την ανάδειξη αυτού του τύπου πατώματος. Επίσης, τονίζεται ότι η λειτουργικότητα του δωματίου αυτού του κανόνα είναι η Playroom. Επίσης, προσθέτοντας τρία καινούργια properties atoms όπως το has_as_Material προσδιορίζουμε το Rubber ως υλικό για το πάτωμα, μέσω του has_as_Pattern προσδιορίζουμε το Ribbed ως σχέδιο για το πάτωμα, και μέσω του has_as_Colors προσδιορίζουμε το Lavender ως χρώμα για το πάτωμα. Επίσης αφαιρούμε class και property atoms που προσδιορίζουν χαλιά, και φυσικά προσθέτουμε στο antecedent μέρος τα απαραίτητα class atoms για το πάτωμα
- RULE15:_IF_PurpleFloors_THEN_PansyPanache_WITH_Trellis:
  Ο συγκεκριμένος κανόνας αποτελεί μια πρόταση που δεν διαθέτει καμία λειτουργικότητα δωματίου και ακολουθεί παρόμοια βήματα υλοποίησης με τον κανόνα του παραδείγματος. Οι μόνες διαφορές εντοπίζονται στον προσδιορισμό του σχεδίου Natural_Effect_Trellis_Motif για τα χαλιά, και φυσικά στην διαγραφή του property atom has_as_Functionality για την αφαίρεση της προκαθορισμένης λειτουργικότητας που προτείνεται

6. Αυτόματη Μετατροπή Εικονικού Χώρου με τη χρήση XSLT

Η XSLT (Extensible Stylesheet Language Transformations) είναι μια γλώσσα βασισμένη στο πρότυπο XML, που χρησιμοποιείται για το μετασχηματισμό των εγγράφων XML σε άλλα έγγραφα XML. Το αρχικό έγγραφο δεν αλλάζει αλλά δημιουργείται ένα καινούργιο έγγραφο βασισμένο στο περιεχόμενο ενός υπάρχοντος. Το πρότυπο επεξεργασίας XSLT περιλαμβάνει:

- ένα ή περισσότερα πηγαία XML έγγραφα
- ένα ή περισσότερα XSLT stylesheet modules
- η XSLT template μηχανή επεξεργασίας (ο επεξεργαστής)
- ένα η περισσότερα έγγραφα αποτελέσματος

Ο επεξεργαστής XSLT παίρνει συνήθως δύο έγγραφα εισόδου – ένα πηγαίο έγγραφο XML και ένα XSLT stylesheet – και παράγει ένα έγγραφο αποτελέσματος. Το XSLT Stylesheet περιέχει μια συλλογή από template κανόνες, λειτουργίες και οδηγίες που καθοδηγούν τον επεξεργαστή στην παραγωγή του τελικού εγγράφου.

Η γλώσσα XSLT είναι δηλωτική, αντί να απαριθμήσει μια επιτακτική ακολουθία ενεργειών που πρέπει να εκτελεστούν σε ένα περιβάλλον συνθηκών, οι κανόνες των template καθορίζουν πώς να χειριστούν ένα κόμβο που ταιριάζει σε ένα συγκεκριμένο XPath pattern, μόνο στην περίπτωση που ο επεξεργαστής τύχει και συναντήσει ένα τέτοιο κόμβο. Παράλληλα, τα περιεχόμενα των αντίστοιχων templates αποτελούνται από αποτελεσματικές λειτουργικές εκφράσεις που αντιπροσωπεύουν άμεσα την αξιολογημένη μορφή τους: το δέντρο αποτελέσματος, το οποίο είναι η βάση της παραγωγής του επεξεργαστή.

Ο επεξεργαστής ακολουθεί ένα καθορισμένο αλγόριθμο: υποθέτοντας ότι έχουμε ένα stylesheet το οποίο έχει ήδη διαβαστεί και προετοιμαστεί, ο επεξεργαστής χτίζει ένα πηγαίο δέντρο από το εισάγων XML έγγραφο. Μετά αρχίζει με την επεξεργασία του root κόμβου του πηγαίου δέντρου, ψάχνοντας μέσα στο stylesheet για το καλύτερο δυνατό ταίριασμα template για αυτόν τον κόμβο, και αξιολογεί τα περιεχόμενα του template αυτού. Οι οδηγίες σε κάθε template – σε γενικές γραμμές - καθοδηγούν τον επεξεργαστή είτε στην δημιουργία κόμβων στο δέντρο αποτελέσματος, είτε στην επεξεργασία περισσότερων κόμβων στο πηγαίο δέντρο με τον ίδιο τρόπο όπως στον κόμβο ρίζας. Εν τέλει, η παραγωγή προέρχεται από το δέντρο αποτελέσματος.

6.1 SVG to X3D Transformation

Ακολουθεί αναλυτική περιγραφή της λειτουργίας του XSLT, θεωρώντας δεδομένο ότι έχουμε τροφοδοτήσει ως πηγαίο XML έγγραφο ένα SVG, που αναπαριστά έναν εικονικό δυσδιάστατο χώρο. Πρωτίστως διευκρινίζουμε ότι πρόκειται για έγγραφο βασισμένο στο πρότυπο XML και καθορίζουμε την έκδοσή του. Έπειτα δημιουργούμε το root element <xsl:transform>, το οποίο δηλώνει ότι θα πραγματοποιήσουμε ένα XSL transformation, συμπεριλαμβάνοντας τα παρακάτω namespaces:

- `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"`, παρέχει πρόσβαση στα XSLT elements, attributes και λοιπά χαρακτηριστικά που αναφέρονται στο επίσημο W3C XSLT namespace. Αναγκαστικά πρέπει να ακολουθεί και ένα attribute της έκδοσης
- `xmlns:xalan="http://xml.apache.org/xalan"`, παρέχει υποστήριξη για τον επεξεργαστή μετατροπής XML εγγράφων που επιλέξαμε
- `xmlns:str="http://exslt.org/strings"`, παρέχει υποστήριξη για τις λειτουργίες που έχουν να κάνουν με το χειρισμό string
- `xmlns:xml="http://www.w3.org/XML/1998/namespace"`, παρέχει τα attribute xml:lang, xml:space, xml:base και xml:id

- `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`, διευκρινίζει ότι elements και data types που εμπεριέχονται στην συγκεκριμένη ηλεκτρονική διεύθυνση θα έχουν ως πρόθεμα το xsi:
- `xmlns:xlink="http://www.w3.org/1999/xlink"`, παρέχει XLink elements και attributes για την δημιουργία και περιγραφή συνδέσμων μεταξύ resources
- `xmlns:math="http://exslt.org/math"`, παρέχει υποστήριξη για τις λειτουργίες που έχουν να κάνουν με μαθηματικές συναρτήσεις

Αφού τελειώσουμε με τις δηλώσεις όλων των απαραίτητων namespaces ακολουθεί το attribute exclude-result-prefixes="#all", με την τιμή "#all" να αποτρέπει στα προθέματα των namespaces να εμφανίζονται στο result-tree. Ορίζουμε απευθείας παιδί της ρίζας το <xsl:output> element με attributes που αντίστοιχα καθορίζουν ότι το εξάγων έγγραφο θα είναι τύπου XML, την έκδοσή του, την σειρά κωδικοποίησής του και καταχωρώντας την τιμή "yes" στο indent attribute να επιτρέπουμε στον XSLT processor να προσθέτει επιπλέον κενά και αλλαγές γραμμής όταν εξάγει το result-tree. Αξίζει να σημειωθεί ότι αυτό γίνεται για λόγους οπτικοποίησης του αποτελέσματος ώστε να είναι ευκολότερη η ανάπτυξη και η αποσφαλμάτωση του κώδικα. Όμως, επειδή κάθε XSLT processor προσθέτει διαφορετικό αριθμό κενών/γραμμών και παράλληλα ποτέ δεν είναι γνωστός ο αριθμός αυτών, είναι προτιμότερο κατά τη διάρκεια της παραγωγής η τιμή του `indent` attribute να είναι ίση με "no".

Μετά το <xsl:output> element, δηλώνουμε έναν αριθμό από υψηλού επιπέδου μεταβλητές που είναι επίσης παιδιά της ρίζας, αλλά σε σχέση με τις τοπικές μεταβλητές, οι τιμές τους δε μπορούν να αλλάξουν κατά την διάρκεια εκτέλεσης του μετασχηματισμού:

- `scaling`, καθορίζει τον αριθμό των pixels που αντιστοιχούν σε 1 μέτρο του X3D κόσμου. Ισούται με την τιμή του attribute "WorldScale" που βρίσκεται μέσα στο metadata element που φέρει την τιμή "world_info" στο id attribute
- `wallHeight`, καθορίζει το ύψος του X3D κόσμου. Ισούται με την τιμή του attribute "RoomHeight" που βρίσκεται μέσα στο metadata element που φέρει την τιμή "world_info" στο id attribute
- `strokeWidth`, δίνει μια προκαθορισμένη τιμή στο πάχος της εξωτερικής γραμμής των αντικειμένων

Ακολούθως, χρησιμοποιώντας το <xsl:template match="/"> element, αντιστοιχούμε το παρών template με την ρίζα του πηγαίου SVG αρχείου, που είναι το SVG element. Με αυτόν τον τρόπο, γίνεται δυνατόν να αντλήσουμε οποιαδήποτε πληροφορία περιέχεται στη δομή του πηγαίου SVG δέντρου. Εφόσον η έξοδος του μετασχηματισμού θα είναι ένα τυποποιημένο X3D δέντρο πρώτο στοιχείο θα είναι το X3D element. Το profile attribute αυτού του element, επισημαίνει ότι πρόκειται για Immersive profile, δίνοντας την δυνατότητα για πλήρη υποστήριξη 3D γραφικών, αλληλεπίδραση χρήστη και scripting. Έπειτα δημιουργείται το Scene element, που εμπεριέχει τον 3D κόσμο που θα στηθεί έχοντας ως πρώτο παιδί ένα τυπικό background element. Μετά αρχίζει η σταδιακή ανάγνωση του πηγαίου SVG αρχείου από την ρίζα του, αξιοποιώντας την function <xsl:for-each select="//*[name()='svg']">.

Για την περίπτωση των elements του πηγαίου SVG αρχείου που έχουν πατέρα ένα g element, δηλώνουμε την function <xsl:for-each select="./child::*[name()='g']"> ως παιδί της προαναφερθέντος “for-each” function. Χρησιμοποιώντας συνεχόμενες substring functions διαχωρίζουμε και ανακτούμε κάθε φορά τις επιθυμητές τιμές από τα translate και rotate πεδία του SVG αρχείου και τις αναθέτουμε στις εξής μεταβλητές:

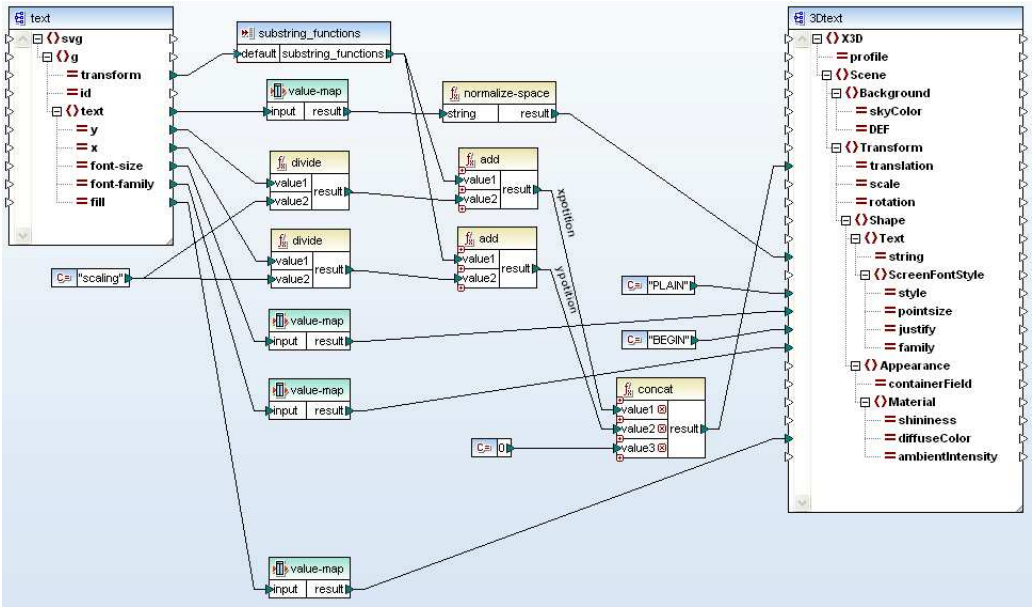
- `transformX1`, η τιμή του πρώτου πεδίου του translate του transform attribute διαιρεμένη με την μεταβλητή scaling
- `transformY1`, η τιμή του δεύτερου πεδίου του translate του transform attribute διαιρεμένη με την μεταβλητή scaling
- `rotation1`, η τιμή του πεδίου του rotate του transform attribute

Αυτές οι μεταβλητές είναι τοπικές επειδή δηλώνονται μέσα σε ένα βρόχο, με αποτέλεσμα οι τιμές τους να ενημερώνονται σε κάθε επανάληψη. Αντίθετα, για τις περιπτώσεις των elements του πηγαίου SVG αρχείου που δεν έχουν ως πατέρα κάποιο g element, όλες οι παραπάνω μεταβλητές θα έχουν τιμή ίση με μηδέν. Παρόλα αυτά, ανεξάρτητα σε ποια περίπτωση ανήκει το κάθε element, η ακόλουθη διαδικασία του μετασχηματισμού παραμένει η ίδια. Δηλώνουμε ένα πλήθος από “for-each” βρόχους,

δύο φορές τον καθέναν τους, εφόσον οφείλουμε να ψάχνουμε για elements εντός και εκτός του g. Ο αριθμός και η διαφοροποίηση αυτών των βρόχων εξαρτάται αποκλειστικά από τα SVG elements που μπορούμε να αναπαραστήσουμε στο X3D:

- `<xsl:for-each select="./child::*[name()='text']">`

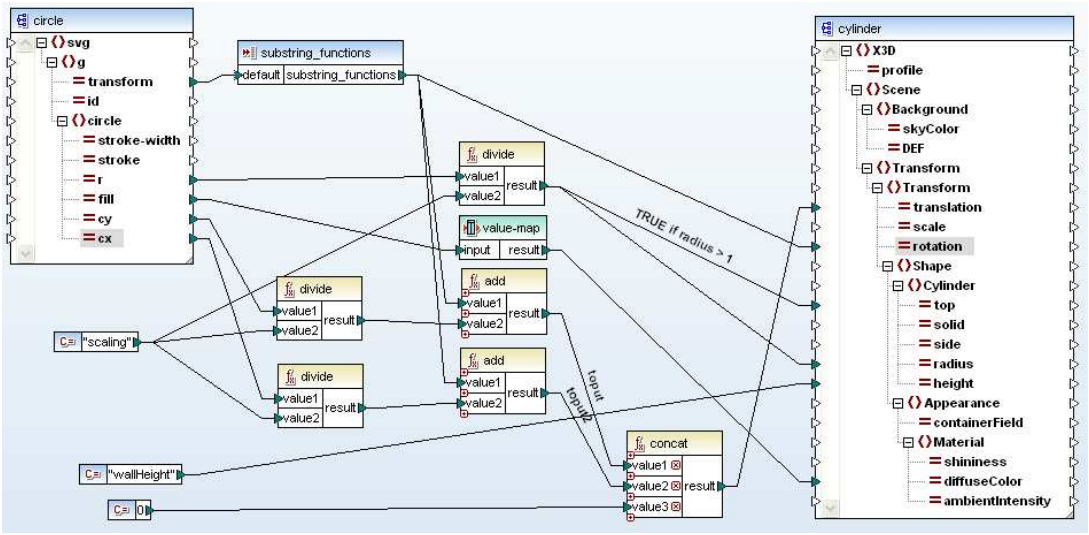
Για κάθε text element, δημιουργούμε ένα Transform element με τα κατάλληλα translation, rotation και scale attributes. Μετά δημιουργούμε ένα Shape element ως παιδί του Transform, που έχει τα εξής δύο παιδιά:
Ένα Text element με string attribute που περιέχει την τιμή του SVG text element και ένα παιδί του, που περιέχει πληροφορίες σχετικά με την γραμματοσειρά (τύπος, μέγεθος, στυλ, στοίχιση).
Ένα Appearance element και το παιδί του, που περιέχει πληροφορίες που αφορούν την οπτική εμφάνιση του Text element, όπως την ένταση φωτός, την λάμψη και το χρώμα (αντιπροσωπεύεται από το fill attribute του text element του SVG).



Εικόνα 27. Γραφική αναπαράσταση της διαδικασίας μετασχηματισμού του text element από το πηγαίο SVG στο 3DText element του παραγόμενου X3D

- `<xsl:for-each select="./child::*[name()='circle']">`

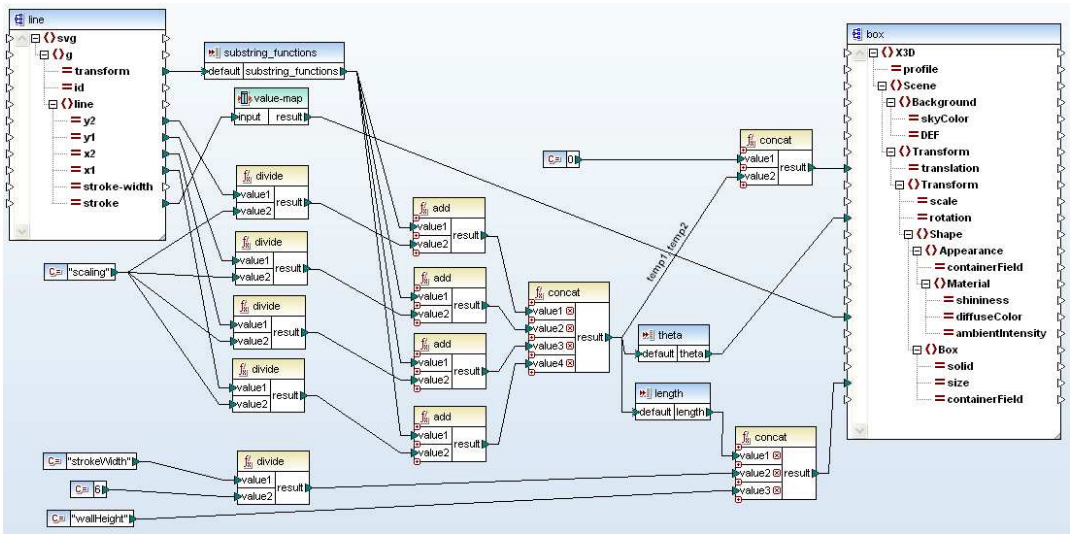
Για κάθε circle element, δημιουργούμε ένα φωλιασμένο Transform element σε ένα άλλο Transform, το οποίο περιέχει τα κατάλληλα translation, rotation και scale attributes. Μετά δημιουργούμε ένα Shape element ως παιδί του φωλιασμένου Transform, που έχει τα εξής δύο παιδιά:
Ένα Cylinder element με attributes που αναπαριστούν το ύψος και την ακτίνα του X3D κυλίνδρου, αφού έχουν υποστεί διαίρεση με την μεταβλητή scaling. Επίσης παίρνουμε ως δεδομένο, ότι αν η ακτίνα είναι μικρότερη από 1m, τότε η τιμή του top attribute θα είναι TRUE, δίνοντας την αίσθηση ενός συμπαγή κυλίνδρου.
Ένα Appearance element και το παιδί του, που περιέχει πληροφορίες που αφορούν την οπτική εμφάνιση του Cylinder element, όπως την ένταση του φωτός, την λάμψη και το χρώμα.



Εικόνα 28. Γραφική αναπαράσταση της διαδικασίας μετασχηματισμού του circle element από το πηγαίο SVG στο Cylinder element του παραγόμενου X3D

- `<xsl:for-each select="./child::*[name()='line']">`

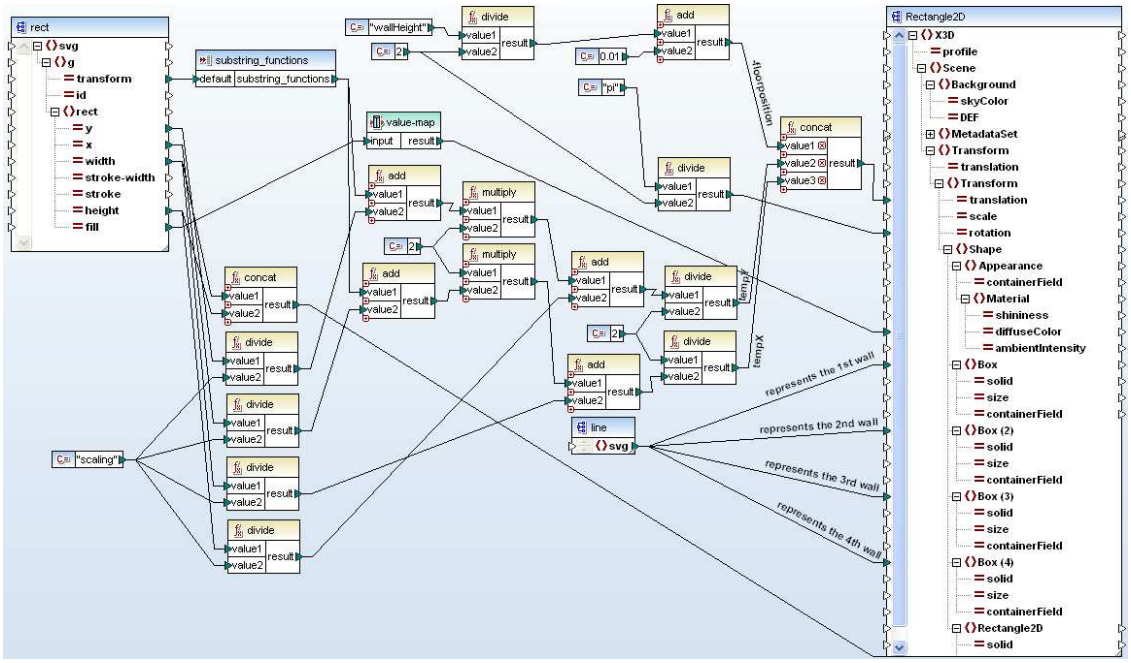
Για κάθε line element, δημιουργούμε ένα Transform element με μοναδικό attribute το translation. Μετά ένα επιπλέον Transform φωλιασμένο στο παραπάνω, περιέχει τα rotation και scale attributes. Ακολουθεί ένα Shape element ως παιδί του τελευταίου Transform, που έχει τα εξής δύο παιδιά:
Ένα Box element με τις τιμές του size attribute του να αναπαριστούν τους x,y και z άξονες αντίστοιχα. Οι τιμές αυτές έχουν συμπληρωθεί μέσω διανυσματικών μαθηματικών πράξεων βάση των παρεχόμενων attributes του line element του SVG.
Ένα Appearance element και το παιδί του, που περιέχει πληροφορίες που αφορούν την οπτική εμφάνιση του Box element, όπως την ένταση του φωτός, την λάμψη και το χρώμα (αντιπροσωπεύεται από το stroke attribute του line element του SVG).



Εικόνα 29. Γραφική αναπαράσταση της διαδικασίας μετασχηματισμού του line element από το πηγαίο SVG στο Box element του παραγόμενου X3D

- `<xsl:for-each select="./child::*[name()='rect']">`

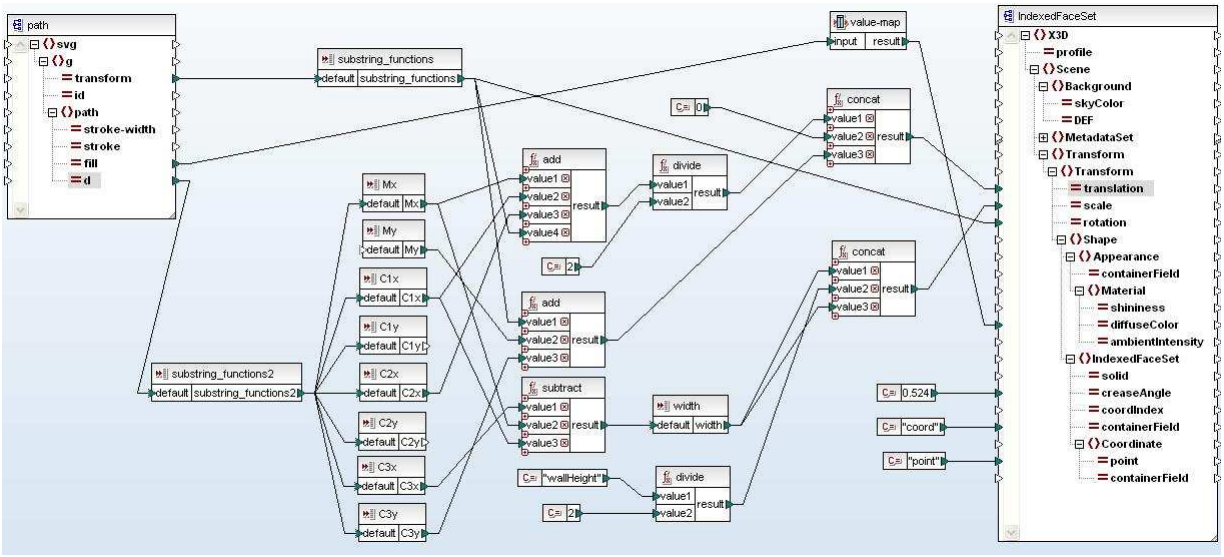
Για κάθε rectangle element, δημιουργούμε πέντε Transform elements. Το πρώτο από αυτά έχει απευθείας παιδί ένα επιπλέον Transform element που περιέχει τα κατάλληλα translation, rotation και scale attributes. Ακολουθεί ένα Shape element ως παιδί του ενθυλακωμένου Transform, που έχει τα εξής δύο παιδιά:
Ένα Rectangle2D element με τις τιμές του size attribute να αναπαριστούν τους x,y,z άξονες αντίστοιχα. Οι τιμές αυτές έχουν συμπληρωθεί μέσω διανυσματικών μαθηματικών πράξεων βάση των παρεχόμενων attributes του rectangle element του SVG.
Ένα Appearance element και το παιδί του, που περιέχει πληροφορίες που αφορούν την οπτική εμφάνιση του Rectangle2D element, όπως την ένταση του φωτός, την λάμψη και το χρώμα.
Τα εναπομείναντα τέσσερα Transform elements, υλοποιούνται σαν να ήταν line elements, βασισμένα στην λογική ότι κάθε Transform element αντιπροσωπεύει μια διαφορετική πλευρά του τετραγώνου.



Εικόνα 30. Γραφική αναπαράσταση της διαδικασίας μετασχηματισμού του rect element από το πηγαίο SVG στο Rectangle2D element του παραγόμενου X3D

- `<xsl:for-each select="./child::*[name()='path']">`

Για κάθε path element, δημιουργούμε ένα φωλιασμένο Transform element σε ένα άλλο Transform, το οποίο περιέχει τα κατάλληλα translation, rotation και scale attributes. Μετά δημιουργούμε ένα Shape element ως παιδί του φωλιασμένου Transform, που έχει τα εξής δύο παιδιά:
Ένα IndexedFaceSet element με τιμές στο coordIndex attribute. Μέσα στο element αυτό υπάρχει ο κόμβος Coordinate που καθορίζει τα τρισδιάστατα vertices που αναφέρονται από το coordIndex attribute του γονικού κόμβου. Το IndexedFaceSet χρησιμοποιεί τους δείκτες στον coordIndex πεδίο για να διευκρινίσει τα πολυγωνικά πρόσωπα κάνοντας indexing στις συντεταγμένες του Coordinate κόμβου.
Ένα Appearance element και το παιδί του, που περιέχει πληροφορίες που αφορούν τη οπτική εμφάνιση του IndexedFaceSet element, όπως την ένταση του φωτός, την λάμψη και το χρώμα.



Εικόνα 31. Γραφική αναπαράσταση της διαδικασίας μετασχηματισμού του path element από το πηγαίο SVG στο IndexedFaceSet element του παραγόμενου X3D

Πριν λάβει τέλος η διαδικασία ανίχνευσης των g elements, καλούμε ένα επιπλέον template να εκτελεστεί. Το συγκεκριμένο template τονίζει την δυνατότητα του πηγαίου SVG αρχείου, να περιέχει ένα συγκεκριμένο είδος g element που έχουν ως παιδιά:

- ένα rectangle element που έχει για string τιμή, το γονικό id attribute ανάλογα με το αντικείμενο που αναπαριστά (κρεβάτι, καρέκλα, πόρτα κτλ). Τα rectangle elements αυτού του τύπου δεν εμφανίζονται στον X3D κόσμο αλλά αντικαθίστανται από το κατάλληλο εξωτερικό X3D αρχείο
- ένα text element που έχει για string τιμή, το γονικό id attribute ανάλογα με τον τύπο του rectangle. Τα text elements αυτού του τύπου δεν εμφανίζονται στον X3D κόσμο, καθώς χρησιμοποιούνται μόνο για λόγους κατανόησης και οπτικού περιεχομένου στο SVG
- και ένα metadata element, που περιέχει μια XML οντολογία σχετική κάθε φορά με τον τύπο του rectangle

Λόγω του γεγονότος ότι οι τοπικές μεταβλητές είναι ορατές και προσπελάσιμες μόνο μέσα στο template το οποίο δηλώθηκαν, επαναλαμβάνουμε τις ίδιες substring functions του πρώτου template, ώστε να διαχωρίσουμε και να ανακτήσουμε τις τιμές από το transform attribute και να τις αναθέσουμε στις μεταβλητές transformX1, transformY1, rotation1. Αυτές οι τιμές μαζί με τα attributes του rectangle element αξιοποιούνται από μαθηματικές συναρτήσεις και δίνουν την ακριβή θέση, βασισμένη στον x και z άξονα, όπου θα τοποθετηθεί το εξωτερικό X3D αρχείο. Ύστερα, ελέγχεται η δομή της XML οντολογίας που περιέχεται στο metadata element.

Αν ανιχνευθεί floor κόμβος (που σημαίνει ότι το αντικείμενο μπορεί να είναι έπιπλο, χαλί κτλ), δημιουργούμε ένα Transform element και καθορίζουμε μια κατάλληλη τιμή στο πεδίο του translation attribute που αναπαριστά τον y άξονα, ώστε το αντικείμενό μας να τοποθετηθεί στο υποθετικό πάτωμα του X3D κόσμου.

```
<xsl:element name="Transform">
      <xsl:attribute name="translation">
            <xsl:value-of select="$tempx"/>              → X axis
            <xsl:text> </xsl:text>
            <xsl:value-of select="-$floorposition"/>     → Y axis
            <xsl:text> </xsl:text>
            <xsl:value-of select="$tempy"/>              → Z axis
      </xsl:attribute>
            …
```

Σχήμα 5. Τοποθέτηση αντικειμένου στο πάτωμα του X3D κόσμου

Αν ανιχνευθεί wall κόμβος (που σημαίνει ότι το αντικείμενο μπορεί να είναι παράθυρο, πίνακας κτλ), δημιουργούμε ένα Transform element και αποδίδουμε διαφορετική τιμή στο πεδίο του translation attribute που αναπαριστά τον y άξονα, με γνώμονα ότι το αντικείμενό μας θα τοποθετηθεί στο κέντρο του X3D κόσμου.

```
<xsl:element name="Transform">
      <xsl:attribute name="translation">
            <xsl:value-of select="$tempx"/>            → X axis
            <xsl:text> </xsl:text>
            <xsl:value-of select="0 "/>                → Y axis
            <xsl:text> </xsl:text>
            <xsl:value-of select="$tempy"/>            → Z axis
      </xsl:attribute>
            …
```

Σχήμα 6. Τοποθέτηση αντικειμένου στο κέντρο του X3D κόσμου

Έπειτα, συνεχίζουμε τον έλεγχο της XML οντολογίας για κάποιον path κόμβο, ο οποίος έχει ως τιμή μια τοπική ή δικτυακή διεύθυνση (URL) που αντιπροσωπεύει το επιθυμητό μοντέλο που θέλουμε να παρουσιάσουμε. Ακολουθεί η δημιουργία ενός Inline element ως παιδί του Transform και αποδίδουμε την τιμή του path κόμβου στο url attribute του. Δίνοντας κατάλληλες τιμές στο translation attribute του Transform element επιτυγχάνουμε την μετακίνηση ολόκληρης της σκηνής (άρα και του μοντέλου) και την τοποθέτησή της στην επιθυμητή θέση μέσα στον X3D κόσμο.

Επειδή είναι αναγκαίος και ο αντίστροφος μετασχηματισμός, δηλαδή η μετατροπή του παραγόμενου X3D δέντρου στο αρχικό SVG δέντρο, ενθυλακώσαμε συγκεκριμένες πληροφορίες του πηγαίου SVG αρχείου χρησιμοποιώντας Metadata Information του X3D specification (MetadataSet, MetadataString, MetadataFloat). Αυτές οι πληροφορίες θα περιέχονται σε ξεχωριστά ομαδοποιημένα MetadataSet βάση του τύπου της πληροφορίας. Κάθε MetadataSet μπορεί να "κρατάει" οποιοδήποτε αριθμό και τύπο από Metadata nodes. Τα MetadataSet που υλοποιούμε στο μετασχηματισμό μας είναι τα ακόλουθα:

- Ένα MetadataSet element, το οποίο είναι παιδί του Scene element και περιέχει δύο MetadataString και δύο MetadataFloat elements. Αυτά τα elements αποθηκεύουν τις τιμές των scaling, ύψους, πλάτους και viewbox attributes που βρίσκονται στο SVG element. Το συγκεκριμένο MetadataSet, το δημιουργούμε πριν οποιοδήποτε άλλο, έχοντας ως απώτερο σκοπό την ομαλή τοποθέτηση και απεικόνιση του SVG κόσμου στην οθόνη μας, όπως ακριβώς ήταν στην αρχική του κατάσταση
- Για κάθε text element που δεν εμφανίζεται στον X3D κόσμο, δημιουργούμε ένα MetadataSet element, που είναι επίσης παιδί του Scene element. Αυτό περιέχει τρία MetadataFloat και τρία MetadataString elements. Τα MetadataFloat αποθηκεύουν τις τιμές των x,y attributes του text element και την τιμή του rotate attribute του γονικού g element. Στη συνέχεια τα MetadataString αποθηκεύουν τιμές σχετικά με την γραμματοσειρά, δηλαδή τα font family, font size attributes και το περιεχόμενο (value) του node. Ο συνδυασμός των παραπάνω, βοηθάει στην επανατοποθέτηση του text element στη αρχική του θέση και στην σωστή απεικόνιση του κειμένου
- Για κάθε rect element που δεν εμφανίζεται στον X3D κόσμο, δημιουργούμε ένα MetadataSet element, αλλά αυτή τη φορά ως απευθείας παιδί του Inline element. Αυτό περιέχει ένα MetadataString, τρία MetadataFloat elements και ένα metadata node (το οποίο δεν αναλύεται από το X3D). Το MetadataString αποθηκεύει την τιμή του id attribute του rect element. Τα MetadataFloat αποθηκεύουν τις τιμές του ύψους και του πλάτους attributes του rect element καθώς και την τιμή του rotate attribute του γονικού g element

Τέλος, εκτελείται ένας αλγόριθμος που καλεί ένα αναδρομικό template, ώστε να ανιχνεύσουμε την XML οντολογία που περιέχεται στο συγκεκριμένο rect element και να την ενσωματώσουμε εξολοκλήρου μέσα στο metadata node.

```
<xsl:for-each
select="./parent::*/descendant::*[name()='metadata']">
      <xsl:element name="{name()}">
            <xsl:for-each select="attribute::*">
                  <xsl:attribute name="{name()}">
                        <xsl:value-of select="normalize-space(.)"/>
                  </xsl:attribute>
            </xsl:for-each>
            <xsl:value-of select="normalize-
            space(.[not(child::*)])"/>
             <xsl:for-each select="./child::*">
                  <xsl:element name="{name()}">
                        <xsl:for-each select="attribute::*">
                              <xsl:attribute name="{name()}">
                                    <xsl:value-of select="normalize-
                                    space(.)"/>
                              </xsl:attribute>
                        </xsl:for-each>
                        <xsl:value-of select="normalize-
                        space(.[not(child::*)])"/>
                        <xsl:call-template
                        name="RecursiveForChildrens"/>
                  </xsl:element>
            </xsl:for-each>
      </xsl:element>
</xsl:for-each>
```

Σχήμα 7. Αλγόριθμος αναζήτησης και δημιουργίας του metadata δέντρου

```
<xsl:template name="RecursiveForChildrens">
      <xsl:for-each select="./child::*">
            <xsl:element name="{name()}">
                  <xsl:for-each select="attribute::*">
                        <xsl:attribute name="{name()}">
                              <xsl:value-of select="normalize-
                              space(.)"/>
                        </xsl:attribute>
                  </xsl:for-each>
                  <xsl:value-of select="normalize-
                  space(.[not(child::*)])"/>
                  <xsl:call-template name="RecursiveForChildrens"/>
            </xsl:element>
      </xsl:for-each>
<xsl:template>
```

Σχήμα 8. Αναδρομικό template για την αναζήτηση και δημιουργία σε βάθος μεγαλύτερο του 2

Τα τρία παραπάνω templates αποτελούν το κυρίως σώμα του XSLT εγγράφου και με το πέρας της εκτέλεσής τους, σηματοδοτούμε και το τέλος της διαδικασίας κλείνοντας το <xsl:transform> element.

Η σελίδα αυτή είναι σκόπιμα κενή

ΒΙΒΛΙΟΓΡΑΦΙΑ

- OWL from Wiki: http://en.wikipedia.org/wiki/Web_Ontology_Language
- OWL Web Ontology Language Guide. W3C Recommendation 10 February 2005: http://www.w3.org/TR/2004/REC-owl-guide-20040210/
- Sean Bechhofer, Ian Horrocks and Peter F. Patel-Schneider. Tutorial on OWL: http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004: http://www.w3.org/Submission/SWRL/
- Grigoris Antoniou and Frank van Harmelen. Chapter 4: Web Ontology Language (OWL): www.ics.forth.gr/isl/swprimer/presentations/Chapter4.ppt
- What is Protege-OWL: http://protege.stanford.edu/overview/protege-owl.html
- Amna Bashart. Semantic Web Rule Language Tutorial (SWRL). FAST-NU (SPRING 2009): http://www.scribd.com/doc/23580395/SWRL-Tutorial-01
- ProtegeWiki:SWRLTab: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab
- ProtegeWiki:SWRLEditor FAQ: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLEditorFAQ
- ProtegeWiki:SWRLTab Built Ins Libraries: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries
- ProtegeWiki:SWRLLanguage FAQ: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ
- ProtegeWiki:SQWRL: http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL
- ProtegeWiki:SWRLJess Bridge: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessBridge
- ProtegeWiki:SWRLJess Tab: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab
- Athanasios G.Malamos, Paraskevi V. Sympa, Georgios S.Mamakis, “Xml Annotation Of Conceptual Characteristics In Interior Decoration”, 6th International Conference, New Horizons in Industry, Business and Education (NHIBE 2009), 27 - 28 August 2009, Santorini
- Suzy Chiazzari, (2005). COLOR YOUR HOME, more than 65,000 at-a-glance room combinations. New York: Collins Design, An Imprint of HarperCollinsPublishers
- Adrienne Chinn, (2007). THE HOME DECORATOR'S COLOUR & TEXTURE SOURCEBOOK, 180 schemes for the home. UK: Apple Press
- XSL Transformations (XSLT) Version 1.0: http://www.w3.org/TR/xslt
- XSLT from Wiki: http://en.wikipedia.org/wiki/XSLT

Η σελίδα αυτή είναι σκόπιμα κενή

(Υπογραφή)

…….

ΚΟΝΤΑΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΠΤΥΧΙΟΥΧΟΣ ΤΕΧΝΟΛΟΓΟΣ ΜΗΧΑΝΙΚΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ