ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΑΧΕΙΡΙΣΗ ΓΝΩΣΗΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ – ΗΥ566
ΔΙΔΑΣΚΩΝ: ΓΡΗΓΟΡΗΣ ΑΝΤΩΝΙΟΥ

Αναφορά ΗΥ-566
Ομάδα 2

Γενιτσαρίδη Ειρήνη	ΜΕΤ 586
Δασκαλάκη Ευαγγελία	ΜΕΤ 598
Δρόσης Γιάννης	ΜΕΤ 593
Πέτσας Αθανάσιος	ΜΕΤ 600
Στρουμπούλης Χρήστος	ΜΕΤ 617
Τζαγκαράκης Χαράλαμπος	ΜΕΤ 582
Τζερμιάς Ζαχαρίας	ΜΕΤ 607
Τσάμης Γιώργος	ΜΕΤ 596
Φιλιππάκη Χρυσή	ΜΕΤ 584
Ψαράκη Μαρία-Γεωργία	ΜΕΤ 556

Σεπτέμβριος 2010

1. Σύντομη περιγραφή του συστήματος

Σε αυτήν την αναφορά παρουσιάζεται η υλοποίηση ενός συστήματος για επεξήγηση αποδείξεων στο σημασιολογικό ιστό, χρησιμοποιώντας αναιρέσιμη λογική. Το σύστημα παράγει αυτόματα επεξηγήσεις αποδείξεων χρησιμοποιώντας το σύστημα λογικού προγραμματισμού XSB.

Μέσω του συστήματος EMERALD δημιουργήθηκε μία εφαρμογή η οποία αποτελείται από agents που ανταλλάσσουν ACL μηνύματα. Στη συγκεκριμένη εφαρμογή υλοποιήθηκε ένας client (customer) agent, που στέλνει σε έναν server (broker) agent μήνυμα επερώτησης σε μία βάση γνώσης του server. Στην συνέχεια ο server στέλνει την απάντηση πίσω στον client και εφ΄ όσον του ζητηθεί, την απόδειξη της απάντησης.

Επίσης, μέσω του συστήματος EMERALD δημιουργήθηκε μία ακόμα εφαρμογή στην οποία ένας client agent στέλνει σε ένα server agent την απόδειξη κάποιας επερώτησης και ο server αφού ελέγξει την εγκυρότητα της απόδειξης, στέλνει πίσω κατάλληλο μήνυμα.

Στο κεφάλαιο 2 περιγράφεται αναλυτικά την λειτουργία του server της πρώτης εφαρμογής και την επικοινωνία του με τον client, καθώς και οι τροποποιήσεις που χρειάστηκαν να γίνουν σε κάποια αρχεία προκειμένου να υποστηριχθούν οι agents από το EMERALD. Όμοια στο κεφάλαιο 3 περιγράφεται η λειτουργία του server της 2ης εφαρμογής και την επικοινωνία του με τον client. Στο κεφάλαιο 4 περιγράφονται οι αλλαγές που έγιναν στα tags της ruleml της απόδειξης που παράγεται από το σύστημα.

2. Λειτουργία του agent server της 1ης εφαρμογής

Η επικοινωνία του client με τον server επιτυγχάνεται με τα παρακάτω βήματα:

1. Ο client agent στέλνει στον server ένα ACL μήνυμα σε μορφή RuleML, το οποίο περιέχει την επερώτηση του client και τα δεδομένα στα οποία θα γίνει η επερώτηση και τους κανόνες που εκφράζουν της προτιμήσεις του client. Συγκεκριμένα, το RuleML περιλαμβάνει ένα link σε ένα RDF αρχείο το οποίο περιέχει τα δεδομένα, ενώ το ίδιο το RuleML αποτελείται από τους κανόνες και την επερώτηση μαζί με το είδος της απάντησης που επιθυμεί (απλή απάντηση ή απάντηση μαζι με απόδειξη).

2. Ο server agent περιμένει να δεχτεί ένα ACL μήνυμα από τον client. Μόλις λάβει κάποιο ACL μήνυμα τότε πραγματοποιεί τις ακόλουθες ενέργειες:

a. Αρχικά, φορτώνει το RuleML αρχείο και αναζητά μέσα σε αυτό το link στο RDF.
b. Αποθηκεύει τοπικά το RDF αρχείο μέσω του link.
c. Χρησιμοποιεί έναν RDF parser και μετατρέπει το RDF αρχείο σε ένα αρχείο με δεδομένα (facts) σε μορφή Prolog.
d. Χρησιμοποιεί έναν RuleML parser και μετατρέπει το RuleML αρχείο σε ένα αρχείο κανόνων σε μορφή Prolog.
e. Χρησιμοποιεί έναν RDF parser και μετατρέπει το query που βρίσκεται μέσα στο RuleML σε μορφή συμβατή με την Prolog.
f. Στη συνέχεια, χρησιμοποιεί μία Prolog engine στην οποία φορτώνει τα αρχεία facts, rules και εκτελεί την επερώτηση.
g. Μετατρέπει την απάντηση της Prolog engine σε RDF μορφή και την αποθηκεύει τοπικά.
h. Χρησιμοποιεί τον RDF parser για να αναζητήσει την προτίμηση του χρήστη για απλή απάντηση στην επερώτηση ή για απάντηση μαζί με απόδειξη η οποία βρίσκεται μέσα στο RuleML του ACL μηνύματος.
i. Εάν ο client επιθυμεί απλή απάντηση, του επιστρέφει το link του RDF αρχείου που περιέχει την απάντηση.
j. Εάν ο client επιθυμεί απάντηση μαζί με απόδειξη τότε :
    - Δημιουργείται ένα αντικείμενο του Invoker και φορτώνονται τα προγράμματα της DR-Prolog.
    - Φορτώνονται τα αρχεία με τα δεδομένα (facts.P) και το αρχείο με τους κανόνες (rules.P).
    - Εκτελείται το query του customer και επιστρέφεται η απόδειξη η οποία αποθηκεύεται τοπικά.
    - Επιστρέφεται στον client ένα link σε ένα αρχείο που περιέχει την απάντηση μαζί με την απόδειξη.

Παρακάτω φαίνεται ένα screenshot με τον κώδικα των παραπάνω βημάτων:

```
//----------- VRISKOUME STO ARXEIO RULEML TO LINK TOU RDF ---------------
String link = findRdfLinkFromRulemlFile(ruleml);

//----------- APOTHIKEUOUME APO TO LINK TO RDF ---------------
saveUrl("C:\\myRdf.rdf", link);

//----------- METATREPOUME TA FACTS APO TO ARXEIO RDF SE FACTS.P ---------------
String factsFileName = "C:\\facts.P";
RdfParser.toProlog("C:\\myRdf.rdf",factsFileName);

//----------- METATREPOUME TA RULES APO TO ARXEIO lala.ruleml SE lala.p ---------
String rulesFileName =  RuleMLparser.createRules(ruleml);

//----------- DIMIOURGOUME APO TO ARXEIO RULEML TO STRING QUERY ---------
String q = RuleMLparser.getQuery(ruleml);//Get the query (and what it represents) from the ruleml file
String query = q.substring(0, q.indexOf("%")-1);//Get the main query

//--------------- GET THE ANSWER OF THE QUERY FROM THE ENGINE ----------------
String query_answer = "";
query_answer = getResult( ruleml, query, factsFileName, rulesFileName);

//--------------- CREATE THE QUERY FOR THE RDF FILE --------------------------
String queryForRdf = query.substring( query.lastIndexOf('('), query.lastIndexOf(')') );
queryForRdf = queryForRdf.replace('(', '[');
queryForRdf = queryForRdf.replace(')', ']');

//--------------- WRITE THE RDF FILE WITH THE ANSWER -------------------------
ResultParser resultRdfParser = new ResultParser();
resultRdfParser.toRDF(queryForRdf, "acceptable", query_answer, "C:\\answer.rdf");////////////////////////////

boolean upperfound=containsVariables(query);
```

2.1.Δημιουργία Reasoner για την 1ης εφαρμογή

Κατασκευάστηκε ο FinalReasoner που εκτελεί reasoning services και αντικαταστάθηκε ο παλιός που είχε το Emerald με αυτόν (FinalReasoner.java).

Επίσης, οι κλάσεις που δημιουργούνται από τον FinalReasoner.java προστέθηκαν στο φάκελο του Emerald (C:\jade\EMERALD\FinalReasoner.class) και ο Reasoner αυτός ορίστηκε ως ο DR-Device agent στο Emerald.bat file :

C:\jade\EMERALD\EMERALD.bat
//
rem @echo off
echo EMERALD 1.0
*if EXIST DR-DeviceAgent set dr-device-agent=DR_Reasoner:**FinalReasoner***
if EXIST R-DeviceAgent set r-device-agent=R_Reasoner:ReasoningAgent_R
if EXIST SPINdleAgent set spindle-agent=SPINdle_Reasoner:SPINdleAgent
if EXIST ProvaAgent set prova-agent=Prova_Reasoner:provaAgent
java jade.Boot -gui %r-device-agent% %dr-device-agent% %prova-agent% %spindle-agent%
//

3. Λειτουργία του agent server της 2ης εφαρμογής

Η επικοινωνία του client με τον server επιτυγχάνεται με τα παρακάτω βήματα:

1. Ο client agent στέλνει στον server ένα ACL μήνυμα, το οποίο καθορίζει τα αρχεία που περιέχουν τα δεδομένα και τους κανόνες (theory.xml), καθώς και την απόδειξη που δόθηκε ως απάντηση στην επερώτηση που έκανε ο client της προηγούμενης εφαρμογής (proof.xml).
2. Ο server agent περιμένει να δεχτεί ένα ACL μήνυμα από τον client. Μόλις λάβει κάποιο ACL μήνυμα τότε πραγματοποιεί τις ακόλουθες ενέργειες:
    k. Διαβάζει το περιεχόμενο του αρχείου του οποίου το link βρίσκεται στο ACL message και περιέχει τα links των αρχείων για τα δεδομένα, τους κανόνες (theory.xml) και την απόδειξη (proof.xml).
    l. Φορτώνει από το πρώτο αρχείο (theory.xml) τα δεδομένα και τους κανόνες.
    m. Φορτώνει από το δεύτερο αρχείο (proof.xml) την απόδειξη.
    n. Ελέγχει την εγκυρότητα της απόδειξης.
    o. Αποστέλλει πίσω στον client με ACL message αν η απόδειξη είναι έγκυρη ή όχι.

3.1. Δημιουργία Proof Validator 2ης εφαρμογής

Κατασκευάστηκε ο ReasonerForTeam3 που εκτελεί Proof Validation Services.

Οι κλάσεις που δημιουργούνται από τον ReasonerForTeam3.java προστέθηκαν στο φάκελο του Emerald (C:\jade\EMERALD\ReasonerForTeam3.class) και ο Proof Validator αυτός ορίστηκε ώς ο DR-Device agent στο Emerald.bat file :

C:\jade\EMERALD\EMERALD.bat
//
rem @echo off
echo EMERALD 1.0
if EXIST DR-DeviceAgent set dr-device-agent=DR_Reasoner: ***ReasonerForTeam3***
if EXIST R-DeviceAgent set r-device-agent=R_Reasoner:ReasoningAgent_R
if EXIST SPINdleAgent set spindle-agent=SPINdle_Reasoner:SPINdleAgent
if EXIST ProvaAgent set prova-agent=Prova_Reasoner:provaAgent
java jade.Boot -gui %r-device-agent% %dr-device-agent% %prova-agent% %spindle-agent%
//

Προκείμενου η εφαρμογή αυτή να γίνει register από το Emerald, έγιναν αλλαγές στα αρχεία broker.clp και customer.clp ορίζοντας την παρεχόμενη υπηρεσία ως "Proof_Validation_Service". Επίσης έγινε αλλαγή στα αρχεία breq.txt και creq.txt ώστε τόσο ο Broker όσο και ο Customer να απαιτούν την ύπαρξη της υπηρεσίας αυτής. Η υλοποίηση του ReasonerForTeam3 είναι αντίστοιχη με αυτήν του FinalReasoner.

4. Αλλαγές που πραγματοποιήθηκαν στα ruleml tags

Αλλαγές στα tags

Πραγματοποιήθηκε αλλαγή στα tags του proof που παράγεται, έτσι ώστε αυτά να είναι συμβατά με την νέα έκδοση της RuleML. Παράγεται ένα proof.xml το οποίο περιέχει την απόδειξη της απάντησης στην ερώτηση του customer. Τα tags τα οποία υπέστησαν αλλαγή, βρίσκονται στον παρακάτω πίνακα:

New tag	Old tag
Head	_head
Body	_body
Atom	atom
Not	not
Op	_op
Fact	fact