CS6770 KRR Programming Assignment Questions

Instructions for Students

Note that the problems will be assigned to the teams randomly.

- **Sample I/O:** https://drive.google.com/drive/folders/1VhozSaB9p-OsH99xms2U2_75h6ZGSrpV?usp=sharing

- Problems 1 and 13 are optional (not allocated to any team currently) and they can be taken by any team as a replacement for an allocated problem (if interested).
  - Note that, however, a maximum 1 Team can take problems 1, 13. It will be allocated on a First Come First Serve basis.
  - A member of the interested team should send an email to one of the TA (cc to Prof. Khemani) stating the selected problem (1 or 13) on/before Sunday 26th March 2023.

- If your team is not happy with the problem allocated or is interested in other problems, a mutual swap between two teams is allowed till Sunday 26th March 2023.
  - Both teams must send a separate email to one of the TA (cc to Prof. Khemani) about this on/before Sunday 26th March 2023, only then it will be considered.

- Watch XML Package Usage Video Here: https://youtu.be/uU2Yq7NobEU (This video contains information about the old KRR package, the following new package is almost similar)

  - Updated KRR (Text to XML) parser: https://drive.google.com/file/d/120AWl4zQ-zWmuQT-yslPvXapxDTzE21s/view?u sp=sharing

  - Updated Language specification document: doc/krr-language-specification-v4.2.pdf in the above package

  - You can use it to create input/output text to XML files for your program

- You will need Java 8 SDK/JRE installed on your system to run it

- **Languages allowed - C/C++, Java, Python**

- **For the teams with the same problem allocated, if copying is found then both groups will get a U grade.**

- **Deadlines:**
  Groups :
  - A B C : 13th April 2023
  - D E   : 20th April 2023
  - F G H : 27th April 2023

The assignments listed below implement parts of a KRR software package that will be implemented by this class, XML will be used for data interchange between software modules. For ease of use, write the input formulas in a user-friendly format (described in the Language Specification document) and use the given program to convert user input to XML output that is very close to an abstract syntax tree.

The parsers and XML converters are developed by Baskaran Sir and tested in Java 8.

Group A (Optional)

1. NL to FOL (1 Team)

Given a set of natural language sentences (like a word problem) construct a corresponding FOL representation. Students interested in NLP can take this.

Group B

2. FOL to Clause Form

Given a set of FOL formulas in a text file, convert each one into clause form. For each predicate and function add the appropriate equality axioms and convert them into clause form. Choose an appropriate naming convention for Skolem constants and functions. The output in the clause form should be produced as text as well as XML files.

3. Forward Chaining

Implement a Forward Chaining system to accept a KB and generate a proof for a given theorem/query. The proof should be in the form of a numbered sequence of statements starting with the statements from the KB that are used in the proof. Your system should read a set of formulas in clause form from an XML file and convert it to Implicit Quantifier notation. The Unification algorithm that operates on the above notation has to be implemented by you.

4. Resolution Method

For a given set of formulas in the clause form generated by assignment 2, implement the resolution method. Implement the Unification algorithm to be used by your resolution method. Allow the user to choose between a set of strategies. The algorithm should output either a substitution or FALSE, along with a trace in a text file. *Optional, for extra credits, display the derivation DAG in graphical form*.

Group C

5. Backward Chaining

Implement a Prolog-like backward chaining system using a depth-first strategy. Accept rules from a text file (as per the language specification) and convert them into XML. Accept the goal from a similar file. Show the final proof for the goal (if possible, graphically as a tree). One should be able to opt for more than one solution on giving a goal. Allow the use of Cut. The algorithm should output either a substitution or FALSE, along with a trace in a text file.

Group D

6. ALC KB to NNF

Given an ALC KB and a query, use the given parser to convert the inputs to XML format. Write a program to extract ALC axioms and assertions from the XML file and convert them to Negation Normal Form (NNF) and write the NNF output to another XML file that can be shared with other teams.

Treat “Thing” as the top concept and “Nothing” as the bottom concept.

7. ALC Tableau

Implement ALC Tableau. For an ALC KB and a query given in Negation Normal Form generated by assignment 6, use ALC Tableau to determine whether the query is entailed by the KB.

Treat “Thing” as the top concept and “Nothing” as the bottom concept.

8. ALC Taxonomy Builder - Subsumption Hierarchy

Implement a taxonomy builder. Given an ALC KB in Negation Normal Form generated by assignment 6, reuse the ALC Tableau implementation from assignment 7 and write a program to build the taxonomy (subsumption hierarchy) of concepts defined in the KB.

Treat “Thing” as the top concept and “Nothing” as the bottom concept.

Group E

9. ALC Taxonomy Builder - Instance Retrieval

Given an ALC KB and a concept description C, prepare the taxonomy using assignment 8 and reuse the ALC Tableau implementation from assignment 7 and write a program to identify all the named individuals that belong to C.

The program should scan ConceptName from the console (in the loop until the user wants to exit) at run time and print corresponding named individuals on-screen and also save to file <ConceptName>.individuals.txt.

Treat “Thing” as the top concept and “Nothing” as the bottom concept.

10. ALC Taxonomy Builder - Identify all classes that a Named Individual belongs to (Instance Checking)

Given an ALC KB and a named individual N, prepare the taxonomy using assignment 8 and reuse the ALC Tableau implementation from assignment 7 and write a program to identify all the classes that N belongs to.

The program should ask the user to enter NamedIndividual and scan NamedIndividual from the console (in the loop until the user wants to exit) at run time and print corresponding classes that it belongs to on-screen and also save to file <NamedIndividual>.classes.txt.

Treat “Thing” as the top concept and “Nothing” as the bottom concept.

Group F

11. Inheritance networks

Allow the user to create an inheritance hierarchy graphically (or read from a file statements describing IsA and InstanceOf relation). Given a taxonomy, answer the question a → p? Create extensions, and identify the preferred extensions.

Group G

12. Discrete Event Calculus problem to FOL representation

Implement a system that encodes a Discrete Event Calculus problem to FOL representation so that one can do deductive reasoning on a given event calculus problem using a theorem prover (from assignments 2-3 or 2-4).

Group H (Optional)

13. Epistemic Logic Problem to FOL Representation (1 Team)

Implement a system that encodes an epistemic logic problem to FOL representation so that one can do deductive reasoning on a problem specified in epistemic logic using a theorem prover (from assignments 2-3 or 2-4).