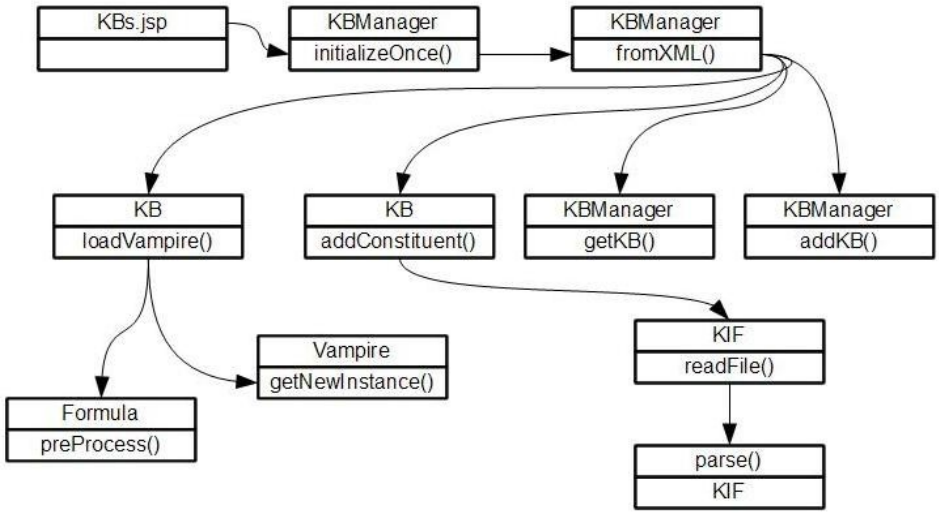
Sigma Implementation

The following Java objects and corresponding methods are used and called for the indicated actions (Note: Only methods / objects that either affect the flow or may need potential modification is included). These objects, methods or flows may possibly be impacted by the addition of a persistence layer.

Initialization

When Sigma is first started, and the user first logs in, all knowledge bases are initialized, which means that KIF files indicated in config.xml are loaded and processed into Java Objects.



KBs.jsp

First redirect after user logs in.

KBManager initializeOnce()

Entry point to create Kbs and add to KBManager.

KBManager fromXML()

Reads config.xml, including all configured knowledge bases and corresponding constituents.

KBManager addKB()

Adds a new KB Object to the HashMap of KBs managed by this KBManager.

KBManager getKB()

Retrieves a named KB from the HashMap of Kbs maintained by this KBManager.

KB addConstituent()

Adds a specified KIF file to this KB. In this flow, KBManager reads a list of KIF files per KB defined in config.xml and passes on the canonical path to addConstituent.

KIF readFile()

Opens and reads the file specified by addConstituent.

KIF parse()

Processes and parses the contents of the constituent, saving it in a variety of HashMaps and ArrayLists in the KB Object.

KB loadVampire()

Prepares the formulas to be loaded into the theorem prover instance.

Formula prePocess()

The method in charge for doing the processing of formulas.

Vampire getNewInstance()

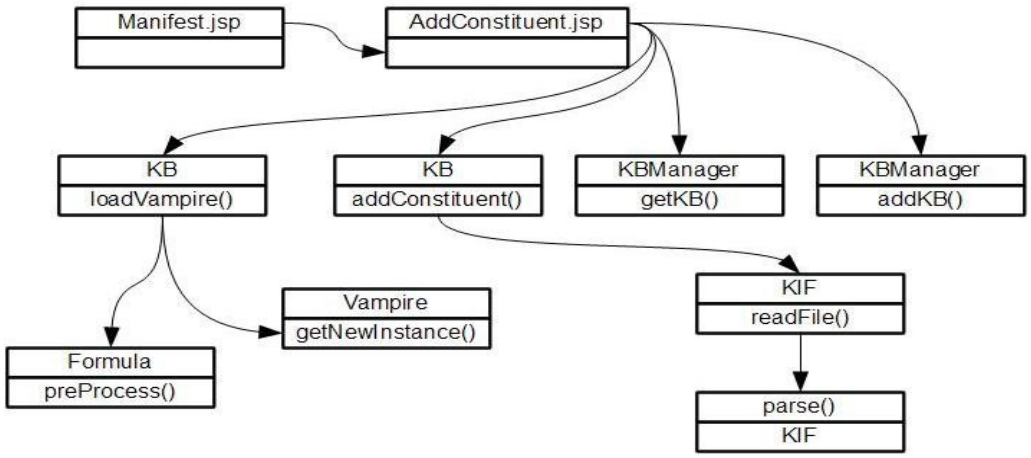
Loads a vampire instance with an arraylist of formulas after preprocessing.

Open questions / issues:

Does vampire do this in-memory as well? If this is done in-memory, will it matter then if we do a persistence layer that accommodate gigabytes of information, but have a theorem prover that will be limited by it? Is it possible to just “assert” on the fly (i.e. only assert relevant formulas when an ask is done) instead of loading all the formulas onto vampire?

Adding a KIF file to a KB

Adding a KIF file to a KB means having to process that KIF file as well as adding assertions to the theorem prover.



Manifest.jsp

Entry point. File that redirects to AddConstituent.jsp to initiate adding of KIF file. Invokes KBManager.addKB() if the KB does not exist yet. Gets the KB to add the constituent to using KBManager.getKB().

AddConstituent.jsp

Entry point. File that invokes the command to upload the KIF file.

KB addConstituent()

Adds a specified KIF file to this KB. In this flow, a single constituent is added to the KB.

KIF readFile()

Opens and reads the file specified by addConstituent.

KIF parse()

Processes and parses the contents of the constituent, saving it in a variety of HashMaps and ArrayLists in the KB Object.

KB loadVampire()

Prepares the formulas to be loaded into the theorem prover instance.

Formula prePocess()

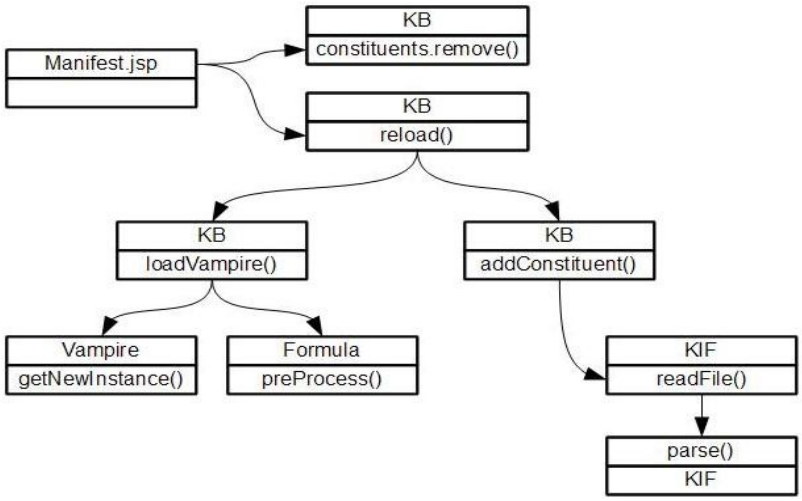
The method in charge for doing the processing of formulas.

Vampire getNewInstance()

Loads a vampire instance with an arraylist of formulas after preprocessing.

Deleting a KIF file from a KB

When a constituent is deleted from a KB, this implies that all processed formulas related to a term that belongs to that constituent needs to be deleted, assertions loaded onto the theorem prover related to the constituent need to be deleted, etc.



KBs.jsp

Entry point. Invokes the method that reloads the KB without the removed constituent. Also removes constituent from config.xml.

KB.constituents remove()

Removes the constituent name from the HashMap of constituents that this KB contains.

KB reload()

Clears all the contents of this KB, and then re-adds all the constituents that this KB has. Since the constituent that was removed in the delete action was removed in the HashMap of constituents, it will not be reloaded.

KB addConstituent()

Adds a specified KIF file to this KB. In this flow, KBManager reads a list of KIF files per KB defined in config.xml and passes on the canonical path to addConstituent.

KIF readFile()

Opens and reads the file specified by addConstituent.

KIF parse()

Processes and parses the contents of the constituent, saving it in a variety of HashMaps and ArrayLists in the KB Object.

KB loadVampire()

Prepares the formulas to be loaded into the theorem prover instance.

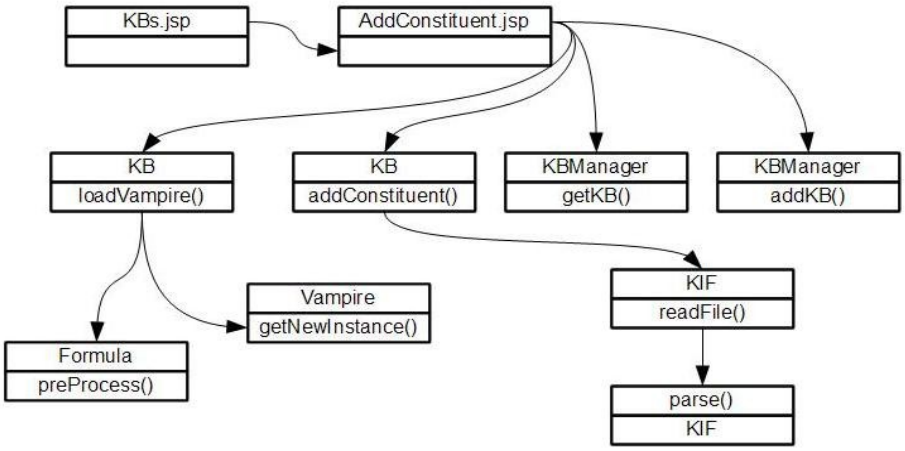
Formula prePocess()

The method in charge for doing the processing of formulas.

Vampire getNewInstance()

Loads a vampire instance with an arraylist of formulas after preprocessing.

Adding a KB



Same process as Adding a Constituent, only that it is KBs.jsp that initiates it instead of Manifest.jsp.

Deleting a KB

Deleting a knowledge base implies removing the KB from the list of KBs that the Kbmanager manages.

KBs.jsp

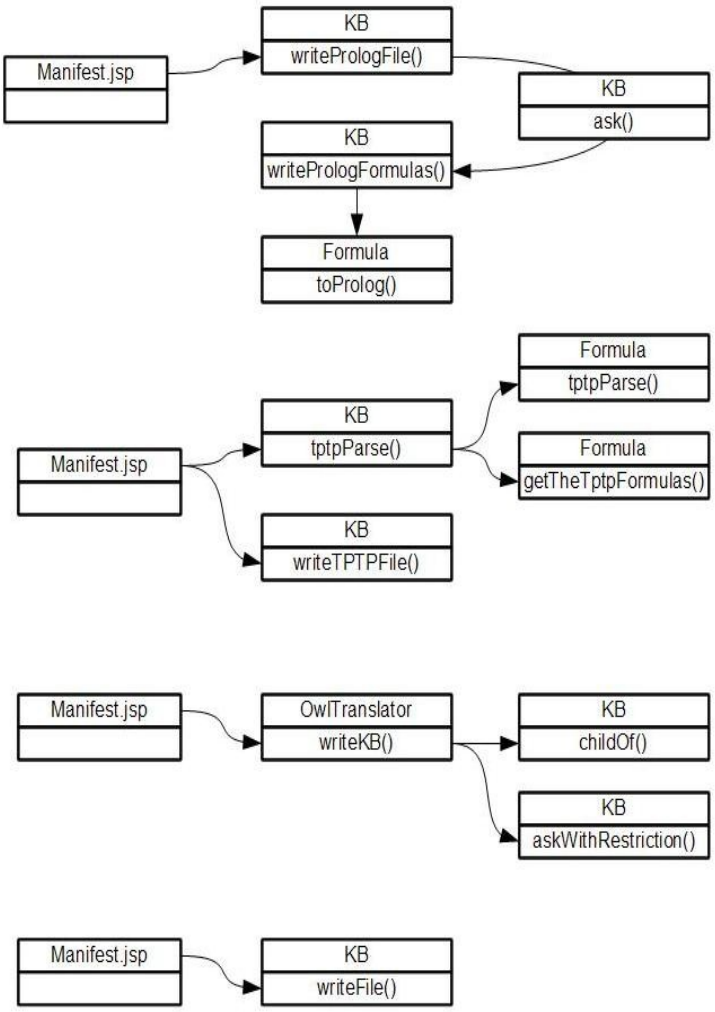
Entry point. Indicates the KB to be deleted and invokes the appropriate method. Also removes the KB and its constituents from config.xml.

KB removeKB()

Removes the KB from the HashMap of KBs that this KBManager maintains. Also terminates the Vampire instance being used.

Saving a file from KB

Sigma allows saving a file from the KB to the following formats: OWL, Prolog, KIF, TPTP & TPTP FOL.



KB writePrologFile()

Method that initiates retrieving terms and formulas and converting them to prolog syntax.

KB ask()

Returns an arraylist of formulas that match the request.

KB writePrologFormulas()

Converts formulas from the KB into prolog syntax.

Formula toProlog()

Converts a specific formula to prolog syntax.

KB tptpParse()

Initiates parsing of KB contents into tptp format.

Formula tptpParse()

Parses a formula from the KB into its tptp form.

Formula getTheTptpFormula()

Returns a formula in its tptp form.

KB writeTPTPFile()

Saves tptp version of formulas in a file.

OwlTranslator writeKB()

Initiates conversion of formulas from KB into OWL syntax

KB childOf()

Determines if a term is the child of a given term.

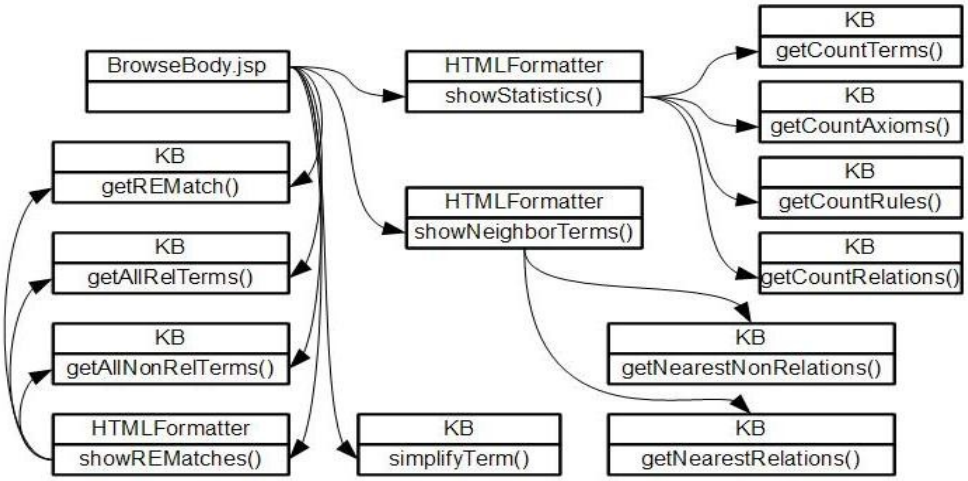
KB askWithRestriction()

Returns an ArrayList of formulas that contain the given terms in the specified argument positions.

KB writeFile()

Re-assembles reloaded KIF Files and saves it into a file.

Searching for a term using the 'KB Word' input box



BrowseBody.jsp

Entry point to either clicking on term links or searching for a term.

HTMLFormatter

Returns search results in HTML format.

HTMLFormatter showStatistics()

Gets counts of different items in the ontology, including terms, axioms, rules and relations.

HTMLFormatter showNeighborTerms()

Searches through an alphabetical list and looks for the 20 terms before and 20 terms after what is closest to the search term. It does this for both relations and nonrelations.

KB getREMatch()

Gets a term and returns an ArrayList containing every term in the KB that has a match with the RE.

KB getAllRelTerms()

Returns a list of all rel terms in an ArrayList.

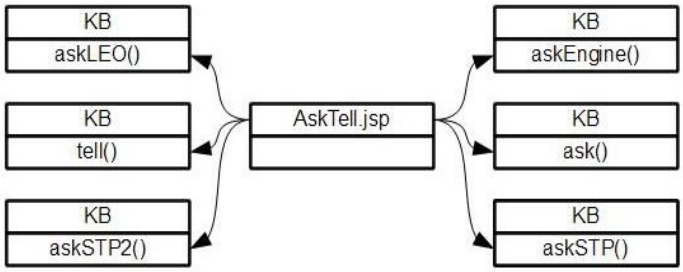
KB getAllNonRelTerms()

Returns a list of all non-rel terms in an ArrayList.

HTMLFormatter showREMatches()

Format search results of a regular expression-based search.

Ask / Tell



AskTell.jsp

Allows users to input assertions or queries.

KB ask()

Returns an arraylist of formulas that match the request.

KB askEngine()

Submits a query to specified InferenceEngine object. Returns an XML formatted String that contains the response of the inference engine.

KB askSTP()

Submits a query to the STP InferenceEngine

KB askSTP2()

Submits a query to the STP2 InferenceEngine

KB askLEO()

Submits a query to the LEO InferenceEngine

KB askSInE()

Submits a query to the SInE InferenceEngine