(51) International Patent Classification: G06P 17/28 (2006.01)
(21) International Application Number: PCT/MY20 10/000208
(22) International Filing Date: 14 October 2010 (14.10.2010)
(26) Publication Language: English
(30) Priority Data: PI 2010/02500 27 May 2010 (27.05.2010) MY
(71) Applicant (for all designated States except US): MIMOS BHD. [MY/MY]; Technology Park Malaysia, 57000 Kuala Lumpur (MY).
(72) Inventors: and

(74) Agent: CHUAH, Jern Ern; Advanz Fidelis Sdn. Bhd., Suite 609, Block D, Phileo Damansara 1, No. 9, Jalan 16/1, 46350 Petaling Jaya, Selangor Drul Ehsan (MY).

Declarations under Rule 4.17:
- as to the identity of the inventor (Rule 4.17(ii))

[Continued on next page]

(54) Title: A METHOD FOR CONDUCTING A VISUAL QUERY AND A SYSTEM THEREFOR

(57) Abstract: The present invention relates to a method that uses a recursive means to conduct a visual query graphically to retrieve information from ontologies. The method is comprised the steps of loading an ontology, forming a visual query graph, converting the visual query graph into a recursive query graph by using a graphical notation, interpreting the recursive query graph and converting the recursive query graph into a recursive internal representation that allows recursion, retrieving query results from the internal representation; and visualizing the results. The present invention is also related to a system (100) that uses a recursive means to conduct a visual query graphically to retrieve information from ontologies. The system (100) is comprised of an ontology loader (10), a recursive query constructor component (20) for forming a visual query graph and for converting the visual query graph into a recursive query graph by using a graphical notation, a recursive graphical query language interpreter component (30) for interpreting the recursive graph and for converting the recursive query graph into a recursive internal representation that allows recursion, and a results visualizer (50).
Published:

— with international search report (Art. 21(3))
FIELD OF THE INVENTION

The present invention relates to a method for conducting a visual query on information and a system therefor.

BACKGROUND ART

In the art, it is known that there exists a method and a system that conducts a visual query on information. It is also common that there also exists a method and a system that uses a graphical query system that allows information retrieval from the ontologies. Such a prior art allows users to create visual query graphs, to interpret the graph automatically, and to visualize the results.

Besides that, in these existing method and system, the user is usually required to log into the system and load an ontology. The system is usually consisted of a visual query constructor that allows the user to construct the visual query graph by selecting the concepts and properties from the loaded ontology data. An interpreting component of the system is then responsible for interpreting the visual query graph to create an equivalent internal representation. Next, the internal representation is sent to the query processor for retrieval of results. The results are also processed later to generate the graphs of the answers as well as the textual representation and are then displayed to the user.

Normally, the system also supports utility functions such as loading/saving graphs, performing regular graph operations such as adding nodes, deleting node and so on.

Accordingly, the queries often involve complicated graph transversal mechanisms due to the complex information in the ontologies. As such, there arises a problem in retrieving sub-graphs of unknown depth from the ontology. Moreover, knowledge is often updated dynamically. To this end, there is a need to retrieve information without the full knowledge of the depth of graphs.
In light of the above, recursion is a method that solves difficult problems by reducing a problem to simpler, but identical sub-problems. It is therefore believed that recursion enables retrieval of sub-graphs of unknown depths even if the data is updated dynamically.

However, no recursion capabilities are present in the existing method and system. The ability to retrieve sub graphs from the said unknown depth is therefore significantly limited. In the existing method and system, recursion is generally not possible to be represented, particularly in terms of a graph notation. There has been no standard means to represent the recursion in a graphical query language. Furthermore, a mechanism is also required to convert a recursive query graph into a program that is able to handle recursion. It is also required that the results of the recursive queries be displayed in a user-friendly and intuitive manner.

To this end, there appears a need for a method and a system that create recursive query graphs, that automatically interpret the said graph into recursive programs, as well as technique to visualize the recursive results graphically.

**SUMMARY OF THE INVENTION**

Accordingly, to solve the disadvantages and drawbacks of the prior art, there is provided a method and a system for conducting a visual query on information. More particularly, the present invention is related to a method and a system that use a recursive means to conduct a visual query graphically to retrieve information from ontologies.

According to one aspect of the present invention, the method is comprised the steps of loading an ontology, forming a visual query graph, converting the visual query graph into a recursive query graph by using a graphical notation, interpreting the recursive query graph and converting the recursive query graph into a recursive internal representation that allows recursion, retrieving query results from the internal representation; and visualizing the results.

According to another aspect of the present invention, the system is for conducting a
visual query on information. The system is further for conducting a recursive graphical query to retrieve information from ontologies. The system is further for conducting a recursive graphical query to retrieve information from ontologies.

The system is comprised of an ontology loader, a recursive query constructor component for forming a visual query graph and for converting the visual query graph into a recursive query graph by using a graphical notation, an interpreting component for interpreting the recursive query graph and for converting the recursive query graph into a recursive internal representation that allows recursion, and a results visualizer.

It is an object of the present invention to provide a method and a system that are able to support recursive capabilities to queries.

It is also an object of the present invention to provide a method and a system that are able to retrieve the information without the full knowledge of the depth of the graphs.

It is further an object of the present invention to provide a method and a system that create recursive query graphs, or convert the visual query graph into a recursive query graph by using a graphical notation.

It is further an object of the present invention to provide a method and a system that interpret the recursive query graph and that convert the recursive query graph into a recursive internal representation that allows recursion.

It is also an object of the present invention to provide a method and a system that construct recursive rules and a query representation that supports recursion.

It is a final object of the present invention to provide a method and a system that visualize the recursive results graphically.

The present invention consists of certain novel features and a combination of parts hereinafter fully described and illustrated in the accompanying drawings and particularly pointed out in the appended claims; it being understood that various changes in the
details may be without departing from the scope of the invention or sacrificing any of the advantages of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

For the purpose of facilitating an understanding of the invention, there is illustrated in the accompanying drawings the preferred embodiments from an inspection of which when considered in connection with the following description, the invention, its construction and operation and many of its advantages would be readily understood and appreciated.

FIG. 1 is a flow chart showing the process of the query system of the present invention.
FIG. 2 shows a system architecture of the recursive visual query system of the present invention.
FIG. 3 shows a visualized recursion.
FIG. 4 is a flow chart showing the method to construct and interpret the recursive graphs.
FIG. 5 is a flow chart that shows the process flow involved in interpreting the recursive graphs.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention relates to a method and a system that conduct a visual query on information. More particularly, the present invention relates to a method and a system 100 that conduct a graphical query to retrieve information from ontologies using a recursive means. Hereinafter, the method and the system 100 shall be described according to the preferred embodiments of the present invention and by referring to the accompanying description and drawings. However, it is to be understood that limiting the description to the preferred embodiments of the invention and to the drawings is merely to facilitate discussion of the present invention and it is envisioned that those skilled in the art may devise various modifications without departing from the scope of the appended claim.
Referring now to different figures of the drawings, the method and the system 100 of the present invention are shown.

A graphical query system and method have conventionally been used in a visual query system (VQS) for information retrieval from ontologies. It has also been known that the method and the system allow users to create visual query graphs, to interpret the graph automatically, and to visualize the results. However, no recursion capabilities are usually present or incorporated in the existing system, thus limiting the ability to retrieve subgraphs of unknown depth. It is therefore preferred that the present invention overcomes the existing problem by introducing the creation of recursive query graphs and a method to automatically interpret the graph into recursive programs, as well as technique to visualize the recursive results graphically.

It is preferred that the present system 100 is consisted of a graphical User Interface (GUI) in which the input comes in the form of a user interaction with the graphical elements. The output of the system is preferably in the form of result graphs, readable text form, or a combination of both.

In the present invention, the method is preferably comprised of the steps as follows:

1. The user logs into the system, and loads an ontology.
2. The user constructs the visual query graph by selecting concepts and properties from the loaded ontology data.
3. The regular visual query graph is converted into a recursive query graph.
4. The recursive query graph is interpreted and converted into an internal structure/representation that is capable of supporting recursion.
5. The internal structure is sent to the Query Processor engine 40 to retrieve the results.
6. The results are then processed to generate graphs of the answers, textual representation therefrom, or a combination of both. The results are also allowed to be processed into other forms of representation.
7. Next, the results are displayed to the user.
8. Additionally the system 100 also supports utility functions for instance to load/save graphs and to perform regular graph operations such as add nodes, delete node and so on.
Referring now to FIG. 1, the flow chart depicting the overall view of the preferred recursive visual semantic query is shown. The method and the system 100 of the present invention will be hereinafter described together in greater detail.

Referring now to FIG. 2, the preferred architecture of the system 100 and the preferred components thereof are shown. The preferred components contributing to the core of the present invention are illustrated in FIG. 2. The system 100 is preferably comprised of components that include the recursive query constructor component 20, recursive graphical query language interpreter component 30 and the recursive results visualizer component 50. It is also preferred that the system 100 is further comprised of a query processor component 40. These preferred components and the preferred method associated with the components are hereinafter described together in greater detail.

**Ontology Loader**

This component is preferably invoked when the user preferably selects an ontology to be loaded. The concepts and properties of the selected ontology are preferably populated on the said GUI for further processing.

**Recursive query constructor**

This component 20 preferably deals with the query graph construction phase. The query graphs are preferably based on the subject-predicate-object notation of Resource Description Framework (RDF). It is preferred that the subjects and objects are represented by nodes whereas predicates are represented as edges between the subjects and objects.

The initial steps involved in creating a regular query preferably follows the same process as exercised by the existing visual query system. The user preferably starts by browsing/searching the list of concepts and selecting a concept from the concepts’ list. A node is then preferably created in a regular visual query graph. Upon selecting the created node, substantially all the properties that are attached to that particular selected
concept are preferably filtered in a properties panel. This filtering is preferably the result of the execution of a SPARQL Protocol and RDF Query Language (SPARQL) query snippet generated from the selected concept. Following this, the user is preferably allowed to keep on adding more properties and concepts to extend the said query graph in order to make complex queries later.

Once the regular query graph is created, the user is preferably able to convert the said regular query graph into a recursive query graph. The first step for this conversion is preferably done by the user selecting a property that he wishes to apply recursion for. Once the property is selected, recursion is preferably allowed to be applied by using a graph notation introduced in the present invention. It is preferred that a dotted arrow is drawn from the recursive property node (the node that properly points to) to the source concept, as illustrated in detail in FIG. 3.

Additionally, it is further preferred that more concepts and properties are allowed to be added to the recursive query graph to create relatively more complex query conditions.

**Recursive graphical query language interpreter**

Referring to FIG. 1, 2, and 4, it is preferred that this component 30 interprets the recursive query graph and converts the said query graph into a program structure that is capable of handling recursion. This is preferably achieved by generating recursive rules from the properties and concepts in the recursive query graph and finally by using the rules to create a recursive query that is executable in a knowledge base processor 70.

The preferred steps involved in the method of the present invention are described as follows:

- Iterate through the recursive graph to identify direct and indirect links.
- Creating a plurality of data from all the properties with the direct links.
- Once the facts are established, recursive heuristics are generated from the constructed data.
- Following this, the recursive heuristics are used to create logical programming structures that support recursion.
It should be noted that once the recursive heuristics are created, it is preferred that these heuristics are allowed to be reused to apply on other concepts and to create diverse queries.

Referring now to FIG. 4, the flow chart shows the preferred processing involved in the recursive query constructor component 20 and the recursive graphical query language interpreter component 30. More detailed flow chart of the preferred processes involved in recursive graphical query language interpreter component 30 is shown in FIG. 5.

**Query processor**

This component 40 is preferably concerned with establishing connections with the knowledge base 70 and executing the query statements created in the previous steps. It is preferred that the query processor component 40 is a generic component that is preferably allowed to be replaced with other knowledge base processor engines.

While in the foregoing specification this invention has been described in relation to certain preferred embodiments thereof and many details have been set forth for purpose of illustration, it will be apparent to those skilled in the art that the invention is susceptible to additional embodiments and that certain of the details described herein can be varied considerably without departing from the basic principles of the invention.
CLAIMS

1. A method for conducting a visual query on information, the method is comprised
the steps of:
   - loading an ontology;
   - forming a visual query graph;
   - interpreting the visual query graph to form an internal representation;
   - retrieving query results from the internal representation; and
   - visualizing the results;

characterized in that the method uses a recursive means to conduct a visual query
graphically to retrieve information from ontologies.

2. A method as claimed in Claim 1 wherein the step of interpreting the visual query
   graph comprises the step of:
   - converting the visual query graph into a recursive query graph by using a
     graphical notation; and
   - interpreting the recursive query graph and converting the recursive query graph
     into a recursive internal representation that allows recursion.

3. A method as claimed in Claim 2 wherein the method is further for retrieving sub-
   graphs of unknown depth from the ontology.

4. A method as claimed in Claim 3 wherein the step of converting the visual query
   graph comprises the steps of identifying a recursive property in the recursive query
   graph, and forming a visual indication representing an indirect link from a node with the
   recursive property to an origin node for establishing recursive relationship between the
   nodes; characterized in that the step of converting the visual graph allows adding
   additional query elements by linking a plurality of concepts, a plurality of properties, or
   both of the additional query elements to the recursive query graph; the step of
   identifying a recursive property comprises the step of selecting a property to be applied
   with the recursion by a user (60); the step of forming a visual indication comprises the
   step of applying the recursion by using the graph notation; and the step of converting the
visual query graph is conducted by a recursive query constructor (20).

5. A method as claimed in any one of Claims 3 and 4 wherein the step of interpreting the recursive query graph comprises the steps of generating a plurality of recursive rules from the concepts and the properties in the recursive query graph, the step of generating the recursive rules comprises the steps of:
   - iterating through the recursive graph to identify direct and indirect links,
   - forming a plurality of data from substantially all the properties with the direct links,
   - repeating the steps of iterating and forming the data if there are relatively more properties present, and
   - forming a plurality of recursive heuristics based on the formed data;

generating the internal representation using the recursive heuristics, the recursive heuristics are allowed to be applied on other concepts to form diverse queries; and

5. A method as claimed in Claim 5 wherein the step of loading an ontology comprises the steps of selecting an ontology to be loaded by the user (60), and populating the concepts and the properties of the selected ontology for further processing; characterized in that the step of loading the ontology is conducted by an ontology loader component (10).

7. A method as claimed in Claim 6 wherein in the step of forming a visual query graph is conducted by interacting with a list of the concepts and the properties in the ontology; characterized in that the step of forming a visual query graph comprises the steps of searching the list of concepts, selecting a concept from the list of concepts, forming a node in the visual query graph, generating a query snippet based on the selected concept, executing the query snippet, and filtering substantially all the properties that are attached to the selected concept in the list of properties; and the step of forming the visual query graph is conducted by the recursive query constructor component (20).
8. A method as claimed in Claim 7 wherein the step of retrieving the query results comprises the steps of sending the internal representation to a query processor component (40) for retrieving the results, the step of sending the internal representation comprises the steps of:
- establishing a plurality of connections with a knowledge base (70),
- executing a query statement generated based on the recursive query in a knowledge base processor, and
- retrieving the results; and
- processing the results for generating a plurality of graphs of a plurality of answers, textual representations, or both; characterized in that the step of retrieving the query results is conducted by the query processor component (40).

9. A method as claimed in Claim 8 wherein the step of visualizing the results comprises the step of forming a visual representation of the answers for displaying the results to the user; characterized in that the step of visualizing the results is conducted by a recursive results visualizer component (50).

10. A system (100) for conducting a visual query on information comprising:
- an ontology loader (10);
- a visual query constructor component for forming a visual query graph;
- an interpreting component for interpreting the visual query graph and converting the visual query graph into an internal representation; and
- a results visualizer;
characterized in that the system (100) uses a recursive means to conduct a visual query graphically to retrieve information from ontologies.

11. A system (100) as claimed in Claim 10 wherein the query constructor component comprises a recursive query constructor component (20) for using a graphical notation and for converting the visual query graph into a recursive query graph by using the graphical notation; the interpreting component comprises a recursive graphical query language interpreter component (30) for interpreting the recursive query graph and for converting the recursive query graph into a recursive internal representation that allows
recursion; and the results visualizer comprises a recursive results visualizer (50) for visualizing recursive results graphically.

12. A system (100) as claimed in Claim 11 wherein the recursive query constructor component (20) is further for identifying a recursive property in the recursive query graph, and for forming a visual indication representing an indirect link from a node with the recursive property to an origin node for establishing recursive relationship between the nodes; characterized in that the recursive query constructor component (20) allows the addition of additional query elements by linking the a plurality of concepts, a plurality of properties, or both of the additional query elements to the recursive query graph; the identification of a recursive property in the recursive query graph allows a user (60) to select a property to be applied with recursion; and the formation of the visual indication allows for the recursion to be applied by using the graph notation.

13. A system (100) as claimed in Claim 12 wherein the recursive graphical query language interpreter (30) is further for generating a plurality of recursive rules from the concepts and the properties in the recursive query graph, characterized in that in the generation of the recursive rules, the recursive graph is iterated through to identify direct and indirect links, a plurality of data is formed from substantially all the properties with the direct links, the iteration of the recursive graph and the data forming are repeated if there are relatively more properties present, and a plurality of recursive heuristics is formed based on the formed data; for generating the internal representation using the recursive heuristics, in which the recursive heuristics are allowed to be applied on other concepts to form diverse queries; and for forming a recursive query using the recursive rules.

14. A system (100) as claimed in Claim 13 wherein the ontology loader (10) is further for selecting an ontology to be loaded by the user (60), and for populating the concepts and the properties of the selected ontology for further processing; characterized in that the system (100) further comprises a graphical user interface having graphical elements for receiving input from the user (60) in the form of a user interaction, and the graphical user interface is for displaying the output of the system (100) in the form of a plurality of result graphs, readable text form, or both; and wherein in the recursive query constructor
component (20), the visual query graph is formed by interacting with a list of the concepts and the properties in the ontology; characterized in that the recursive query constructor component (20) is for searching the list of concepts, selecting a concept from the list of concepts, for forming a node in the visual query graph, for generating a query snippet based on the selected concept, executing the query snippet, and filtering substantially all the properties that are attached to the selected concept in the list of properties.

15. A system (100) as claimed in Claim 14 wherein the query processor component (40) is for sending the internal representation to a query processor component (40) to retrieve the results, in which a plurality of connections is established with a knowledge base (70), a query statement which is generated based on the recursive query is executed in a knowledge base processor, and retrieving the results; and for processing the results to generate a plurality of graphs of a plurality of answers or textual representations; and the recursive results visualizer component (50) is for forming a visual representation of the answers to display the results to the user.
Load the ontology (ontology loader)

Construct query graph by interacting with the concepts and properties in the ontology (recursive query constructor)

Convert the query graph into a recursive query graph (recursive query constructor)

Automatic mapping of the query graph into syntactically valid internal structures (recursive graphical query language interpreter)

Execute the automatically generated query statement and retrieve the results (query processor)

Process the results to generate readable textual answer forms (query processor)

Create visual representation of the answers (recursive results visualizer)

FIG. 1
Start

Identify the recursive property in the query graph.

Create a visual clue indicating an indirect link from the node with the recursive property to the origin node to establish recursive relationships between nodes.

Add more query elements?
Yes

Link the concepts/properties of the additional query elements to the query graph.

No

The query graph is interpreted to create a recursive program.

The recursive program is sent to the query processor for execution.

End

FIG. 4
Start

Iterate through the graph to identify direct and indirect links

Create a plurality of data from all the properties with direct links:

More properties present?

Yes

Create recursive heuristics based on the constructed data

Generate program structures handling recursion using the created heuristics

No

End

FIG. 5
A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/28(2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/28; G06F 3/00; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: visual, query, ontology

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>DNF AWANG ISKANDAR 'Visual Ontology Query Language/ In:First International</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See abstract and sections 1-6.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>FADHIL AMINEH et al. 'OntoQL: A Graphical Query Language for OWL Ontologies.</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>' In: 2007 International Workshop on Description Logics (DL2007), Italy,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8-10 June 2007, Poster session.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See abstract and sections 1-2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See abstract and claims.</td>
<td></td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C.  See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

09 JUNE 2011 (09.06.2011)

Date of mailing of the international search report

21 JUNE 2011 (21.06.2011)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 189 Cheongsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer

LEE, Jong Ick
Telephone No. 82-42-481-8373

Form PCT/ISA/210 (second sheet) (July 2009)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 2009-0187556 A1</td>
<td>23.07.2009</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>