(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 June 2014 (26.06.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/098560 A2

(51) International Patent Classification: *G06F 17/27* (2006.01)

(21) International Application Number: PCT/MY2013/000252

(22) International Filing Date: 12 December 2013 (12.12.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PI 2012701201 17 December 2012 (17.12.2012) MY

(71) Applicant: MIMOS BERHAD [MY/MY]; Technology Park Malaysia, 57000 Kuala Lumpur (MY).

(72) Inventors: BEN MOHAMED, Khalil; c/o Mimos Berhad, Technology Park Malaysia, 57000 Kuala Lumpur (MY). CHU MIN XIAN, Benjamin; c/o Mimos Berhad, Technology Park Malaysia, 57000 Kuala Lumpur (MY). DICKSON, Lukose; c/o Mimos Berhad, Technology Park Malaysia, 57000 Kuala Lumpur (MY). ZAHARI, Fadzly; c/o Mimos Berhad, Technology Park Malaysia, 57000 Kuala Lumpur (MY).

(74) Agent: H.A. RASHID, Ahmad, Fadzlee; IPVOLUSI SDN BHD, A-6-6, Centrio Pantai Hill Park, No.1, Jalan Pantai Murni, Kuala Lumpur 59200 (MY).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *of inventorship (Rule 4.17(iv))*

[Continued on next page]

(54) **Title**: A SYSTEM AND METHOD FOR TRANSFORMING AN ABSTRACT REPRESENTATION TO A LINGUISTIC REPRESENTATION AND VICE VERSA

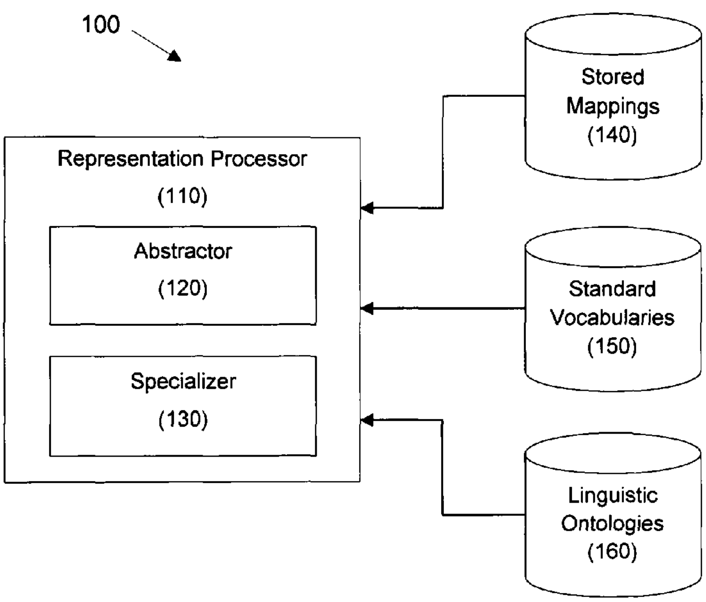

FIG. 1a

(57) **Abstract**: The present invention relates to a system and method for linguistic processing. The system (100) comprises a representation processor (110) having an abstractor component (120) and a specializer component (130), a stored mappings database (140), a standard vocabularies database (150), and a linguistic ontologies database (160). The abstractor component (120) transforms a linguistic representation into an abstract representation by using abstraction rules. The abstractor component (120) includes a concepts and properties extractor component (121), a verb determinator component (122), a schemas extractor component (123), a schemas mapper component (124), and a concepts and properties matcher component (125). The specializer component (120) transforms an abstract representation into a linguistic representation by using specialization rules. The specializer component (130) includes a triple extractor component (131), a property matcher component (132), a verb determinator component (133), a verb mapper component (134), a semantic roles mapper component (135) and a triple assembler component (136).

WO 2014/098560 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

A SYSTEM AND METHOD FOR TRANSFORMING AN ABSTRACT REPRESENTATION TO A LINGUISTIC REPRESENTATION AND VICE VERSA

FIELD OF INVENTION

The present invention relates to a system and method for linguistic processing. More particularly, the present invention relates to a system and method for transforming linguistic representation to abstract representation and vice versa.

BACKGROUND OF THE INVENTION

Current computational approaches to text processing are able to produce complex linguistic structures. These linguistic structures represent the meaning of the natural language text; but that does not mean that the structure can be easily used. Linguistic structure in its broader context should be concerned with making real world references to convey, process, and assign meaning, as well as to manage and resolve ambiguity.

An example of an approach to produce the linguistic structures is disclosed by US Patent No. 7,464,026 B2 which relates to a system and method for performing semantic analysis that interprets a linguistic structure output by a natural language linguistic analysis system. The semantic analysis system converts the linguistic output by the natural language linguistic analysis system into a data structure model referred to as a semantic discourse representation structure (SemDRS).

However, such approach does not use schema extracted from standard vocabularies and linguistic ontologies. Instead, it uses application schema which is dependent on each application. Furthermore, the transformation rules are manually built for each specific application schema.

Therefore, there is a need to provide a system and method for linguistic processing that is able to transform linguistic representation to abstract representation and vice versa.

SUMMARY OF INVENTION

In a first aspect of the present invention, a system (**100**) for transforming an abstract representation to a linguistic representation and/or vice versa comprises a representation processor (**110**), a stored mappings database (**140**), a standard vocabularies database (**150**), and a linguistic ontologies database (**160**). The representation processor (**110**) includes an abstractor component (**120**), wherein the abstractor component (**120**) is used to transform a linguistic representation into an abstract representation by using abstraction rules; and a specializer component (**130**), wherein the specializer component (**120**) is used to transform an abstract representation into a linguistic representation by using specialization rules.

Preferably, the abstractor component (**120**) comprises a concepts and properties extractor component (**121**), wherein the concepts and properties extractor component (**121**) is used to extract all the concepts, properties and linguistic tags based on the input of a linguistic representation; a verb determinator component (**122**), wherein the verb determinator component (**122**) is used to determine whether a concept is a verb using the linguistic ontologies database (**160**); a schemas extractor component (**123**), wherein the schemas extractor component (**123**) is used to extract a set of schemas from the standard vocabularies (**150**) and the linguistic ontologies (**160**) databases; a schemas mapper component (**124**), wherein the schemas mapper component (**124**) is used to map a concept to the extracted schemas from the schemas extractor component (**123**); and a concepts and properties matcher component (**125**), wherein the concepts and properties matcher component (**125**) is used to process all the unprocessed concepts and properties based on the abstraction rules.

Preferably, the specializer component (**130**) comprises a triple extractor component (**131**), wherein the triple extractor component (**131**) is used to extract all the triples from an abstract representation; a property matcher component (**132**), wherein the property matcher component (**132**) is used to match a property of a triple to a schema in the stored mappings database (**140**) based on the specialization rules; a verb determinator component (**133**), wherein the verb determinator component (**133**) is used to determine whether a property is a verb by using the linguistic ontologies database (**160**); a verb mapper component (**134**), wherein the verb mapper component (**134**) is used to map a property of a triple to a schema in the linguistic ontologies database (**160**); a semantic roles mapper component (**135**), wherein the semantic roles mapper component (**135**) is used to convert a triple or a triple with a schema to a set of triples of a linguistic representation by using the linguistic ontologies database (**160**); and a triple assembler component (**136**), wherein the triples assembler component (**136**) is used to perform maximal join of all possible linguistic representation triples.

In a second aspect of the present invention, a method for transforming a linguistic representation to an abstract representation is provided. The method is characterised by the steps of receiving a linguistic representation as an input; extracting all the concepts, properties, linguistic tags of the linguistic representation by a concepts and properties extractor component (**121**); determining whether the extracted concepts in the set are empty; sending the extracted concepts to a verb determinator component (**122**) if the extracted concepts in the set are not empty; determining whether the concept is a verb by the verb determinator component (**122**); sending the concept to a schemas mapper component (**124**); mapping the concept to a schema provided by a schemas extractor component (**123**); storing the mapped schemas in a stored mappings database (**140**); sending a set of triples with unprocessed concept and property from the schema mapper component (**124**) to a concepts and properties matcher component (**125**); and processing the set of triples based on an abstraction rules by the concepts and properties matcher component (**125**).

Preferably, the step of determining whether the concept is a verb by the verb determinator component (**122**) includes the steps of determining whether the concept is a verb by using the linguistic ontologies database (**160**); and checking whether the concept with its linked properties maps to one of the available schemas for this verb by using the linguistic ontologies database (**160**).

In a third aspect of the present invention, a method for transforming an abstract representation to a linguistic representation is provided. The method is characterised by the steps of receiving an abstract representation as an input; extracting all the triples from the abstract representation by a triple extractor component (**131**) to produce a set of triples; determining whether the triples in the set are empty by the triple extractor component (**131**); sending the triples to a property matcher component (**132**) if the triples in the set are not empty; matching the property of the triples with a schema in a stored mappings database (**140**) and specialization rules by the property matcher component (**132**); sending the schema and the triples to a semantic roles mapper component (**135**) and proceeding to step (m) if there is a match between the property of the triples with a schema in the stored mappings database (**140**) and the specialization rules; sending the triples to a verb determinator component (**133**) if there is no match between the property of the triples with a schema in the stored mappings database (**140**) and the specialization rules; determining whether the property of the triples is a verb by the verb determinator component (**133**); sending the triples to the semantic roles mapper component (**135**) and proceeding to step (m) if the property of the triples is a verb; sending the triples to a verb mapper component (**134**) if the property of the triples is not a verb; mapping the property of the triple to a schema in a linguistic ontologies database (**160**) by the verb mapper component (**134**); sending the triples and the schema to the semantic roles mapper component (**135**); and converting the triples and/or the schema to a set of triples of a linguistic representation by the semantic roles mapper component (**135**).

Preferably, the method includes the step of sending the triples to a triples assembler component (**136**) and performing maximal join of all possible linguistic representation triples by the triples assembler component (**136**) if the triples in the set are empty.

Preferably, the step of converting the triples and/or the schema to a set of triples of a linguistic representation includes finding a suitable concept type hierarchy and/or schema in the linguistic ontologies database (**160**) and mapping the concepts of the triples with the suitable concept type hierarchy and/or schema.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

FIGS. 1(a-c) show block diagrams of a system (**100**) to transform an abstract representation to a linguistic representation and vice versa according to an embodiment of the present invention.

FIG. 2 shows a flowchart of a method for transforming linguistic representation to abstract representation according to an embodiment of the present invention.

FIG. 3 shows a flowchart of a method for transforming abstract representation to linguistic representation according to an embodiment of the present invention.

DESCRIPTION OF THE PREFFERED EMBODIMENT

A preferred embodiment of the present invention will be described herein below with reference to the accompanying drawings. In the following description, well known functions or constructions are not described in detail since they would obscure the description with unnecessary detail.

Referring to **FIG. 1a**, there is shown a block diagram of a system (**100**) to transform an abstract representation to a linguistic representation and vice versa according to an embodiment of the present invention. The system (**100**) is able to transform a linguistic representation into an abstract representation. Moreover, the system (**100**) is able to transform an abstract representation into a linguistic representation. The linguistic representation refers to a structure representing meaning of a natural language text in terms of an entity-and-relation model of a non-linguistic domain. As an example, a natural language text "*John writes a book with a quill*" is provided as a linguistic representation below:

[Person:John]<-(agnt)<-[write]->(thme)->[Book]

->(inst)->[Quill]

On the other hand, the abstract representation is a more high-level structure compared to the linguistic representation, containing standardized terms, easier to read by a human but still processable by a computer. An example of an abstract representation of the above linguistic representation is provided below:

[Person:John] ->(dc:author)->[Book]

->(use)->[Quill]

The system (**100**) comprises of a representation processor (**110**), a stored mappings database (**140**), a standard vocabularies database (**150**) and a linguistic ontologies database (**160**).

The representation processor (**110**) includes an abstractor component (**120**) and a specializer component (**130**). The representation processor (**110**) is used to receive a linguistic representation and an abstract representation and transform it into an abstract representation and a linguistic representation respectively by using predefined abstraction rules and specialization rules.

The stored mappings database (**140**) is used to store mapped schemas. As an example of a stored mapping is provided below:
[Author]<-(agnt)<-[write]->(thme)->[Text] maps to [Author]->(dc:author)->[Text].

The standard vocabularies database (**150**) is used to describe relationships and concepts of a term. Such standard vocabularies database includes but not limited to Friend of a Friend (FOAF) knowledge base.

The linguistic ontologies database (**160**) used preferably include VerbNet and FrameNet, wherein VerbNet is lexical database of verb and FrameNet is a lexical database.

FIG. 1b shows a block diagram of the abstractor component (**120**) of the representation processor (**110**). The abstractor component (**120**) is used to transform a linguistic representation into an abstract representation by using the predefined abstraction rules. The abstractor component (**120**) comprises of a concepts and properties extractor component (**121**), a verb determinator component (**122**), a schemas extractor component (**123**), a schemas mapper component (**124**), and a concepts and properties matcher component (**125**).

The concepts and properties extractor component (**121**) is used to extract all the concepts and properties based on the input of a linguistic representation. Moreover, the concepts and properties extractor component (**121**) is also used to identify linguistic tags for the concepts identified. Thus, the concepts and properties extractor component (**121**) produces a set of concepts, properties and linguistic tags based on the linguistic representation provided.

The verb determinator component (**122**) is used to determine whether a concept is a verb or not by using the linguistic ontologies database (**160**). In particular, the verb determinator component (**122**) uses VerbNet of the linguistic ontologies database (**160**) to determine whether the concept is a verb and thereon, the verb determinator component (**122**) uses FrameNet to check whether the concept with its linked properties maps to one of the available schemas for this verb.

The schemas extractor component (**123**) is used to extract a set of schemas from the standard vocabularies (**150**) and the linguistic ontologies (**160**) databases. In particular, the schemas extractor component (**123**) extracts the associated schemas for a verb from FrameNet. Moreover, the schemas extractor component (**123**) keeps track of relationships between schemas from FrameNet and schemas from the standard vocabularies database (**150**).

The schemas mapper component (**124**) is used to map a concept to the extracted schemas from the schemas extractor component (**123**) to produce a set of triples with unprocessed concept and property. As an example, where the concept is "write" from the linguistic representation of *"[Person:John]<-(agnt)<-[write]->(thme)->[Book]"* and the extracted schema from FrameNet is provided as *"[Author]<-(agnt)<-[write]->(thme)->[Text]"* while the extracted schema from the standard vocabularies database is provided as *"[Author]->(dc:author)->[Text]"*; the schemas mapper component (**124**) maps "*write*" from the linguistic representation to "*write*" from the extracted schema from FrameNet by determining whether "*Person:John*" can be mapped to "*Author*", "*Book*" can be mapped to "*Text*", "*agnt*" can be mapped to "*agnt*" and "*thme*" can be mapped to "*thme*" by using the concept hierarchy of the linguistic ontologies database (**160**). Thereon, the schemas mapper component (**124**) determines whether there is a relationship between the extracted schema from FrameNet and the extracted schema from the standard vocabularies database (**150**), wherein the concept "*write*" of the linguistic representation is mapped to the extracted schema from the standard vocabularies database (**150**) if there is a relationship between the two extracted schemas. Thus, the schemas mapper component (**124**) produces the triple as "*Person:John dc:author Book*".

The concepts and properties matcher component (**125**) is used to process all the unprocessed concepts and properties based on predefined abstraction rules. For example, the unprocessed concept and property of the linguistic representation of "*[Person:John]<-(agnt)<-[write]->(thme)->[Book] ->(inst)->[Quill]*" are provided as "*inst*" and "*Quill*" as "*Quill*" is not a verb and the schemas of "*write*" do not have the property of "*inst*" to be matched to. Thus, the concepts and properties matcher component (**125**) processes "*inst*" and "*Quill*" by using the abstraction rule of "*if C1 is a verb and there exists C2 such that [C2] (agnt) [C1] AND [C1] (inst) [C3] exists, THEN C2 use C3.*"

FIG. 1c shows a block diagram of the specializer component (**130**) of the representation processor (**110**). The specializer component (**120**) is used to transform an abstract representation into a linguistic representation by using the predefined specialization rules. The specializer component (**130**) includes a triple extractor component (**131**), a property matcher component (**132**), a verb determinator component (**133**), a verb mapper component (**134**), a semantic roles mapper component (**135**) and a triple assembler component (**136**).

The triple extractor component (**131**) is used to extract all the triples from an abstract representation provided as an input.

The property matcher component (**132**) is used to match a property of a triple to a schema in the stored mappings database (**140**) and predefined specialization rules. As an example, where a property for a triple "*Person:John dc:author Book*" is provided as "*dc:author*", a schema in the stored mappings database (**140**) is mapped to the property, wherein "*[Author]<-(agnt)<-[write]->(thme)->[Text]*" is mapped to "*[Author]->(dc:author)->[Text]*". Thereon, the triple is matched to a hypothesis of the predefined specialization rules.

The verb determinator component (**133**) is used to determine whether a property is a verb or not by using the linguistic ontologies database (**160**). In particular, the verb determinator component (**133**) uses VerbNet of the linguistic ontologies database (**160**) to determine whether the concept is a verb and thereon, the verb determinator component (**133**) uses FrameNet to determine whether the concept with its linked properties maps to one of the available schemas for this verb. As an example for the linguistic representation of "*[Person:John]<-(agnt)<-[write]->(thme)->[Book]*", the verb determinator component (**133**) determines whether "*write*" is a verb and since it is a verb, the verb determinator component (**133**) determines the available schemas for "*write*" that resulted to a schema of "*[Author]<-(agnt)<-[write]->(thme)->[Text]*".

The verb mapper component (**134**) is used to map a property of a triple to a schema in the linguistic ontologies database (**160**). For example, if the property of a tripe is provided as "*dc:author*", the verb mapper component determine whether the label which is "*author*" is a verb or not, wherein the property is mapped to a schema if the label is a verb or transforming the label to the closest similar verb if the label is not a verb.

The semantic roles mapper component (**135**) is used to convert a triple or a triple with a schema to a set of triples of a linguistic representation. The conversion is done by using the linguistic ontologies database (**160**) to find a suitable concept type hierarchy and/or schema to map with the concepts of the triples. For instance, where a triple is provided as "*Person:John write Book*", the semantic roles mapper component (**135**) extracts semantic roles for the verb "*write*" from FrameNet which results to the semantic roles of "*Animate*" and "*Resource*". Thereon, the semantic roles mapper component (**135**) maps "*Animate*" to "*Person:John*" and "*Resource*" to "*Book*". As a result, the semantic roles mapper component (**135**) outputs "*Person:John agnt Write*" and "*Write thme Book*".

The triples assembler component (**136**) is used to perform maximal join of all possible linguistic representation triples. As an example, the triples assembler component (**136**) performs a maximal join on an input of "*[Person:John]-(agnt)-[Write]*" and "*[Write]-(thme)-[Book]*" to result in the output of "*[Person:John]-(agnt)-[Write]-(thme)-[Book]*".

Referring to **FIG. 2**, there is shown a flowchart of a method for transforming a linguistic representation to an abstract representation by using the system (**100**) of **FIG. 1**. Initially, the concepts and properties extractor component (**121**) receives a linguistic representation as an input, wherein the linguistic representation is a conceptual graph comprising of concepts and properties.

The concepts and properties extractor component (**121**) extracts all the concepts and properties of the linguistic representation as in step **201**. For a linguistic representation provided as *"[Person:John]<-(agnt)<-[write]->(thme)->[Book] ->(inst)->[Quill]"*, the extracted concepts include "*Person:John*", "*write*", "*Book*" and "*Quill*", the extracted properties include "*agnt*", "*thme*" and "*inst*". In addition to that, the concepts and properties extractor component (**121**) also identifies linguistic tags for the extracted concepts based on the extracted properties.

Thereon, the concepts and properties extractor component (**121**) determines whether the extracted concepts in the set are empty as in decision **202**. If the extracted concepts are empty, the method ends.

Otherwise, the extracted concepts are sent to the verb determinator component (**122**) to determine whether the concept is a verb as in step **203**. The verb determinator component (**122**) determines whether the concept is a verb by using the linguistic ontologies database (**160**). In particular, the verb determinator component (**122**) uses VerbNet of the linguistic ontologies database (**160**) to determine whether the concept is a verb and thereon, the verb determinator component (**122**) uses FrameNet to check whether the concept with its linked properties maps to one of the available schemas for this verb. As an example for the linguistic representation of "*[Person:John]<-(agnt)<-[write]->(thme)->[Book]*", the verb determinator component (**122**) determines whether "*write*" is a verb and since it is a verb, the verb determinator component (**122**) determines the available schemas for "*write*" that resulted to a schema of "*[Author]<-(agnt)<-[write]->(thme)->[Text]*".

In decision **204**, if the concept is not a verb, the method returns to decision **202** wherein the concepts and properties extractor component (**121**) determines whether the extracted concepts in the set are empty. However, if the concept is a verb as in decision **204**, the concept is sent to the schemas mapper component (**124**).

The schemas mapper component (**124**) maps the received concept to a schema provided by the schemas extractor component (**123**) as in step **205**. Thereon, the schema mapper component (**124**) stores all schemas that have been mapped with the concept received from the verb determinator component (**122**). The mapped schemas are stored in the stored mappings database (**140**). Thus, the schema mapper component (**124**) produces a set of triples with unprocessed concept and property which is sent to the concepts and properties matcher component (**125**). As an example, where the concept is "write" from the linguistic representation of *"[Person:John]<-(agnt)<-[write]->(thme)->[Book]"* and the schemas extractor component (**123**) provides a schema extracted from FrameNet as *"[Author]<-(agnt)<-[write]->(thme)->[Text]"* while a schema extracted from the standard vocabularies database as "*[Author]->(dc:author)->[Text]*"; the schemas mapper component (**124**) maps "*write*" from the linguistic representation to "*write*" from the extracted schema from FrameNet by determining whether "*Person:John*" can be mapped to "*Author*", "*Book*" can be mapped to "*Text*", "*agnt*" can be mapped to "*agnt*" and "*thme*" can be mapped to "*thme*" by using the concept hierarchy of the linguistic ontologies database (**160**). Thereon, the schemas mapper component (**124**) determines whether there is a relationship between the extracted schema from FrameNet and the extracted schema from the standard vocabularies database (**150**), wherein the concept "*write*" of the linguistic representation is mapped to the extracted schema from the standard vocabularies database (**150**) if there is a relationship between the two extracted schemas. The schemas mapper component (**124**) produces the triple as "*Person:John dc:author Book*".

In step **206**, the concepts and properties matcher component (**125**) processes the set of triples from the schema mapper component (**124**) by using the predefined abstraction rules. As a result, the concepts and properties matcher component (**125**) produces an abstract representation of the linguistic representation provided as the input. As an example, concepts and properties matcher component (**125**) processes the triple of "*Person:John dc:author Book*" with unprocessed concept and property of of "*inst*" and "*Quill*" by using the abstraction rule below:

if C1 is a verb and there exists C2 such that [C2] (agnt) [C1] AND [C1] (inst) [C3] exists, THEN C2 use C3.

As a result, the concepts and properties matcher component (**125**) produces "*John dc:author Book*" and "*John use Quill*" as the abstract representations.

FIG. 3 shows a flowchart of a method for transforming an abstract representation to a linguistic representation by using the system (**100**) of **FIG. 1**. Initially, the triple extractor component (**131**) receives an abstract representation such as a Resource Description Framework (RDF) document. Thereon, the triple extractor component (**131**) extracts all the triples from the input as in step **301**. Thus, the triple extractor component (**131**) produces a set of triples based on the abstract representation provided as the input.

In decision **302**, the triple extractor component (**131**) determines whether the triples in the set are empty. If the triples are empty, the triples are sent to the triples assembler component (**136**). Thereon, the triples assembler component (**136**) performs maximal join of all possible linguistic representation triples as in step **303**. As an example, the triples assembler component (**136**) performs a maximal join on a set of triples of "*[Person:John]-(agnt)-[Write]*" and "*[Write]-(thme)-[Book]*" that resulted to an output of "*[Person:John]-(agnt)-[Write]-(thme)-[Book]*".

However, if the triples in the set are not empty, the triples are sent to the property matcher component (**132**). The property matcher component (**132**) matches the property of the triples with a schema in the stored mappings database (**140**) and predefined specialization rules as in step **304**. As an example, if the triple is provided as "*Person:John dc:author Book*", the property matcher component (**132**) matches "dc:author" to a schema in the stored mappings database (**140**), wherein "*[Author]<-(agnt)<-[write]->(thme)->[Text]*" is mapped to "*[Author]->(dc:author)->[Text]*". Thereon, the triple is matched to a hypothesis of the predefined specialization rules.

In decision **305**, if there is a match between the property of the triples with a schema in the stored mappings database and predefined specialization rules, the schema and the triples are sent to the semantic roles mapper component (**135**). The semantic roles mapper component (**135**) converts the triples and the schema to a set of triples of a linguistic representation as in step **306**. The conversion is done by using the linguistic ontologies database (**160**) to find a suitable concept type hierarchy to map with the concepts of the triples. As an example, the semantic roles mapper component (**135**) converts a triple provided as "*Person:John write Book*" to a set of triples of "*Person:John agnt Write*" and "*Write thme Book*" by extracting semantic roles for the verb “write” from FrameNet that resulted to the semantic roles of “*Animate*” and “*Resource*”, and mapping “*Animate*” to “*Person:John*” and “*Resource*” to “*Book*”. Once the semantic roles mapper component (**135**) has converted the triples and schema to a set of triples of linguistic representation, the method returns to decision **302**.

If there is no match between the property of the triples and a schema in the stored mappings database (**140**) in decision **305**, the triples is sent to the verb determinator component (**133**). The verb determinator component (**133**) determines whether the property of the triples is a verb by using the linguistic ontologies database (**160**) as in step **307**.

If the property of the triples is a verb as in decision **308**, the verb determinator sends the triples to the semantic roles mapper component (**135**). Thereon, the semantic roles mapper component (**135**) converts the triples to a set of triples of a linguistic representation as in step **306**. The conversion is done by using the linguistic ontologies database (**160**) to find a suitable concept type hierarchy and schema to map with the concepts of the triples. Thereon, the method returns to decision **302**.

If the property of the triples is not a verb as in decision **308**, the verb determinator component (**133**) sends the triples to the verb mapper component (**134**). Thereon, the verb mapper component (**134**) maps the property of the triple to a schema in the linguistic ontologies database (**160**) as in step **309**. The triples and the schema are then sent to semantic roles mapper component (**135**).

The semantic roles mapper component (**135**) converts the triples and schema to a set of triples of a linguistic representation as in step **306**. The conversion is done by using the linguistic ontologies database (**160**) to find a suitable concept type hierarchy to map with the concepts of the triples. As a result, the semantic roles mapper component (**135**) produces a set of triples based on the abstract representation provided as the input.

While embodiments of the invention have been illustrated and described, it is not intended that these embodiments illustrate and describe all possible forms of the invention. Rather, the words used in the specifications are words of description rather than limitation and various changes may be made without departing from the scope of the invention.

CLAIMS

1. A system (**100**) for transforming an abstract representation to a linguistic representation and/or vice versa comprising:
   a) a representation processor (**110**),
   b) a stored mappings database (**140**),
   c) a standard vocabularies database (**150**), and
   d) a linguistic ontologies database (**160**);

   wherein said system (**100**) is characterised in that said representation processor (**110**) includes:
   e) an abstractor component (**120**), wherein said abstractor component (**120**) is used to transform a linguistic representation into an abstract representation by using abstraction rules; and
   f) a specializer component (**130**), wherein said specializer component (**120**) is used to transform an abstract representation into a linguistic representation by using specialization rules.

2. The system (**100**) as claimed in claim **1**, wherein said abstractor component (**120**) comprising:
   a) a concepts and properties extractor component (**121**), wherein said concepts and properties extractor component (**121**) is used to extract all the concepts, properties and linguistic tags based on the input of a linguistic representation;
   b) a verb determinator component (**122**), wherein said verb determinator component (**122**) is used to determine whether a concept is a verb using said linguistic ontologies database (**160**);
   c) a schemas extractor component (**123**), wherein said schemas extractor component (**123**) is used to extract a set of schemas from said standard vocabularies (**150**) and said linguistic ontologies (**160**) databases;
   d) a schemas mapper component (**124**), wherein said schemas mapper component (**124**) is used to map a concept to the extracted schemas from said schemas extractor component (**123**); and
   e) a concepts and properties matcher component (**125**), wherein said concepts and properties matcher component (**125**) is used to process

all the unprocessed concepts and properties based on the abstraction rules.

3. The system (**100**) as claimed in claim **1**, wherein said specializer component (**130**) comprising:
   a) a triple extractor component (**131**), wherein said triple extractor component (**131**) is used to extract all the triples from an abstract representation;
   b) a property matcher component (**132**), wherein said property matcher component (**132**) is used to match a property of a triple to a schema in said stored mappings database (**140**) based on the specialization rules;
   c) a verb determinator component (**133**), wherein said verb determinator component (**133**) is used to determine whether a property is a verb by using said linguistic ontologies database (**160**);
   d) a verb mapper component (**134**), wherein said verb mapper component (**134**) is used to map a property of a triple to a schema in said linguistic ontologies database (**160**);
   e) a semantic roles mapper component (**135**), wherein said semantic roles mapper component (**135**) is used to convert a triple or a triple with a schema to a set of triples of a linguistic representation by using said linguistic ontologies database (**160**); and
   f) a triple assembler component (**136**), wherein said triples assembler component (**136**) is used to perform maximal join of all possible linguistic representation triples.

4. A method for transforming a linguistic representation to an abstract representation by using the system (**100**) as claimed in claim **2**, is characterised by the steps of:
   a) receiving a linguistic representation as an input;
   b) extracting all the concepts, properties, linguistic tags of the linguistic representation by a concepts and properties extractor component (**121**);
   c) determining whether the extracted concepts in the set are empty;

d) sending the extracted concepts to a verb determinator component (**122**) if the extracted concepts in the set are not empty;
e) determining whether the concept is a verb by the verb determinator component (**122**);
f) sending the concept to a schemas mapper component (**124**);
g) mapping the concept to a schema provided by a schemas extractor component (**123**);
h) storing the mapped schemas in a stored mappings database (**140**);
i) sending a set of triples with unprocessed concept and property from the schema mapper component (**124**) to a concepts and properties matcher component (**125**); and
j) processing the set of triples based on an abstraction rules by the concepts and properties matcher component (**125**).

5. The method as claimed in claim **4**, wherein the step of determining whether the concept is a verb by the verb determinator component (**122**) includes the steps of:
a) determining whether the concept is a verb by using the linguistic ontologies database (**160**); and
b) checking whether the concept with its linked properties maps to one of the available schemas for this verb by using the linguistic ontologies database (**160**).

6. A method for transforming an abstract representation to a linguistic representation by using the system (**100**) as claimed in claim **3**, is characterised by the steps of:
a) receiving an abstract representation as an input;
b) extracting all the triples from the abstract representation by a triple extractor component (**131**) to produce a set of triples;
c) determining whether the triples in the set are empty by the triple extractor component (**131**);
d) sending the triples to a property matcher component (**132**) if the triples in the set are not empty;

e) matching the property of the triples with a schema in a stored mappings database (**140**) and specialization rules by the property matcher component (**132**);

f) sending the schema and the triples to a semantic roles mapper component (**135**) and proceeding to step (m) if there is a match between the property of the triples with a schema in the stored mappings database (**140**) and the specialization rules;

g) sending the triples to a verb determinator component (**133**) if there is no match between the property of the triples with a schema in the stored mappings database (**140**) and the specialization rules;

h) determining whether the property of the triples is a verb by the verb determinator component (**133**);

i) sending the triples to the semantic roles mapper component (**135**) and proceeding to step (m) if the property of the triples is a verb;

j) sending the triples to a verb mapper component (**134**) if the property of the triples is not a verb;

k) mapping the property of the triple to a schema in a linguistic ontologies database (**160**) by the verb mapper component (**134**);

l) sending the triples and the schema to the semantic roles mapper component (**135**); and

m) converting the triples and/or the schema to a set of triples of a linguistic representation by the semantic roles mapper component (**135**).

7. The method as claimed in claim **6**, wherein the method includes the step of sending the triples to a triples assembler component (**136**) and performing maximal join of all possible linguistic representation triples by the triples assembler component (**136**) if the triples in the set are empty.

8. The method as claimed in claim **6**, wherein the step of converting the triples and/or the schema to a set of triples of a linguistic representation includes finding a suitable concept type hierarchy and/or schema in the linguistic ontologies database (**160**) and mapping the concepts of the triples with the suitable concept type hierarchy and/or schema.

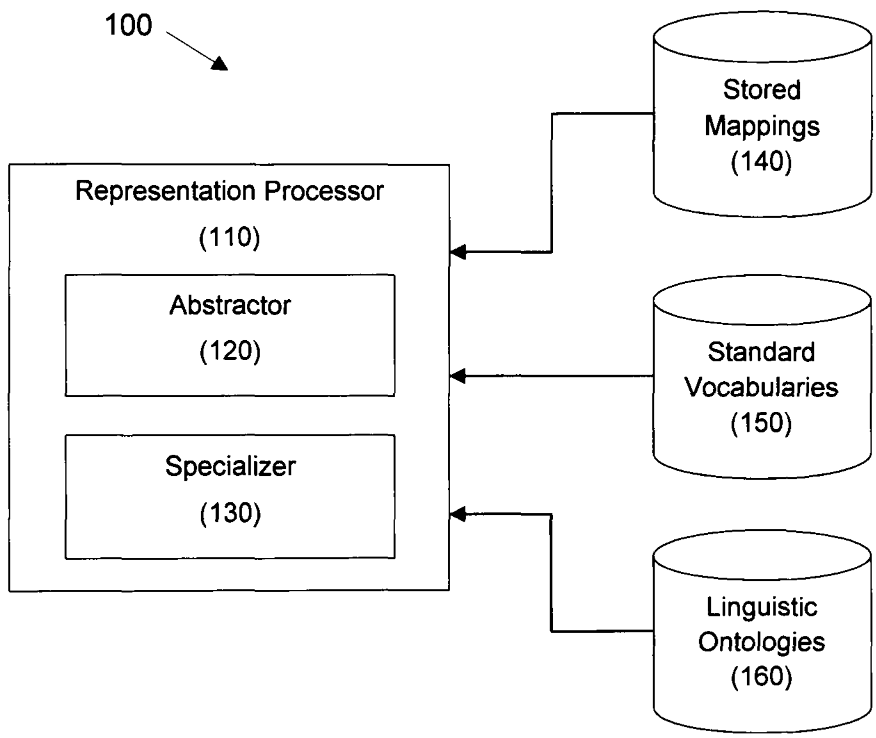
100
Representation Processor
(110)
Abstractor
(120)
Specializer
(130)
Stored
Mappings
(140)
Standard
Vocabularies
(150)
Linguistic
Ontologies
(160)

FIG. 1a

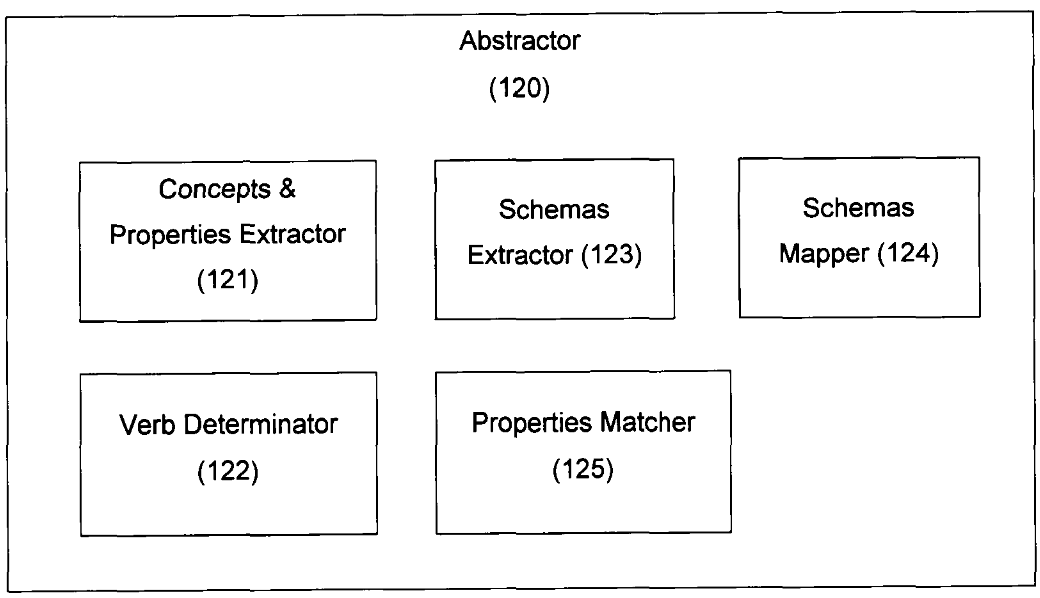
Abstractor
(120)
Concepts &
Properties Extractor
(121)
Schemas
Extractor (123)
Schemas
Mapper (124)
Verb Determinator
(122)
Properties Matcher
(125)

FIG. 1b

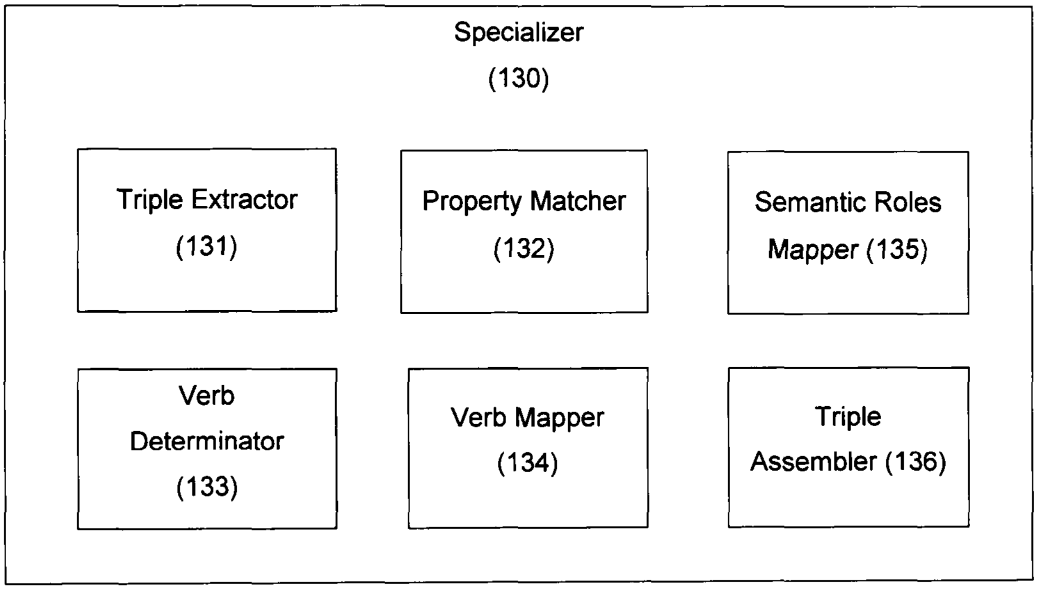
Specializer
(130)
Triple Extractor
(131)
Property Matcher
(132)
Semantic Roles
Mapper (135)
Verb
Determinator
(133)
Verb Mapper
(134)
Triple
Assembler (136)

FIG. 1c

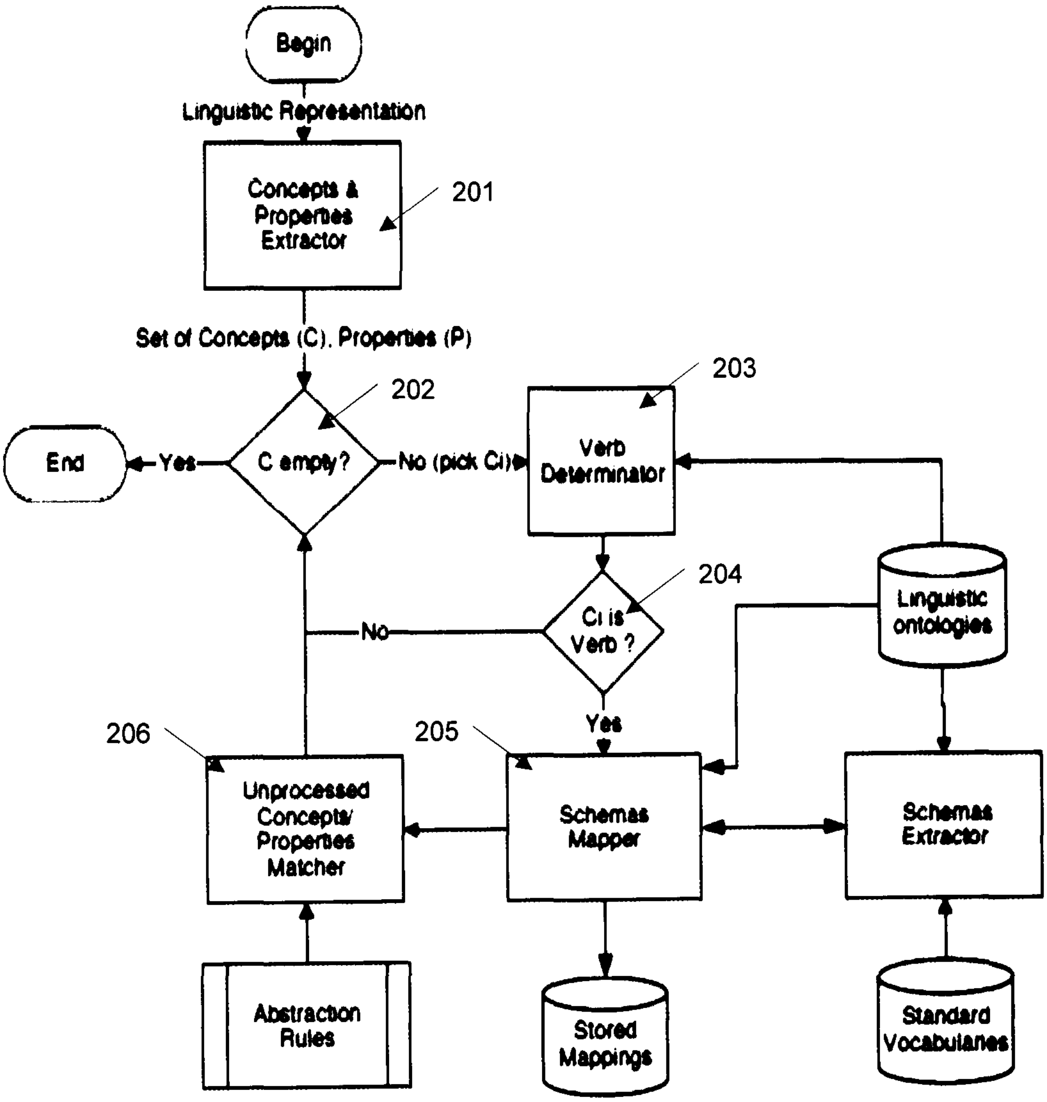
Begin
Linguistic Representation
Concepts & Properties Extractor
201
Set of Concepts (C), Properties (P)
202
C empty?
Yes
End
No (pick Ci)
203
Verb Determinator
204
Ci is Verb ?
No
Yes
Linguistic ontologies
206
Unprocessed Concepts/ Properties Matcher
205
Schemas Mapper
Schemas Extractor
Abstraction Rules
Stored Mappings
Standard Vocabularies

FIG. 2

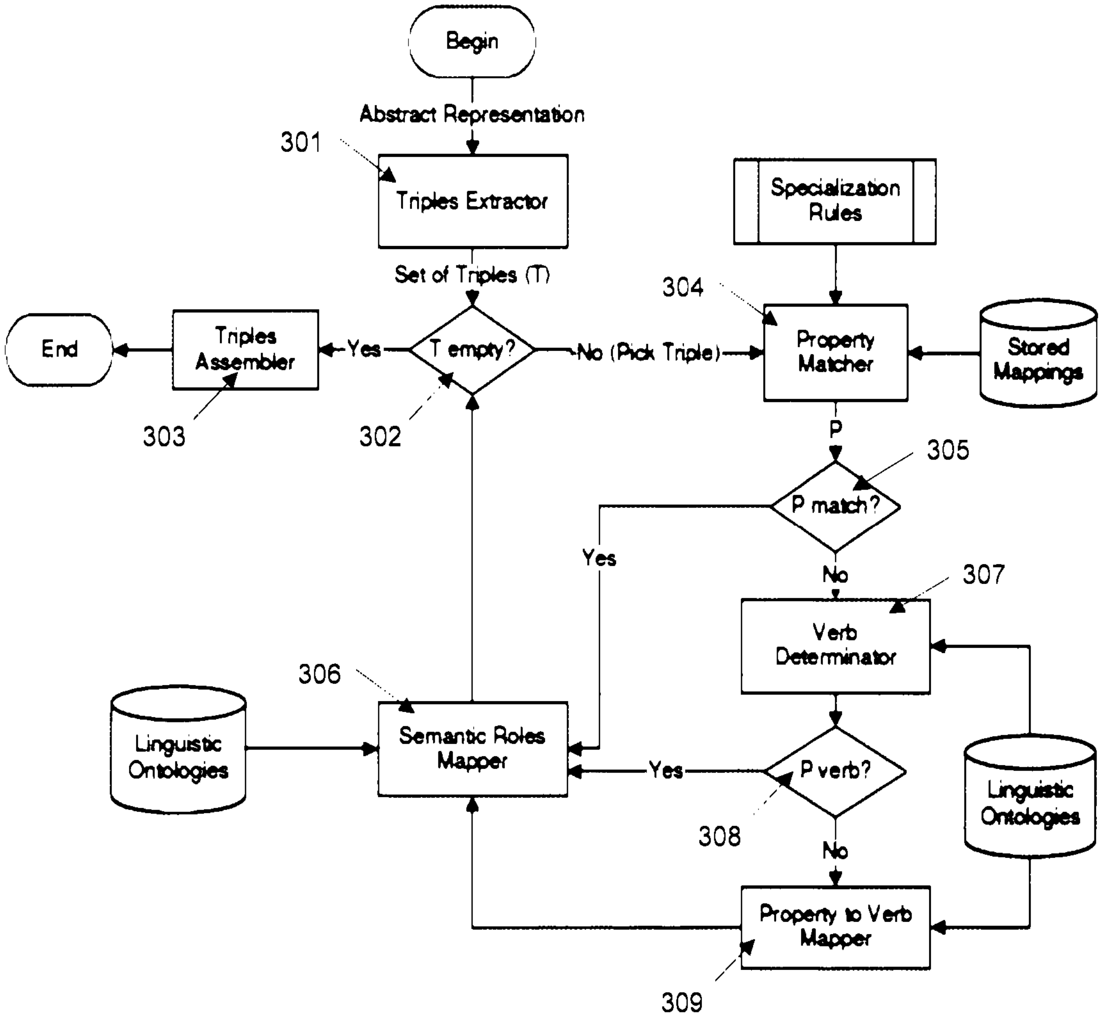
Begin
Abstract Representation
301
Triples Extractor
Set of Triples (T)
Specialization Rules
304
End
Triples Assembler
Yes
T empty?
No (Pick Triple)
Property Matcher
Stored Mappings
303
302
P
305
P match?
Yes
No
307
Verb Determinator
306
Linguistic Ontologies
Semantic Roles Mapper
Yes
P verb?
Linguistic Ontologies
308
No
Property to Verb Mapper
309

FIG. 3