EXHIBIT A

US007177798B2

(12) **United States Patent**
Hsu et al.

(10) **Patent No.: US 7,177,798 B2**
(45) **Date of Patent: Feb. 13, 2007**

(54) **NATURAL LANGUAGE INTERFACE USING CONSTRAINED INTERMEDIATE DICTIONARY OF RESULTS**

(75) Inventors: **Cheng Hsu**, Latham, NY (US); **Veera Boonjing**, Troy, NY (US)

(73) Assignee: **Rensselaer Polytechnic Institute**, Troy, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 578 days.

(21) Appl. No.: **09/861,860**

(22) Filed: **May 21, 2001**

(65) **Prior Publication Data**

US 2002/0059069 A1 May 16, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/544,676, filed on Apr. 17, 2000, now abandoned.

(60) Provisional application No. 60/205,725, filed on May 19, 2000.

(51) **Int. Cl.**
G06F 17/27 (2006.01)

(52) **U.S. Cl.** **704/9**; 704/10; 706/11; 706/45

(58) **Field of Classification Search** 704/270.1, 704/9, 7, 201, 10; 706/45, 11, 927, 55; 707/2–9, 707/104.1; 715/762, 709
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,047,614	A	9/1991	Bianco	
5,057,915	A	10/1991	Von Kohorn	
5,412,804	A *	5/1995	Krishna	707/2
5,412,806	A *	5/1995	Du et al.	707/2
5,418,951	A *	5/1995	Damashek	707/5
5,471,611	A *	11/1995	McGregor	707/4
5,493,677	A *	2/1996	Balogh et al.	707/104.1
5,537,618	A	7/1996	Boulton et al.	
5,578,808	A	11/1996	Taylor	
5,596,994	A	1/1997	Bro	
5,644,727	A	7/1997	Atkins	
5,696,962	A	12/1997	Kupiec	
5,701,400	A *	12/1997	Amado	706/45
5,706,442	A	1/1998	Anderson et al.	
5,710,886	A	1/1998	Christensen et al.	
5,721,827	A	2/1998	Logan et al.	
5,749,081	A	5/1998	Whiteis	

(Continued)

FOREIGN PATENT DOCUMENTS

CH	681 573	A5	4/1993

(Continued)

OTHER PUBLICATIONS

International Search Report from International Patent Application PCT/US00/09265, filed Apr. 7, 2000.

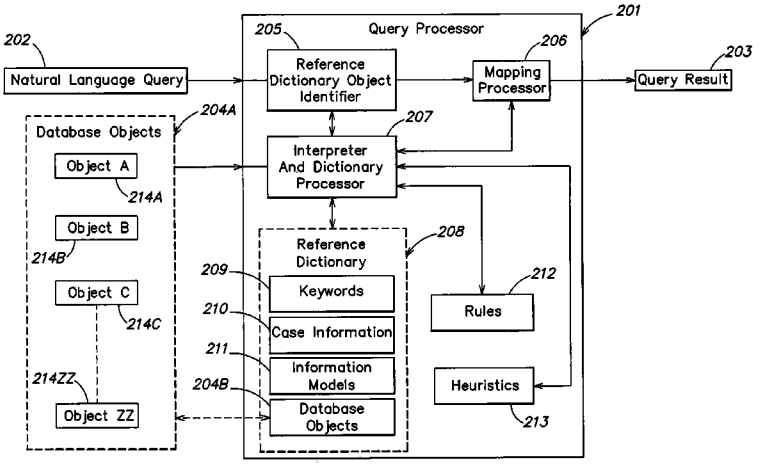
(Continued)

Primary Examiner—Vijay B. Chawan
(74) *Attorney, Agent, or Firm*—Wolf, Greenfield & Sacks, P.C.

(57) **ABSTRACT**

A method for processing a natural language input provided by a user includes: providing a natural language query input to the user; performing, based on the input, a search of one or more language-based databases; providing, through a user interface, a result of the search to the user; identifying, for the one or more language-based databases, a finite number of database objects; and determining a plurality of combinations of the finite number of database objects. The one or more language-based databases include at least one metadata database including at least one of a group of information types including case information, keywords, information models, and database values.

21 Claims, 18 Drawing Sheets



U.S. PATENT DOCUMENTS

5,759,101	A		6/1998	Von Kohorn
5,794,050	A		8/1998	Dahlgren et al.
5,794,207	A		8/1998	Walker et al.
5,794,237	A	*	8/1998	Gore, Jr. 704/7
5,825,881	A		10/1998	Colvin, Sr.
5,845,255	A		12/1998	Mayaud
5,857,184	A	*	1/1999	Lynch 707/4
5,862,223	A		1/1999	Walker et al.
5,864,844	A	*	1/1999	James et al. 704/9
5,875,437	A		2/1999	Atkins
5,884,323	A		3/1999	Hawkins et al.
5,895,464	A	*	4/1999	Bhandari et al. 707/3
5,930,769	A		7/1999	Rose
5,933,822	A	*	8/1999	Braden-Harder et al. 707/5
5,940,811	A		8/1999	Norris
5,941,944	A		8/1999	Messerly
5,948,040	A		9/1999	DeLorme et al.
5,956,699	A		9/1999	Wong et al.
5,963,924	A		10/1999	Williams et al.
5,963,926	A		10/1999	Kumomura
5,970,474	A		10/1999	LeRoy et al.
5,974,146	A		10/1999	Randle et al.
5,982,891	A		11/1999	Ginter et al.
5,987,132	A		11/1999	Rowney
5,987,140	A		11/1999	Rowney et al.
5,987,440	A		11/1999	O'Neil et al.
5,999,908	A		12/1999	Abelow
6,023,684	A		2/2000	Pearson
6,024,288	A		2/2000	Gottlich et al.
6,026,345	A		2/2000	Shah et al.
6,026,375	A		2/2000	Hall et al.
6,055,514	A		4/2000	Wren
6,070,147	A		5/2000	Harms et al.
6,076,088	A	*	6/2000	Paik et al. 707/5
6,081,774	A	*	6/2000	de Hita et al. 704/9
6,094,649	A	*	7/2000	Bowen et al. 707/3
6,105,865	A		8/2000	Hardesty
6,119,101	A		9/2000	Peckover
6,125,356	A		9/2000	Brockman et al.
6,233,578	B1	*	5/2001	Machihara et al. 707/10
6,260,024	B1		7/2001	Shkedy
6,275,824	B1		8/2001	O'Flaherty et al.
6,356,905	B1		3/2002	Gershman et al.
6,505,175	B1		1/2003	Silverman et al.
6,505,183	B1	*	1/2003	Loofbourrow et al. 706/45
6,546,388	B1	*	4/2003	Edlund et al. 707/5
6,556,983	B1	*	4/2003	Altschuler et al. 706/55
6,625,583	B1		9/2003	Silverman et al.
6,711,585	B1	*	3/2004	Copperman et al. 707/104.1
6,718,324	B2	*	4/2004	Edlund et al. 707/5
6,766,320	B1	*	7/2004	Wang et al. 707/5
2002/0087578	A1	*	7/2002	Vroman 707/104.1
2004/0236778	A1	*	11/2004	Junqua et al. 707/100

FOREIGN PATENT DOCUMENTS

EP	0 863 453	A1	9/1998
WO	WO 97/26612	A1	7/1997
WO	WO 98/41956	A1	9/1998
WO	WO 99/01834	A1	1/1999
WO	WO 99/08238	A1	2/1999

OTHER PUBLICATIONS

Burns et al., "Development of a Web-Based Intelligent Agent for the Fashion Selection and Purchasing Process via Electronic Commerce," Department of Administrative Sciences, Kent State University, pp. 140-142.

Decker et al., "Designing Behaviors for Information Agents," The Robotics Institute, Carnegie Mellon University, Jul. 5, 1996, pp. 1-15.

Decker et al., "Matchmaking and Brokering," The Robotics Institute, Carnegie-Mellon University, May 16, 1996, pp. 1-19.

Gregg, "DSS Access on the WWW: An Intelligent Agent Prototype," School of Accountancy and Information Systems, College of Business, Arizona State University, Tempe, pp. 155-157.

Hadidi et al., "Student's Acceptance of Web-Based Course Offerings: An Empirical Assessment," Department of Management Information Systems, School of Business and Management, University of Illinois at Springfield, pp. 1051-1053.

He et al., "Personal Security Agent: KQML-Based PKI," The Robotics Institute, Carnegie Mellon University, Oct. 1, 1997, pp. 1-14.

Moore et al., "The Information Warfare Advisor: An Architecture for Interacting with Intelligent Agents Across the Web," Science Applications International Corporation, Software & Systems Integration Group, pp. 186-188.

Pannu et al., "A Learning Personal Agent for Text Filtering and Notification," The Robotics Institue, School of Computer Science, Carnegie Mellon University, pp. 1-11.

Sameshima et al., "Authorization with security attributes and privilege delegation Access control beyond the ACL," Computer Communications, 20, 1997, pp. 376-384.

Sugumaran, "A Distributed Intelligent Agent-Based Spatial Decision Support System," Department of Business Administration, Le Moyne College, pp. 403-405.

Sycara et al., "Coordination of Multiple Intelligent Software Agents," International Journal of Cooperative Information Systems, pp. 1-31.

Sycara et al., "Distributed Intelligent Agents," The Robotics Institute, Carnegie Mellon University, pp. 1-32.

Zeng et al., "Cooperative Intelligent Software Agents," The Robotics Institute, Carnegie Mellon University, Mar. 1995, pp. 1-10.

Zhao, "Intelligent Agents for Flexible Workflow Systems," Department of Information and Systems Management, Hong Kong University of Science and Technology, pp. 237-239.

Weizenbaum, J. 1966, *ELIZA-A Computer Program for the Study of Natural Language Communication between Man and Machine, Communications of the ACM* vol. 9 No. 1: pp. 36-44.

Waltz, D.L. 1978, *An English Language Question Answering System for a Large Relational Database, Communications of the ACM* vol. 21 No. 7: pp. 526-539.

Codd, E.F., *How about Recently? (English Dialog with Relational Databases Using Rendezvous Version 1)*. In B. Shneiderman (Eds.). *Databases: Improving Usability and Responsiveness*, 1978, pp. 3-28.

Hendrix, G.G. Sacerdoti, E.D. Sagalowicz, C. and Slocum, J. 1978, *Development a Natural Language Interface to Complex Data, ACM Trans. on Database Systems* vol. 3. No. 2: pp. 105-147.

Gross, B.J. Appelt, D.E. Martin, P.A. and Pereira, F.C.N. 1987, *TEAM; an Experiment in Design of Transportable Natural-Language Interfaces, ACM Trans*actions vol. 32: pp. 173-243.

Janus, J.M. 1986, *The Semantics-based Natural Language Interface to Relational Databases*, in L. Bolc and M. Jarke (Eds). *Cooperative Interfaces to Information Systems*. pp. 143-187, New York: Springer-Verlag.

Motro, A. 1990, *FLEX, A Tolerant and Cooperative User Interface to Databases, IEEE Transactions on Knowledge and Data Engineering*. vol. 2 No. 2: pp. 231-246.

Guida, G. and Tasso C. 1982, *NLI: A Robust Interface for Natural Language Person-Machine Communication, Int. J. Man-Machine Studies* vol. 17: pp. 417-433.

International Search Report for International Application No. PCT/US01/16459, mailed Jun. 3, 2002.

Kamel, M. et al., "A graph based knowledge retrieval system," Proceedings of the International Conference on Systems, Man, and Cybernetics, Los Angeles, CA, Nov. 4-7, 1990, pp. 267-275.

Owel, V. et al., "Natural language query filtration in the Conceptual Query Language," Proceedings of the Thirtieth Hawaii International Conference on System Sciences, Wailea, HI, Jan. 7-10, 1997, pp. 539-549.

Shimazu, Hideo et al., "CAPIT: Natural Language Interface Design Tool with Keyword Analyzer and Case-Based Parser," NEC Research and Development, vol. 33, No. 4, Oct. 1, 1992, pp. 679-688, Tokyo, Japan.

Wu, X et al., "KDA: a knowledge-based database assistant," Fifth International Conference on Data Engineering, Los Angeles, CA, Feb. 1-10, 1989, pp. 402-209.

* cited by examiner

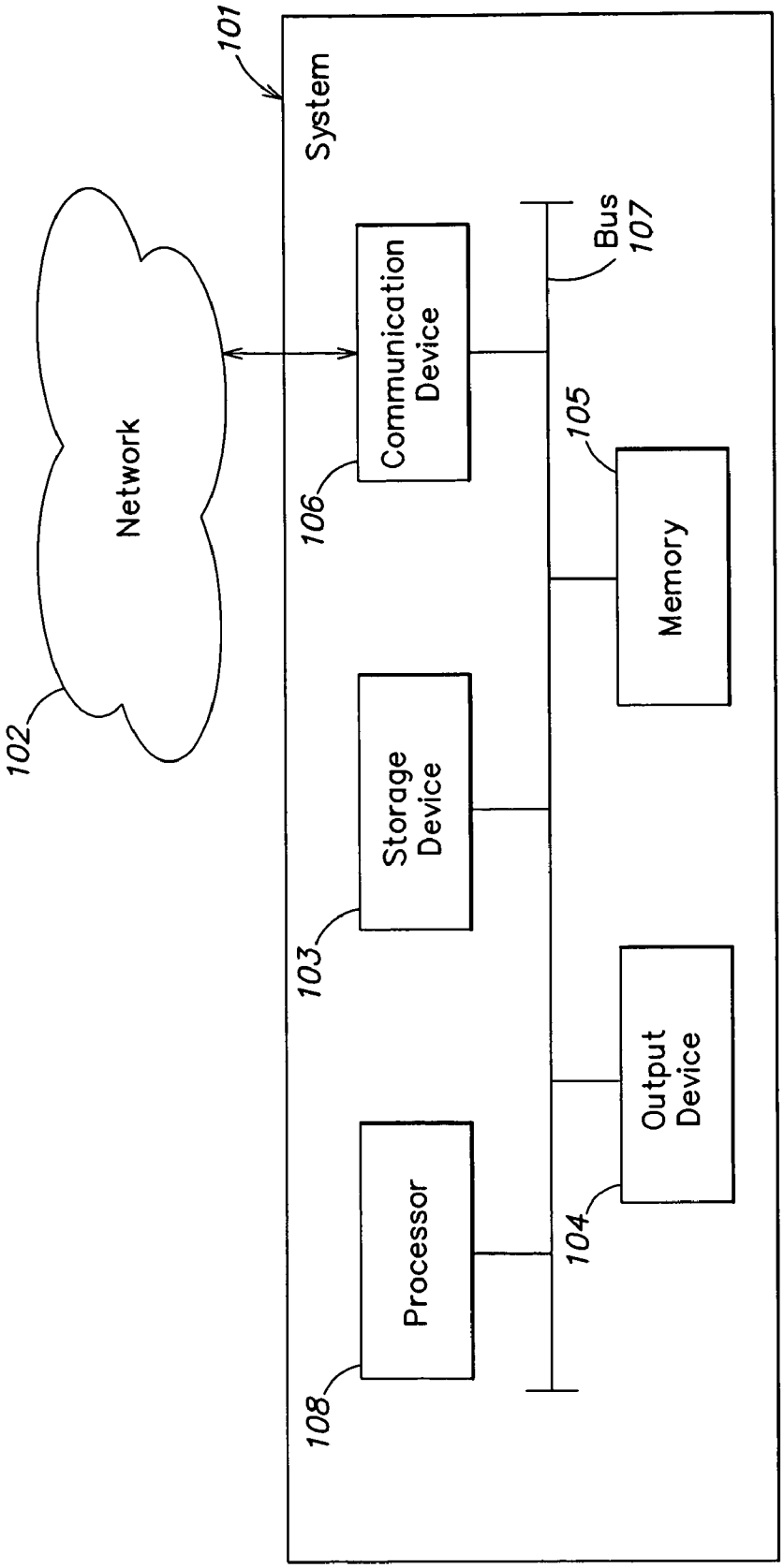

101
System
Network
102
Communication Device
106
Bus
107
105
Memory
Storage Device
103
Output Device
104
Processor
108

FIG. 1

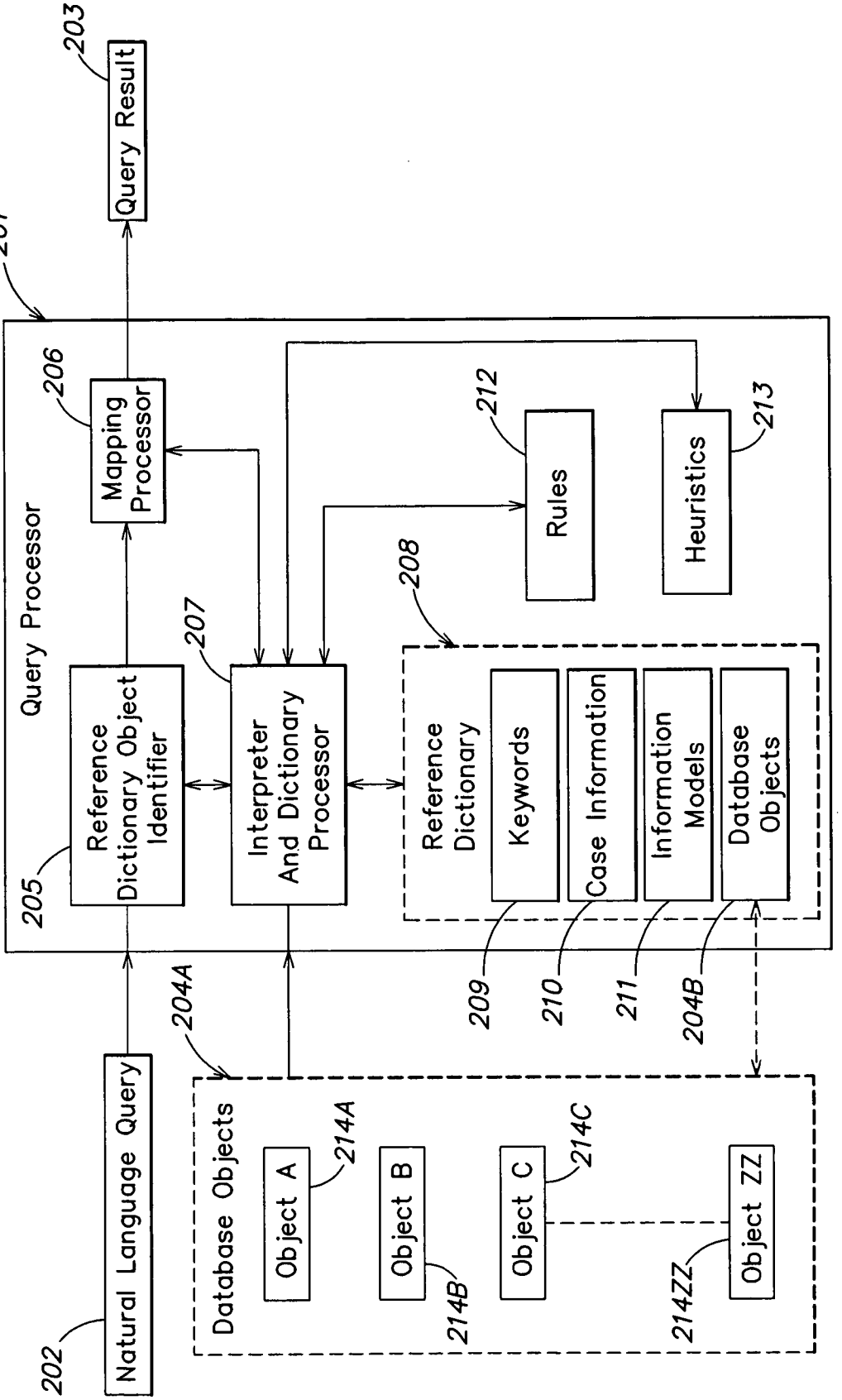

203
Query Result
201
Query Processor
206
Mapping Processor
212
Rules
213
Heuristics
208
207
Reference Dictionary Object Identifier
Interpreter And Dictionary Processor
Reference Dictionary
Keywords
Case Information
Information Models
Database Objects
205
209
210
211
204B
204A
Natural Language Query
Database Objects
Object A
214A
Object B
214B
Object C
214C
Object ZZ
214ZZ
202

FIG. 2

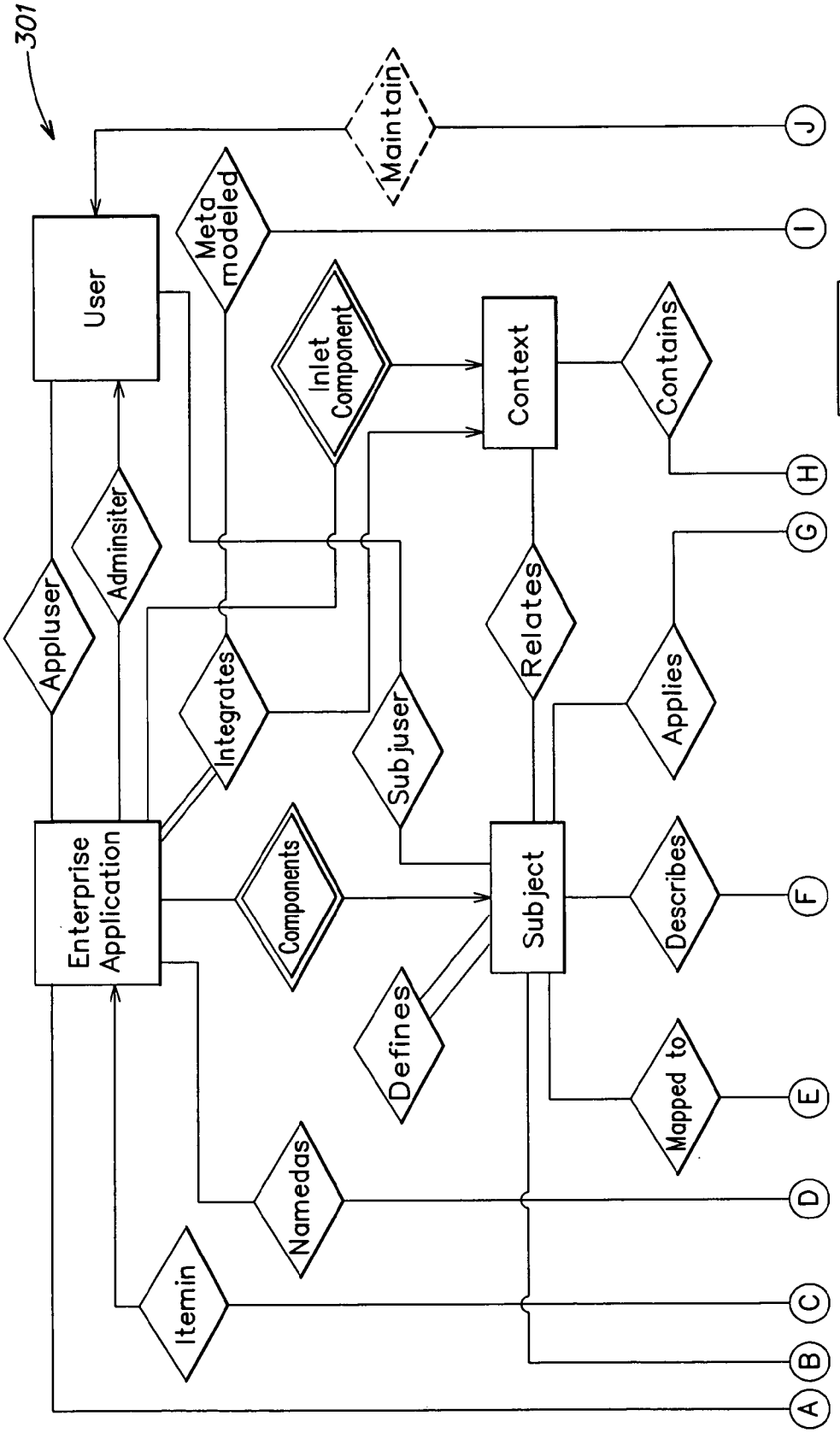


FIG. 3A

FIG. 3A
FIG. 3B
FIG. 3C

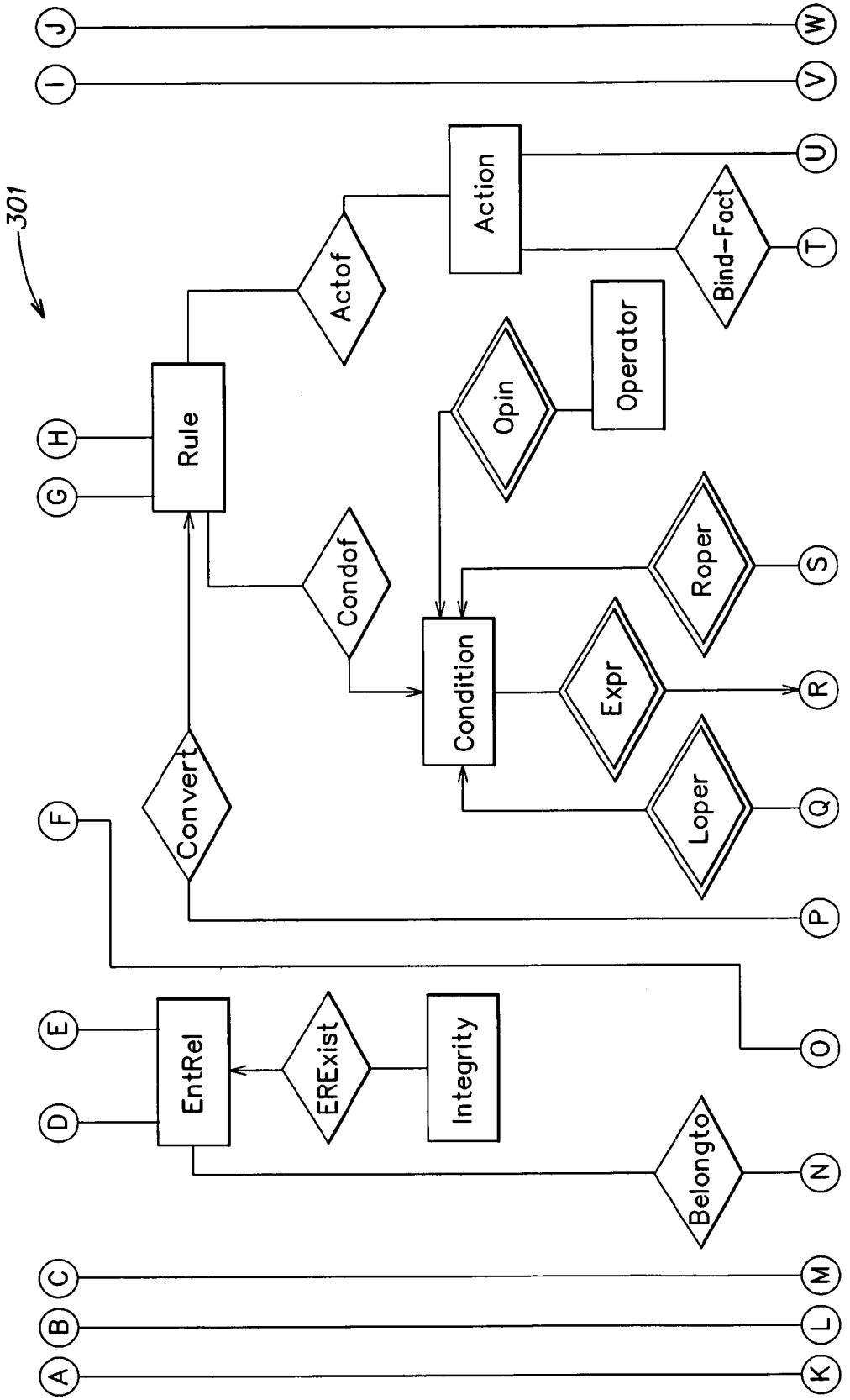

301
Rule
Actof
Action
Bind-Fact
Opin
Operator
Condof
Roper
Condition
Expr
Loper
Convert
EntRel
ERExist
Integrity
Belongto
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W

FIG. 3B

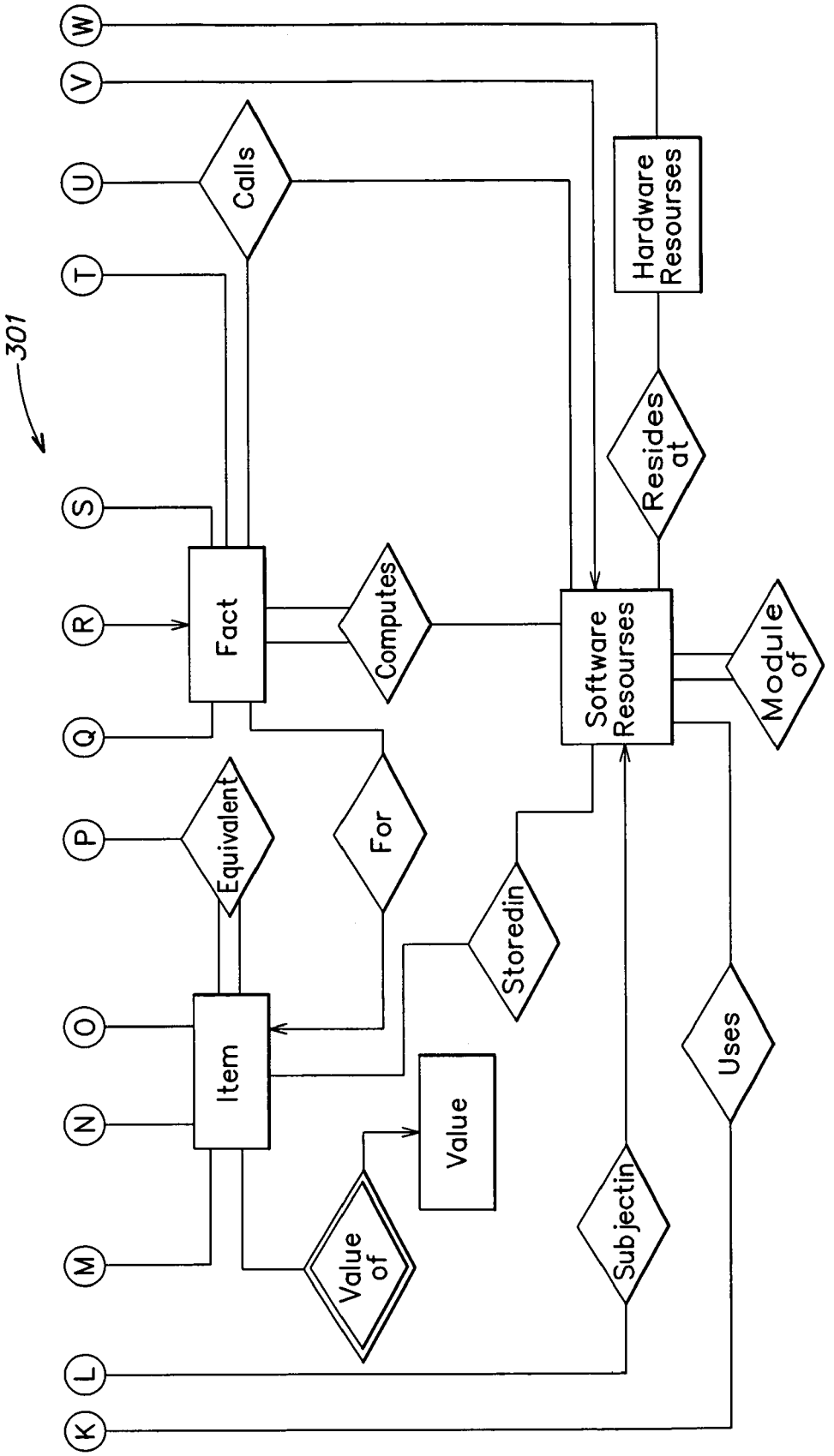

301
W
V
U
T
S
R
Q
P
O
N
M
L
K
Calls
Hardware Resourses
Resides at
Fact
Computes
Software Resourses
Module of
Equivalent
For
Storedin
Item
Value
Value of
Subjectin
Uses

FIG. 3C

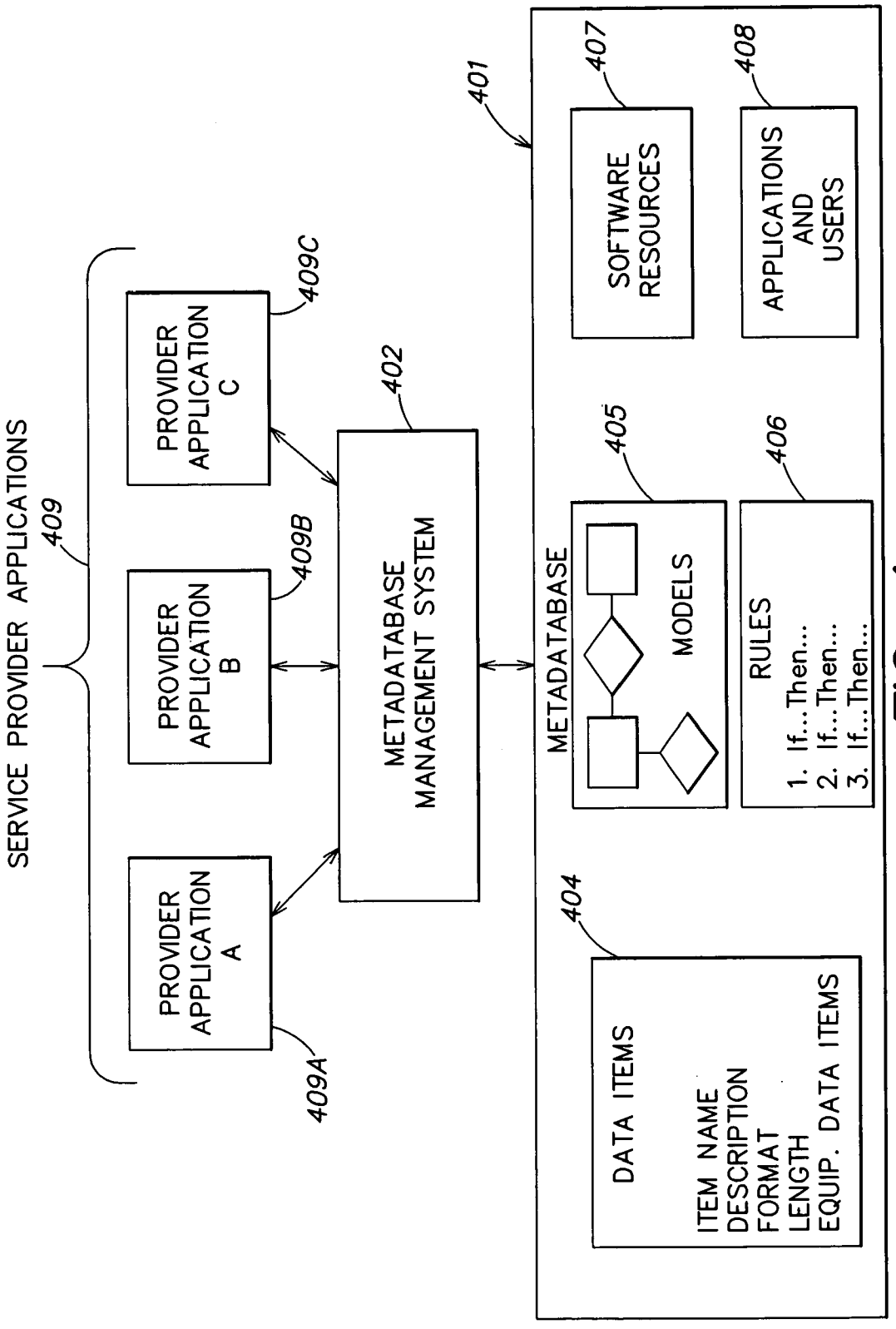

SERVICE PROVIDER APPLICATIONS
409
PROVIDER APPLICATION A
409A
PROVIDER APPLICATION B
409B
PROVIDER APPLICATION C
409C
METADATABASE MANAGEMENT SYSTEM
402
401
METADATABASE
405
MODELS
406
RULES
1. If...Then...
2. If...Then...
3. If...Then...
404
DATA ITEMS
ITEM NAME
DESCRIPTION
FORMAT
LENGTH
EQUIP. DATA ITEMS
407
SOFTWARE RESOURCES
408
APPLICATIONS AND USERS

FIG. 4

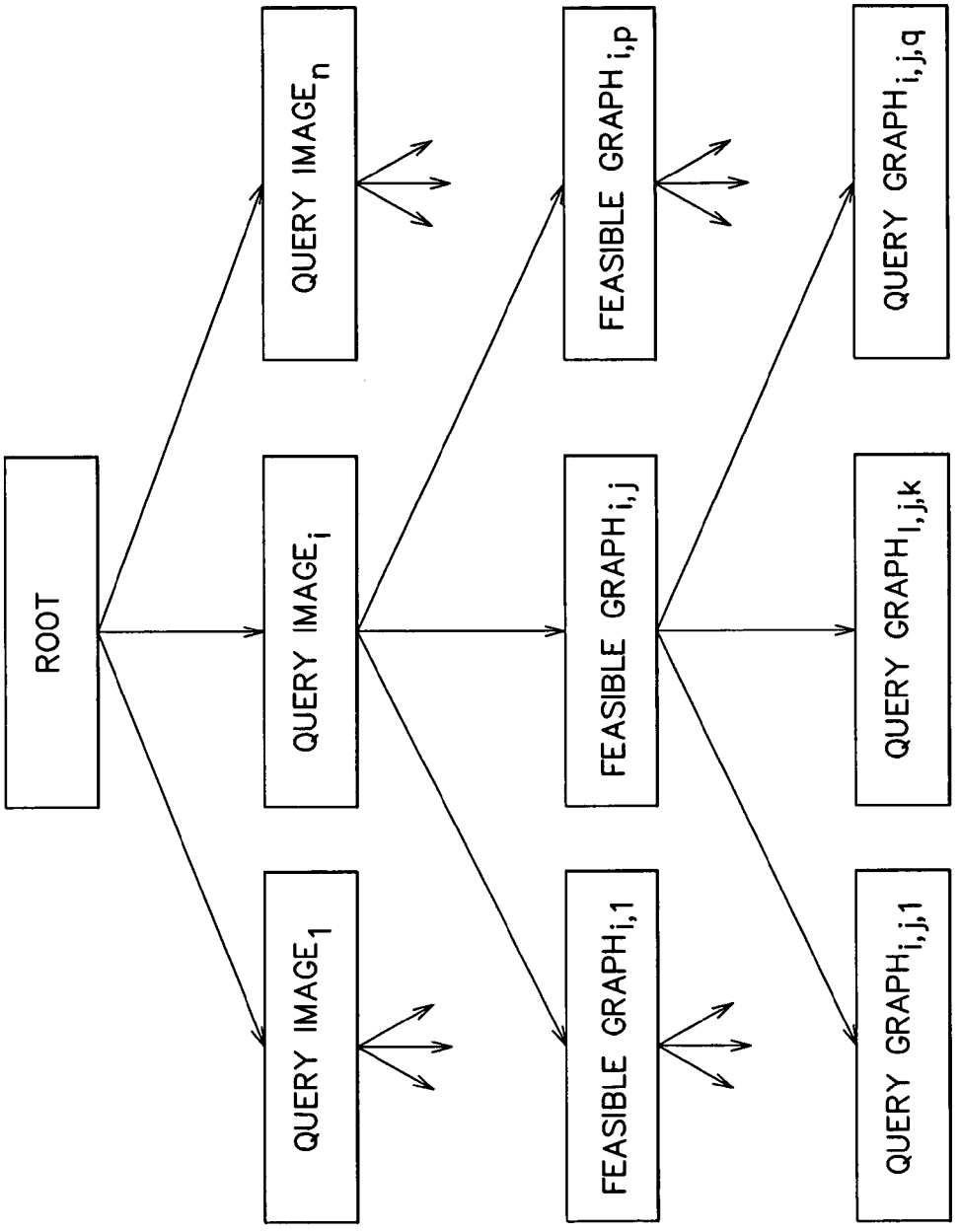

ROOT
QUERY IMAGE$_1$
QUERY IMAGE$_i$
QUERY IMAGE$_n$
FEASIBLE GRAPH$_{i,1}$
FEASIBLE GRAPH$_{i,j}$
FEASIBLE GRAPH$_{i,p}$
QUERY GRAPH$_{i,j,1}$
QUERY GRAPH$_{i,j,k}$
QUERY GRAPH$_{i,j,q}$

FIG. 5

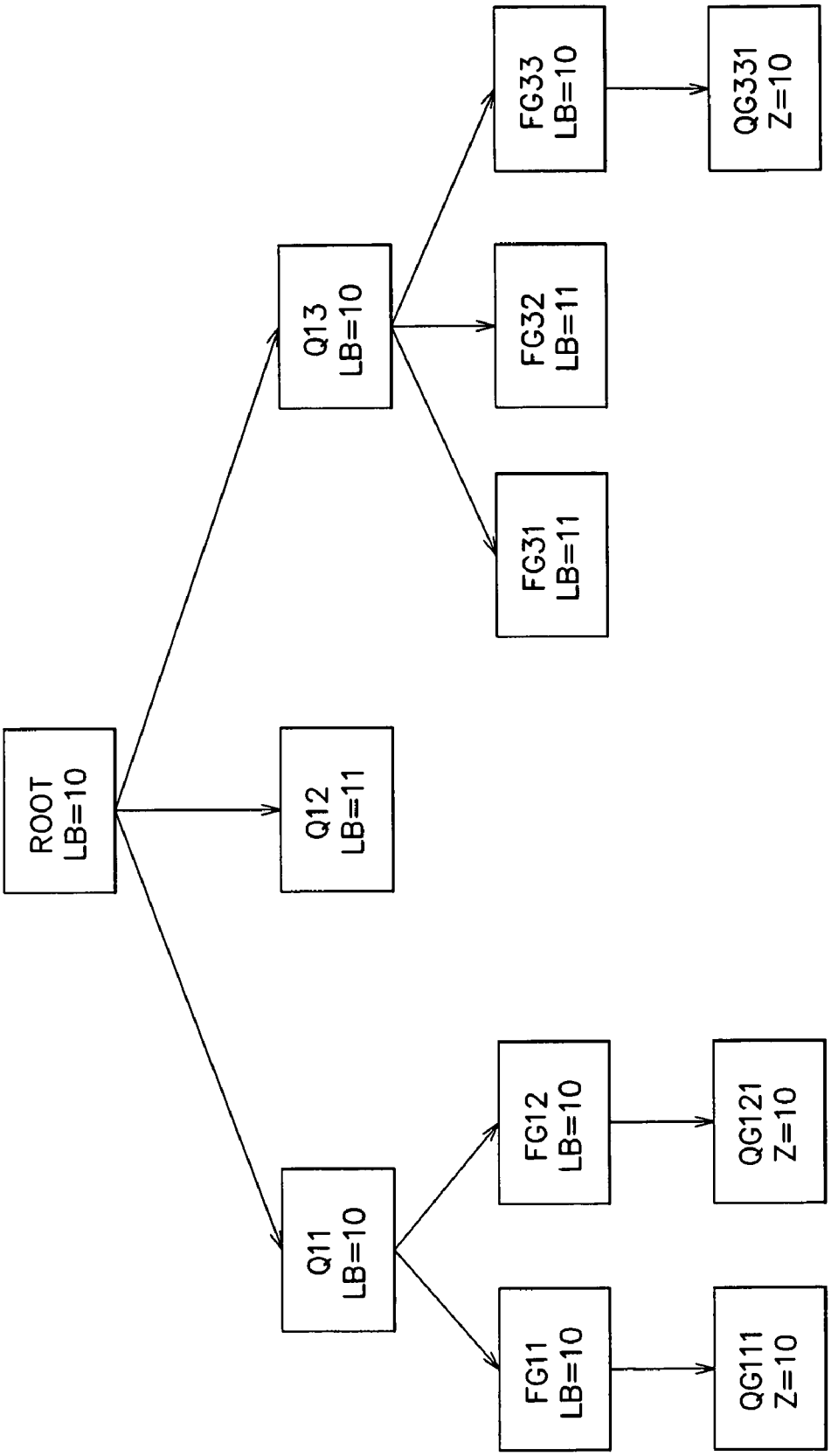

ROOT
LB=10
Q11
LB=10
Q12
LB=11
Q13
LB=10
FG11
LB=10
FG12
LB=10
FG31
LB=11
FG32
LB=11
FG33
LB=10
QG111
Z=10
QG121
Z=10
QG331
Z=10

FIG. 6

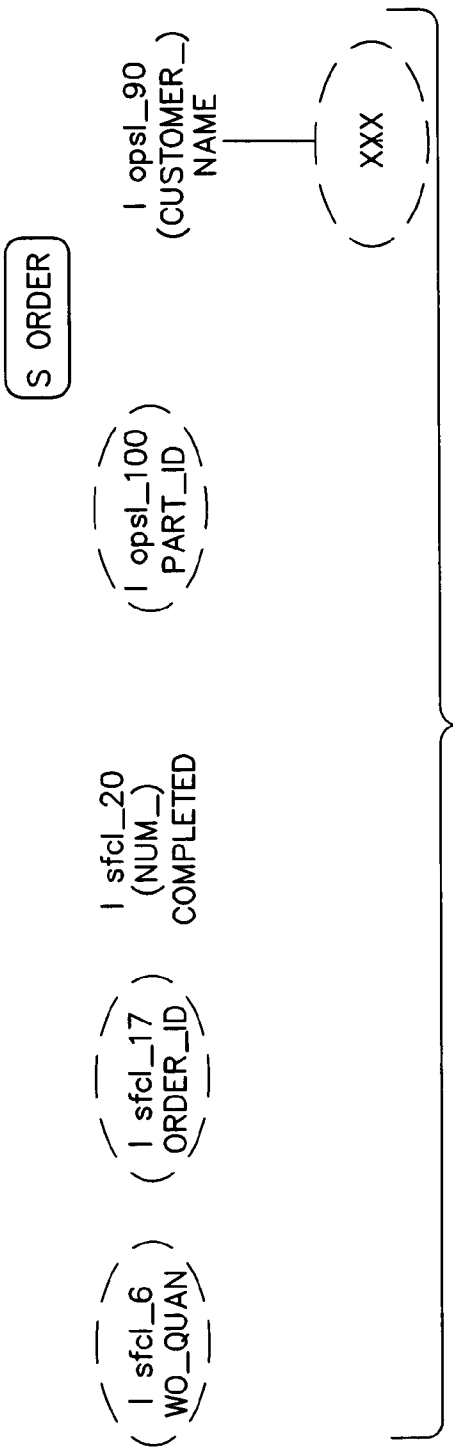

S ORDER
I opsl_90 (CUSTOMER_) NAME
XXX
I opsl_100 PART_ID
I sfcl_20 (NUM_) COMPLETED
I sfcl_17 ORDER_ID
I sfcl_6 WO_QUAN

FIG. 7

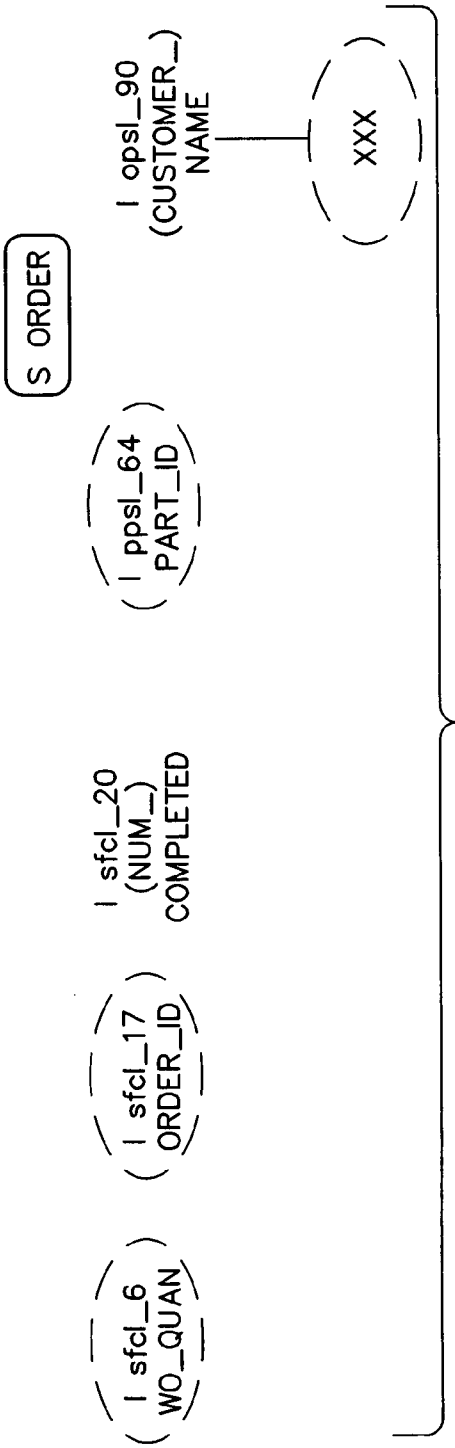

S ORDER
I opsl_90 (CUSTOMER_) NAME
XXX
I ppsl_64 PART_ID
I sfcl_20 (NUM_) COMPLETED
I sfcl_17 ORDER_ID
I sfcl_6 WO_QUAN

FIG. 8

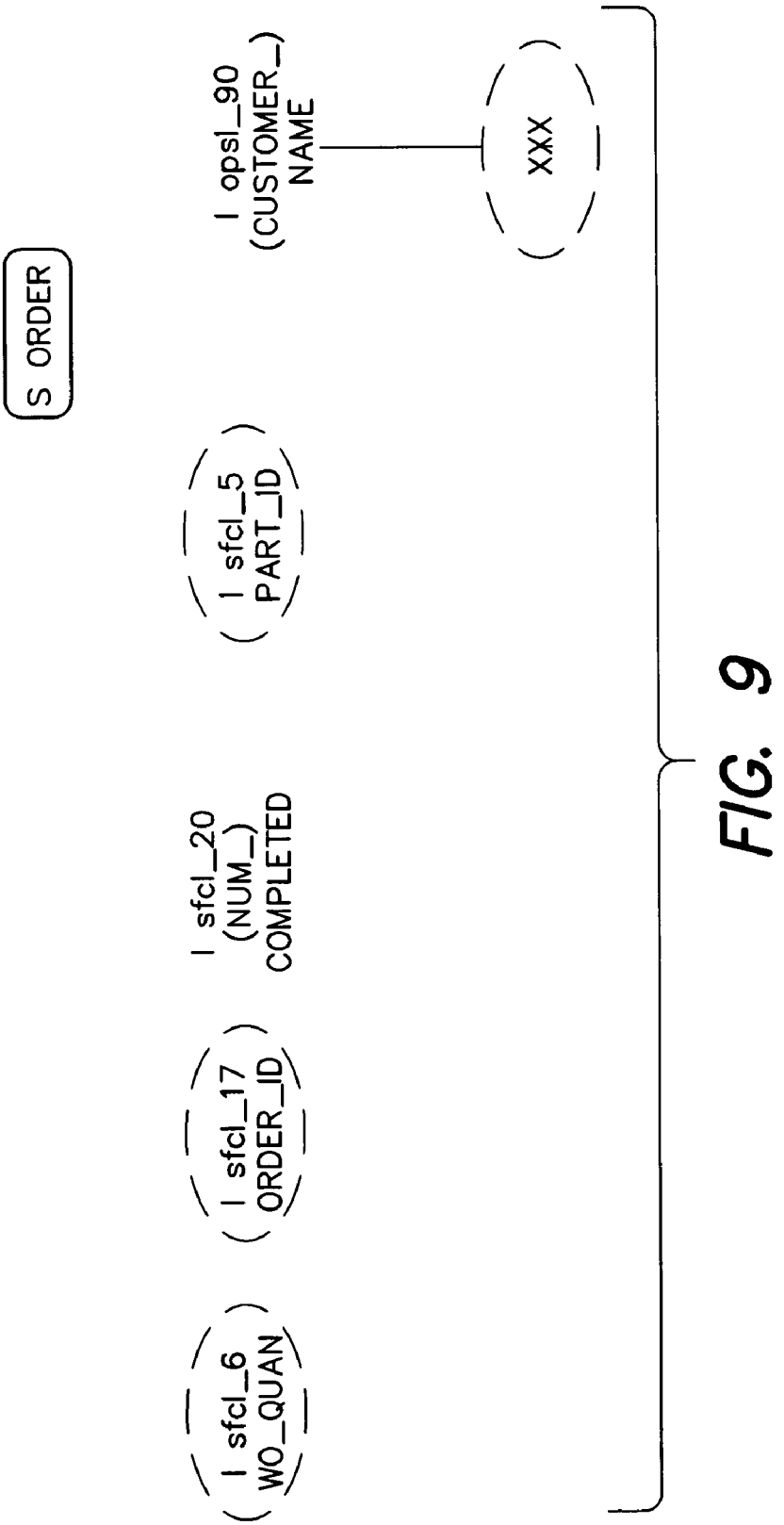

S ORDER
l opsl_90
(CUSTOMER_)
NAME
XXX
l sfcl_5
PART_ID
l sfcl_20
(NUM_)
COMPLETED
l sfcl_17
ORDER_ID
l sfcl_6
WO_QUAN

FIG. 9

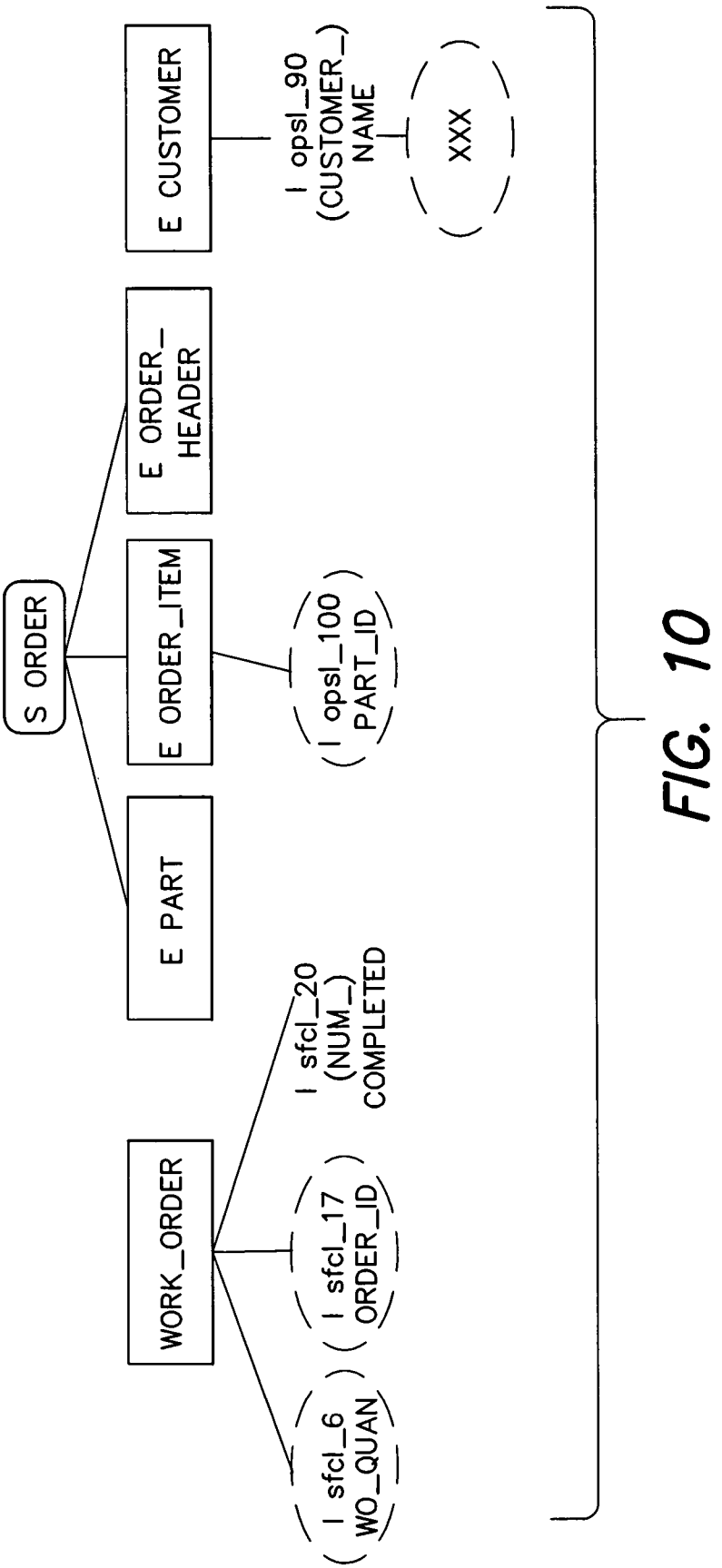

S ORDER
E CUSTOMER
E ORDER_HEADER
E ORDER_ITEM
E PART
WORK_ORDER
I opsl_90 (CUSTOMER_) NAME
XXX
I opsl_100 PART_ID
I sfcl_20 (NUM_) COMPLETED
I sfcl_17 ORDER_ID
I sfcl_6 WO_QUAN

FIG. 10

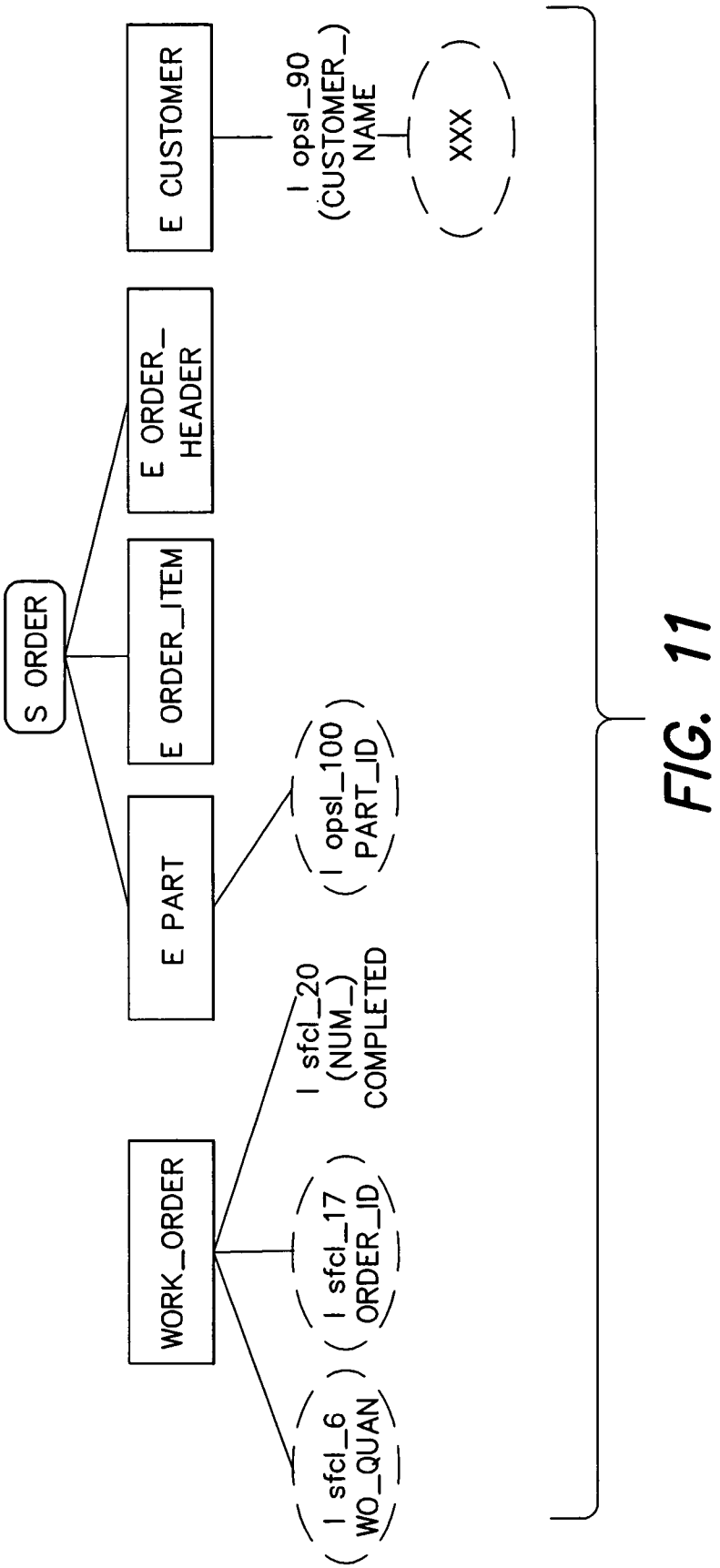

S ORDER
E ORDER_HEADER
E ORDER_ITEM
E PART
E CUSTOMER
WORK_ORDER
I opsl_90 (CUSTOMER_NAME
XXX
I opsl_100 PART_ID
I sfcl_20 (NUM_COMPLETED
I sfcl_17 ORDER_ID
I sfcl_6 WO_QUAN

FIG. 11

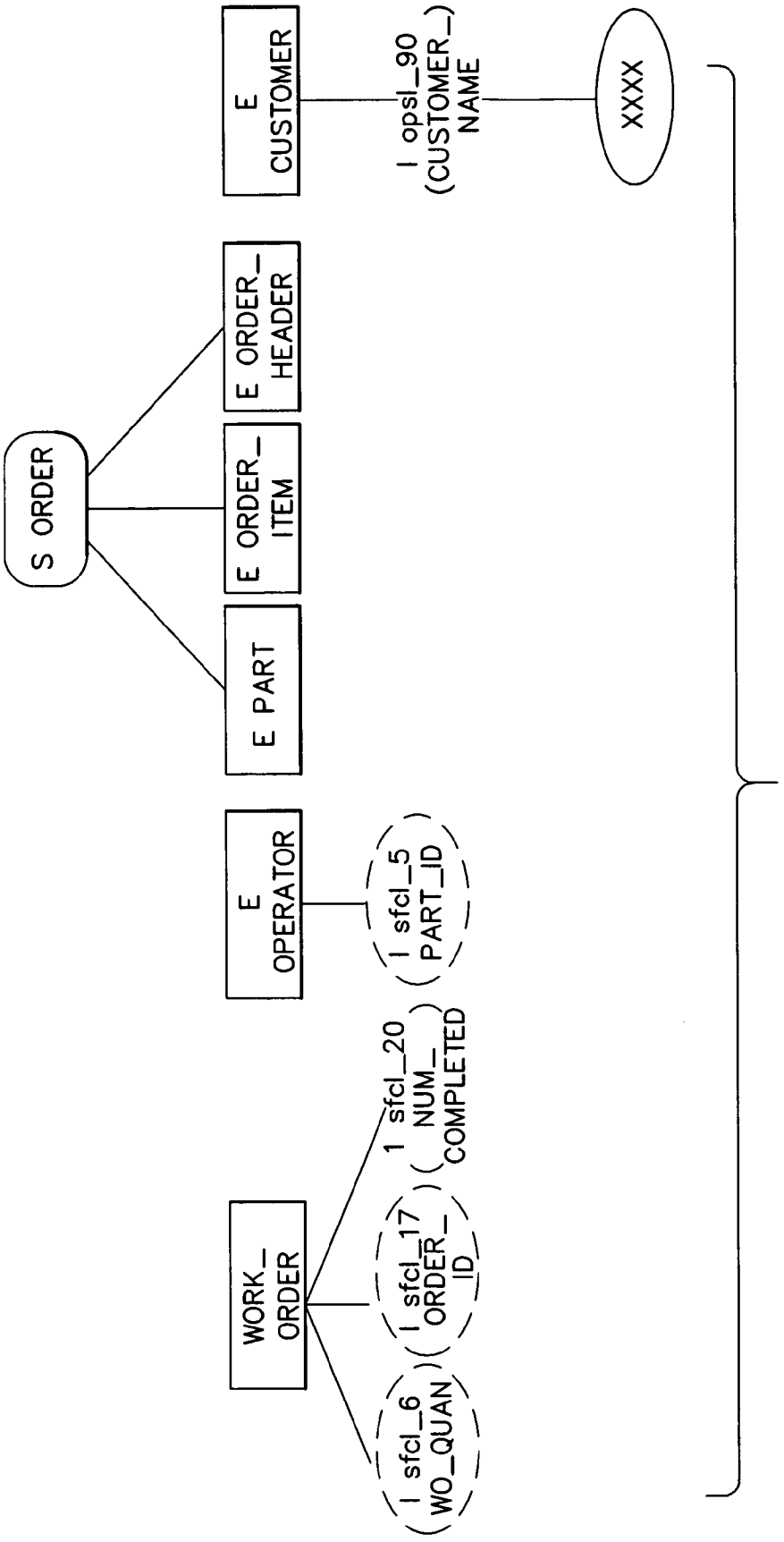

S ORDER
E ORDER_ HEADER
E ORDER_ ITEM
E PART
E OPERATOR
E CUSTOMER
WORK_ ORDER
1 sfcl_6 WO_QUAN
1 sfcl_17 ORDER_ ID
1 sfcl_20 NUM_ COMPLETED
1 sfcl_5 PART_ID
1 opsl_90 (CUSTOMER_) NAME
XXXX

FIG. 12

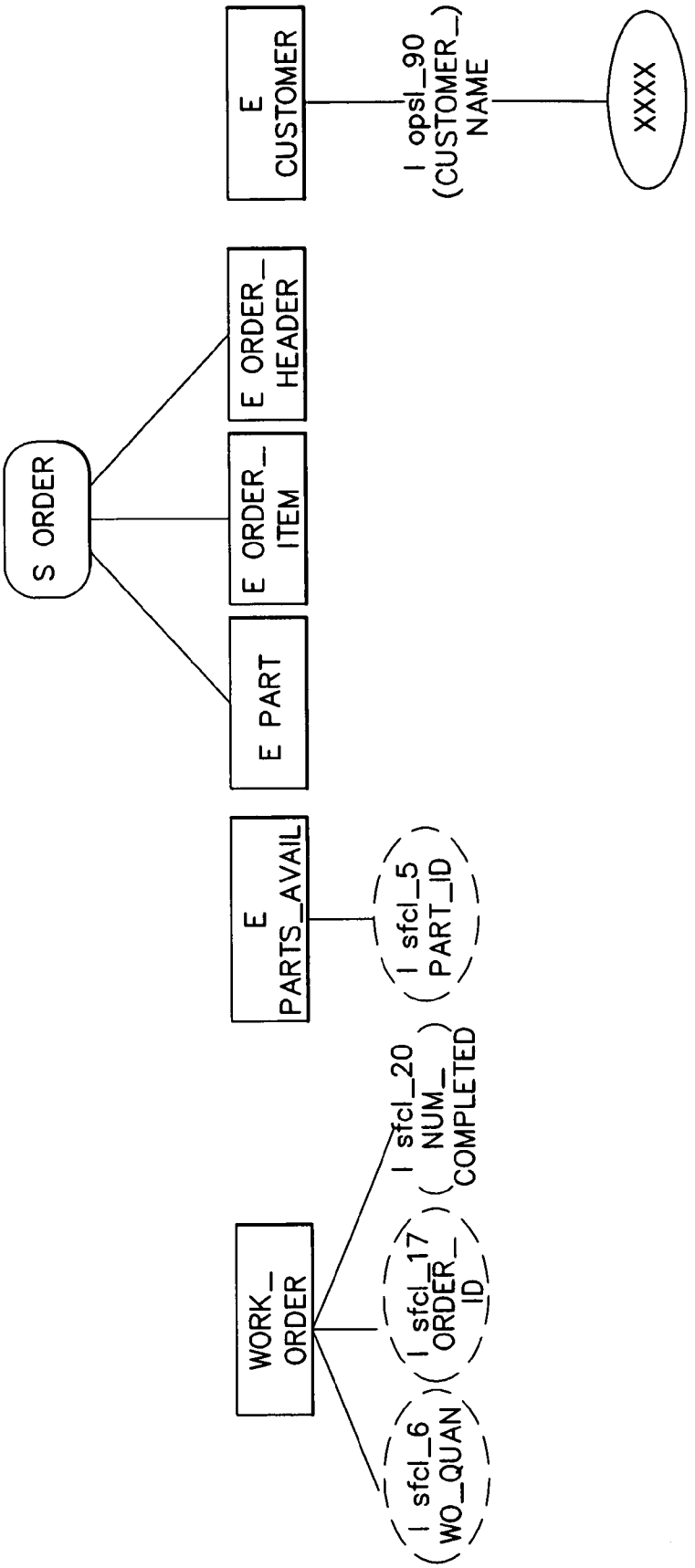

S ORDER
E ORDER_ HEADER
E ORDER_ ITEM
E PART
E PARTS_AVAIL
E CUSTOMER
WORK_ ORDER
I sfcl_6 (WO_QUAN)
I sfcl_17 (ORDER_ ID)
I sfcl_20 (NUM_ COMPLETED)
I sfcl_5 (PART_ID)
I opsl_90 (CUSTOMER_ NAME)
XXXX

FIG. 13

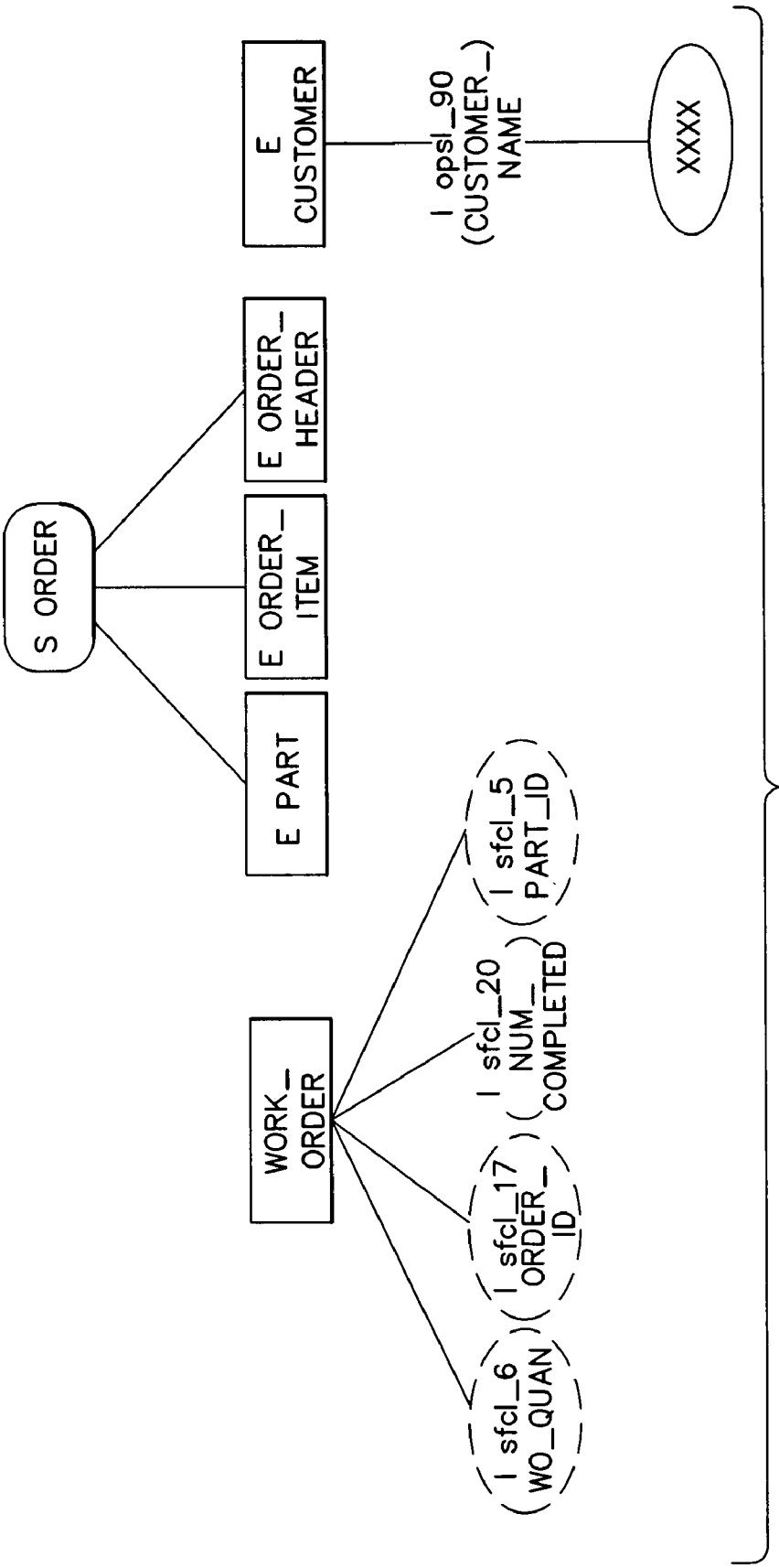

S ORDER
E ORDER_ HEADER
E ORDER_ ITEM
E PART
E CUSTOMER
I opsl_90 (CUSTOMER_) NAME
XXXX
WORK_ ORDER
I sfcl_5 PART_ID
I sfcl_20 NUM_ COMPLETED
I sfcl_17 ORDER_ ID
I sfcl_6 WO_QUAN

FIG. 14

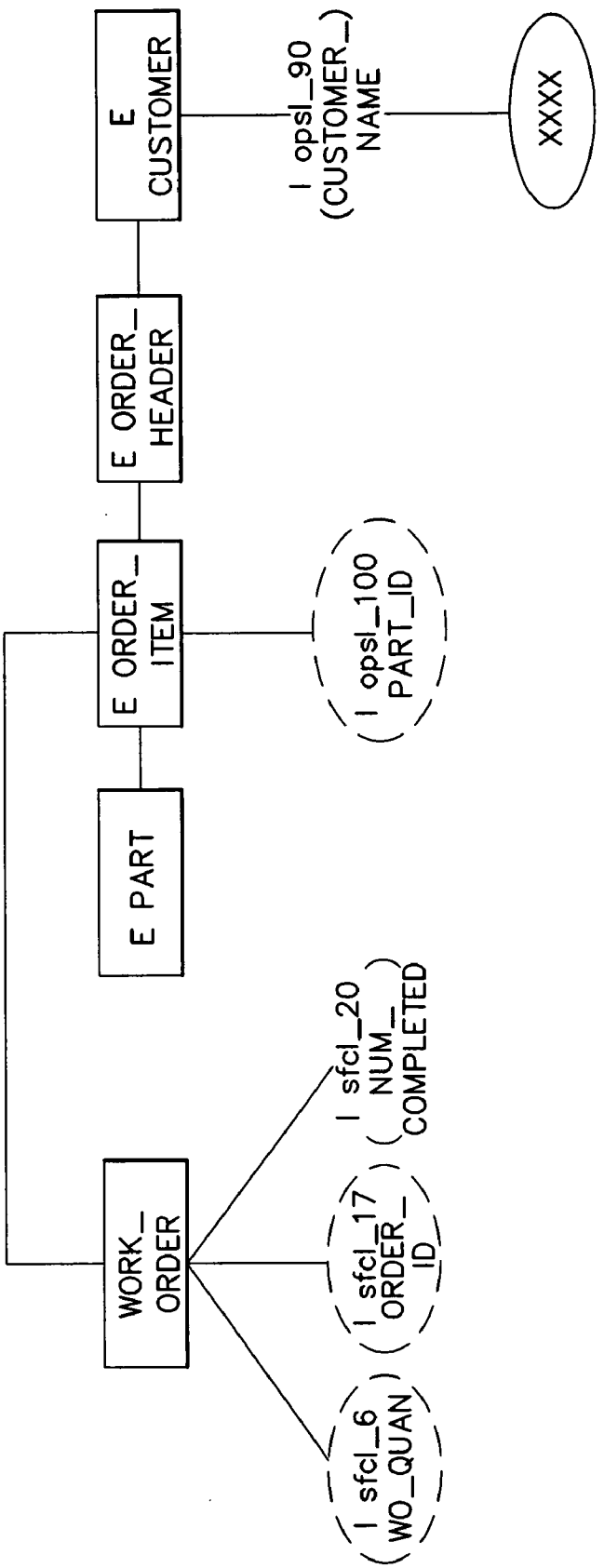

E CUSTOMER
I opsl_90 (CUSTOMER_) NAME
XXXX
E ORDER_ HEADER
E ORDER_ ITEM
I opsl_100 PART_ID
E PART
WORK_ ORDER
I sfcl_20 NUM_ COMPLETED
I sfcl_17 ORDER_ ID
I sfcl_6 WO_QUAN

FIG. 15

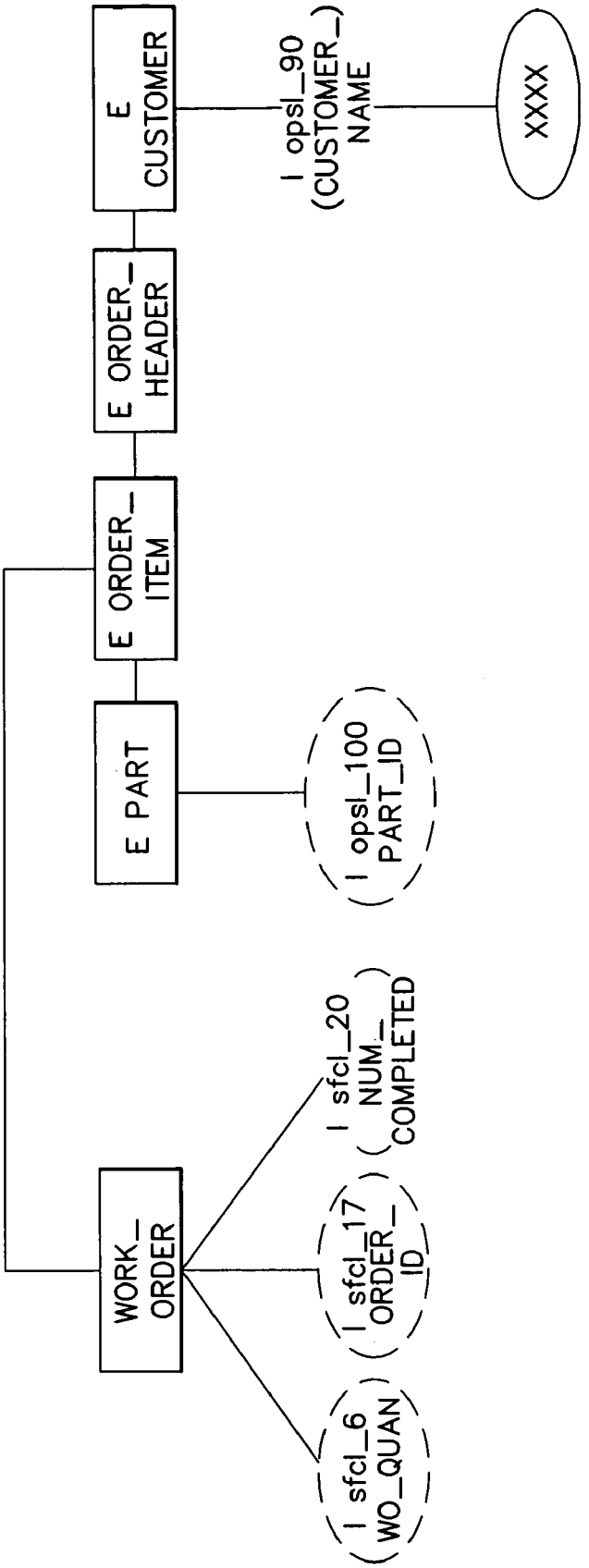

E CUSTOMER
I opsl_90 (CUSTOMER_) NAME
XXXX
E ORDER_ HEADER
E ORDER_ ITEM
E PART
I opsl_100 PART_ID
WORK_ ORDER
I sfcl_20 NUM_ COMPLETED
I sfcl_17 ORDER_ ID
I sfcl_6 WO_QUAN

FIG. 16

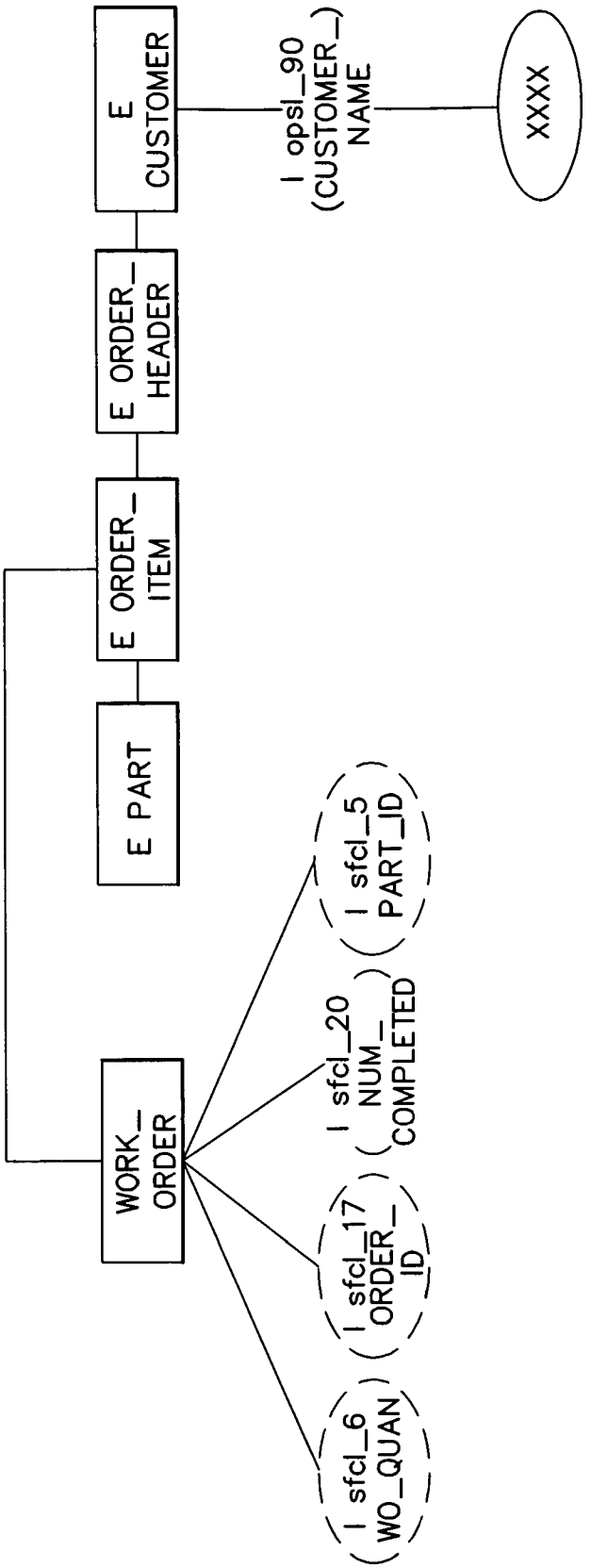

E CUSTOMER
I opsl_90 (CUSTOMER_) NAME
XXXX
E ORDER_ HEADER
E ORDER_ ITEM
E PART
WORK_ ORDER
I sfcl_5 PART_ID
I sfcl_20 NUM_ COMPLETED
I sfcl_17 ORDER_ ID
I sfcl_6 WO_QUAN

FIG. 17

NATURAL LANGUAGE INTERFACE USING CONSTRAINED INTERMEDIATE DICTIONARY OF RESULTS

RELATED APPLICATIONS

This application claims the benefit under Title 35 U.S.C. §119(e) of U.S. Provisional Application Ser. No. 60/205,725, filed May 19, 2000, entitled "SEARCH AND LEARN: A NEW APPROACH TO NATURAL LANGUAGE USER INTERFACE FOR ENTERPRISE DATABASES" by Cheng Hsu and Veera Boonjing, and is a continuation-in-part of application Ser. No. 09/544,676 entitled "SYSTEM AND METHOD FOR ACCESSING PERSONAL INFORMATION," filed Apr. 17, 2000 by Cheng Hsu, et al and which is now abandoned, and the contents of the aforementioned applications are incorporated herein by reference.

FIELD OF THE INVENTION

The field of the invention generally relates to user interfaces, and more specifically, to user interfaces that recognize natural language.

BACKGROUND OF THE INVENTION

Natural language (NL) processing has achieved considerable progress in areas such as speech recognition end generation. Natural language systems have become commonplace, especially in server-based Internet applications, speech recognition products, database search tools, and other environments where human interaction is required. However, decades of hard work by some of the brightest minds in the Artificial Intelligence field has proven the understanding of speech one of the most evasive information technology goals. Researchers, therefore, have lowered their expectations for practical NL systems. For instance, existing NL prototypes focus on specific topics of conversation as more broader applications are more difficult to apply conventional NL techniques. Examples of these focused prototypes include the JUPITER telephone-based conversational system developed at the Massachusetts Institute of Technology that provides weather forecasts and the MOVIELINE system developed at Carnegie Mellon University that provides local movie schedules. These prototypes serve as a first step towards a broader range understanding of NL solutions. Alternatively, some leading industrial efforts concentrate on building a logical structure (conceptual networks—not linguistics) for a general dictionary to support understanding and translation (e.g., the MINDNET language processing software developed by Microsoft Corporation of Redmond, Wash.).

As applied in the database areas, existing natural language interfaces (NLIs) place severe restrictions on the syntax with which users articulate their natural queries. Users typically balk at such restrictions. Thus, prompted in part by the Internet, several systems (e.g., the ASK.COM® search engine of Ask Jeeves, Inc., Emeryville, Calif.) strive to make the restrictions invisible to users; but these restrictions are still within these systems. In these systems, accuracy of interpretation and effort required of the user depends on how closely a database query matches underlying "templates." Same emerging designs for database queries use semantic models and dictionaries, in a spirit similar to the logical structure approach. However, NL systems have not achieved the accuracy and reliability expected of them.

Demand for natural queries continues to grow not only in the area of database searching but also in the area of Internet search engines. The ASK.COM® search engine is an example of an Internet search engine that allows a user to perform natural queries in the form of a free format input. However, the search engine's results rely on (1) its recognized keywords, (2) predefined keyword related to recognized keywords (text classification), and (3) predefined possible questions (templates) associated with each group of keywords from (1) and (2). Its processing steps are (1) capture all recognized keywords from the inputs, (2) determine all keywords tat relate to recognized keywords, (3) retrieve and display predefined questions (each question wit one keyword group). Though the ASK.COM® search engine looks natural, it functions structurally as a text classification system, and does not actually interpret natural language queries. Therefore, it does not have the capability of processing natural language queries.

SUMMARY OF THE INVENTION

A problem realized with many conventional natural language system designs is that these designs require exceedingly large collections of linguistic terms that users use, but still might not be able to assure successful closure of users' queries. Because of design complexity and keyword database size, most systems are not practical to implement. A better approach to processing natural language inputs is therefore needed.

According to one embodiment of the invention, a system and method is provided that utilizes a definitive model of enterprise metadata, a design of keywords with simplified complexity, a graphical model of logical structure, a branch and bound search algorithm, and a case-based interaction method to process natural language inputs.

According to one aspect of the invention, a method is provided for processing a natural language input provided by a user. The method comprises providing a natural language query input to the user, performing, based on the input, a search of one or more language-based databases including at least one metadata database comprising at least one of a group of information types comprising case information, keywords, information models, and database values, and providing, through a user interface, a result of the search to the user. According to another embodiment of the invention, the method further comprises a step of identifying, for the one or more language-based databases, a finite number of database objects, and determining a plurality of combinations of the finite number of database objects. According to another embodiment of the invention, the method further comprises a step of mapping the natural language query to the plurality of combinations. According to another embodiment of the invention, elements of the metadata database are graphically represented.

According to another embodiment of the invention, the step of mapping comprises steps of identifying keywords in the natural language query, and relating the keywords to the plurality of combinations. According to another embodiment of the invention, the method further comprises a step of determining a reference dictionary comprising case information, keywords, information models, and database values. According to another embodiment of the invention, the step of mapping further comprises resolving ambiguity between the keywords and the plurality of combinations. According to another embodiment of the invention, the step of resolv-

ing includes determining an optimal interpretation of the natural language query using at least one of a group comprising rules and heuristics.

According to another aspect of the invention, a method for processing a natural language input is provided comprising providing a plurality of database objects, identifying a finite number of permutations of the plurality of database objects, the database objects being stored in a metadata database comprising at least one of a group of information comprising case information, keywords, information models, and database values, and interpreting at least one of the permutations to determine a result of the natural language input. According to another embodiment of the invention, the step of providing a plurality of database objects includes providing at least one of the group comprising data types, data instances, and database computational operators. According to another embodiment of the invention, the step of interpreting includes mapping the at least one of the permutations to a database query. According to another embodiment of the invention, the database query is formulated in a structured query language (SQL) query.

According to another embodiment of the invention, one or more permutations are eliminated. According to another embodiment of the invention, the method further comprises providing a reference dictionary comprising cases, keywords, information models, and database values, identifying, in the natural language input, a plurality of elements belong to the reference dictionary determining complete paths that implied by the plurality of elements that span elements of the natural language input and which belong to the graphics of the reference dictionary. According to another embodiment of the invention, the method further comprises determining a path that include elements of at least the informational models and database values. According to another embodiment of the invention, the method further comprises providing rules and heuristics for searching, and determining an optimum permutation based on the rules and heuristics. According to another embodiment of the invention, the method further comprises adding, as a result of user input, new cases and keywords to the reference dictionary. According to another embodiment of the invention, elements of the metadata database are graphically represented.

According to another aspect of the invention, a method is provided for processing a natural language input comprising determining, from the natural language input, a plurality of recognized terms, the recognized terms existing in a data dictionary having a logical graph structure, determining a minimum number of the plurality of recognized terms, and determining vertices associated with the minimum number of the logical graph structure, determining at least one minimum cost query graph that contains a minimum amount of vertices, if there are more than one minimum cost query graphs, remove at least one redundant cost query graph and producing a solution set of cost query graphs, determining, within the solution set, at least one cost query graph that is a complete solution, and translating the at least one cost query graph to a query language statement.

Further features and advantages of the present invention as well as the structure and operation of various embodiments of the present invention are described in detail below with reference to the accompanying drawings. In the drawings, like reference numerals indicate like or functionally similar elements. Additionally, the left-most one or two digits of a reference numeral identifies the drawing in which the reference numeral first appears.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description when taken in conjunction with the accompanying drawings in which similar reference numbers indicate the same or similar elements.

In the drawings,

FIG. **1** shows a general purpose computer in which one embodiment of the invention may be implemented;

FIG. **2** shows a natural language query processor in accordance with one embodiment of the invention;

FIG. **3** shows a reference dictionary in accordance with one embodiment of the invention;

FIG. **4** shows a metadatabase system in accordance with one embodiment of the invention;

FIG. **5** shows an example complete search graph in accordance with one embodiment of the invention;

FIG. **6** shows an example search graph in accordance with one embodiment of the invention;

FIG. **7** shows an example query image in accordance with one embodiment of the invention;

FIG. **8** shows an example query image in accordance with one embodiment of the invention;

FIG. **9** shows an example query image in accordance with one embodiment of the invention;

FIG. **10** shows an example feasible graph in accordance with one embodiment of the invention;

FIG. **11** shows an example feasible graph in accordance with one embodiment of the invention;

FIG. **12** shows an example feasible graph in accordance with one embodiment of the invention;

FIG. **13** shows an example feasible graph in accordance with one embodiment of the invention;

FIG. **14** shows an example feasible graph in accordance with one embodiment of the invention;

FIG. **15** shows an example query graph in accordance with one embodiment of the invention;

FIG. **16** shows an example query graph in accordance with one embodiment of the invention; and

FIG. **17** shows an example query graph in accordance with one embodiment of the invention.

DETAILED DESCRIPTION

A typical goal of a natural language interface (NLI) to databases is to allow users to access information stored in databases by articulating questions directly in their natural language such as English. A large number of natural language interfaces have been developed by specialists in computational linguistics and artificial intelligence to achieve this goal. However, they do not seem to be truly "natural". In the context of this research, "truly natural" means any styles of linguistic articulation that a native speaker could understand and use sensibly. Available results tend to require of users using only some well-structured sentence templates and a tightly controlled vocabulary set to articulate queries. These artificial constructs are hardly natural to those who prefer their own choice of vocabulary, phrases, and expressions. Natural articulation goes beyond any fixed design of templates and significant-words dictionary. For example, a user of a Computer-Integrated Manufacturing database might query the system in any number of ways in a truly natural environment, such as:

I have asked you many times and I still have not heard a clear answer. Listen, I would like to know the status of my orders. I want to know what models you have started working on and how many of them are completed. I placed these orders on, I believe, Sep. 9, 1999. Could you please help? Thanks.

Step on John Smith's order. I want to know the exact status of all parts he ordered including their descriptions, prices, and quantities completed to-date. I understand that you're supposed to deliver it on Feb. 29, 2000. I know I might not have given you all the information you need, but please do your best, anyway. Okay? I'm waiting . . .

Just give me everything we have about John Smith and Jim Kowalski; both are our customers.

Orders of John Smith

Green's orders

PZ1 customers

Processing queries is difficult for the computer to interpret precisely. Queries embody the hallmark of natural languages: ambiguity and implicitness. Not only do multiple, different interpretations exist simultaneously for the same articulation of a query, the articulation might also even be incomplete or incorrect. Thus, a good NLI has to be able to either understand the natural articulation or to enumerate and evaluate all possible interpretations. Because experience can be used to improve interpretation (either understanding or enumeration-evaluation), an NLI also needs to be able to learn from its performance, including both successes and failures. Learning can involve a number of possible tasks throughout the entire query process. The NLI could respond intelligently to the user's articulation and if needed solicit more information about the query. It could confirm the meanings and results with the user in order to assist evaluation and assure correctness. The NLI could also retain valuable cases (usage patterns) so that the NLI could improve its performance. The first two tasks of learning could help "close the loop" so that a query is always executed properly (completeness and correctness of query processing), while the third reduces the complexity of interpretation (e.g., the number of possible interpretations).

Most previous research efforts on NLI have set out to find ways for the computer to understand the user's articulation, following the more established tradition of Artificial Intelligence (AI). But their results have stopped significantly short of being truly natural. They have all endeavored to devise particular controls and limitations on the naturalness of input and use these artifacts to assist interpreting queries into some standard database query languages. A handful of basic approaches limit naturalness so that the system can reduce the complexities of interpretation and process the queries successfully. These approaches include (1) a template-based approach, (2) a syntax-based approach, (3) a semantics-grammar-based approach, (4) an intermediate-representation-language-based approach, and (5) a semantics-model-based approach. These approaches differ in the way each controls input and in the extent to which each imposes controls on the user.

Each of the first four approaches requires users to articulate only in natural language forms that the system provides—or at least they assume that the user's articulation is consistent with these underlying forms. When this basic requirement or assumption does not hold in practice, the system fails to function properly (e.g., with poor performance and low accuracy), or even fails altogether. These forms typically feature some generic, linguistic prototype consisting of only one single sentence per query. Their disadvantage is in their restriction on naturalness.

The fifth and last approach seeks for naturalness, allowing free-format text as input, but it couples a particular NLI design with a particular domain of application. If the first four approaches are "top-down" in relying on predefined natural language forms, the last one in contrast exhausts all possible interpretations from the "bottom up". Its basic method is to provide a semantic model or a dictionary as the roadmap to generate possible interpretations. Its control is implicit: it assumes that users always query the databases known to the system, and can therefore always tune the NLI according to this known kernel of meaning.

One could argue that users are bound to refer, either directly or indirectly, to these known database objects (types or semantic models, instances or values, and operators) in their natural queries. If they do not use these database objects directly, they still have to use other words and phrases (henceforth known as "keywords") that correspond to these objects. Thus, the domain of interpretation is finite, compared to natural language processing in general. The semantic model provides a network of meanings of keywords, and a dictionary provides a more extensive collection of keywords beyond the usual semantic model. The critical success factor of the last approach depends clearly on the semantic model-dictionary employed, which must be powerful enough at least to span the range of possible usage which natural language encompasses. Because a database object can be only a grossly simplistic element of the natural vocabulary, keywords must shoulder the burden of representing naturalness. Their number could increase exponentially as the number of users and usage patterns increase, giving the bottom-up model the same disadvantages of the top-down model.

According to one embodiment of the invention, an NLI is provided that provides truly natural query capability to end users of enterprise databases in that the NLI interprets any style of articulation and to learn from the users in a way that improves both effectiveness and efficiency. The strategy uses a concept referred to herein as search and learn. This approach recognizes implicit enumeration-evaluation as a basic solution paradigm to the problem of natural language queries. Based on this analysis, a reference dictionary is used that integrates enterprise metadata (information models and contextual knowledge) with case-based reasoning. The new design affects two vital functions: (1) the generation of all possible interpretations of a natural query suitable for evaluation, and (2) the reduction of the complexity of keywords and the reduction of growth of keywords. According to one embodiment of the invention, a reference dictionary is used to search for an optimal solution and the dictionary "learns" from experience, achieving maximum naturalness with minimum enumeration. Compared to conventional approaches, this new approach promises realistic performance and completeness of a solution because the new reference dictionary and learning capability allows for the determination of complete solutions. In a broader sense, the new approach, according to one embodiment of the invention, identifies that the NLI problem is primarily a search problem and relates the problem to the vast tradition of constrained optimization (e.g., scheduling and traveling salesman).

According to one embodiment of the invention, it is realized that models are the conceptual networks of enterprise databases, which are the known domain of enterprise users' queries. Therefore, it develops an innovative approach to use enterprise metadata to search and interpret

the meaning of natural queries. The objective is to support queries expressed in any combinations of multiple sentences (essays), any forms of sentence (complete or not), and any vocabulary (personal or standard) against databases that have well-defined information models. Furthermore, this objective may be accomplished with practical performance free of the above problems. A simplified model of the general logical structure paradigm is provided which is more generic than the current NLI prototypes, when applied to enterprise database queries. According to various embodiments of the invention, an NLI system includes one or more of a new class of reference dictionary (of enterprise metadata), a branch-and-bound search method, and a case-based reasoning design, to implicitly evaluate a number of possible interpretations of the user's natural articulation and determine the optimal solution. An advantage of various aspects of the invention is that there is no reliance on linguistics; nor is there a requirement of an all-encompassing linguistic-grammatical dictionary. Various embodiments of the invention implement alternative approaches to the traditional machine learning NL approaches.

Enterprise databases represent well-defined application domains and their metadata provide webs of concept for the domains. One task is to determine a minimally sufficient set of metadata and develop a feasible logical structure for natural queries, such that the system could interpret the queries using the structure alone. Users are bound to refer, either directly or indirectly, to database objects (including data semantics, structures, instances or values, and operators) in their natural queries. If they do not use directly these database objects, they articulate their queries in terms of other significant words and phrases (i.e., keywords) that correspond sufficiently to these objects. Therefore, a natural query is reducible to a particular combination of these database objects and keywords. Conversely, a particular combination of database objects and keywords could represent a particular natural query. These keywords are overwhelming when all possible permutations of natural words (phrases) in queries are considered (i.e., a linguistic dictionary). However, keywords could be manageable if they are based on database objects and other known enterprise metadata.

The next problem is that correspondences between natural articulation and metadata are usually not unique due to, for example, incomplete matching or ambiguity. Because there may be ambiguity, implicit enumeration and evaluation may resolve such ambiguities. When a logical structure of all database objects and keywords is in place, combinations become ordered sequences according to the structure, and thereby allow precise evaluation. In other words, a particular permutation of database objects and keywords of the logical structure has a particular interpretation for database query. If the system identifies all permissible permutations implied by a query and evaluates their relative merits, the system has essentially interpreted the query. Then, it is possible to map the query to the underlying database query language (such as SQL) without having to understand linguistically the human meaning of the words and phrases used. This approach is feasible since the space of search (domain of interpretation) is finite; a particular database or distribution of databases has only finite objects. The situation is fundamentally different from conventional natural language processing where the possibility of permutation is virtually infinite. Of course, even the relatively small, finite number of database objects could generate a large number of possible permutations. Thus, according to one embodiment of the invention, a metadata search may be used to efficiently manage database objects.

The above argument presents a way to systematically resolve ambiguity in natural queries. That is, according to one embodiment of the invention, it is realized that ambiguity is the deviation from an exact and sufficient match on the logical structure, including no matching of metadata as well as all forms of multiple matching. In the worst case, the system might not be able to identify any database object and keyword at all. In response to a failure in identifying an object, the system may search the entire space of existing permutations and evaluate each of them for the user to determine which one is correct. Note that, this approach still has a better lower bound than generic natural language processing, which fails without user feedback. Each additional database object the system recognizes as the result of user feedback serves to eliminate certain space from the search and narrow down the possible interpretations. In the best case, the system identifies a single complete permutation in the input and determines a unique and correct interpretation for the user.

It is clear that a minimally sufficient set of metadata can include all proven database objects, of which the information models are largely constant (known at the design time) but the database values are extensible at run time. Therefore, according to various embodiment of the invention, a "base" set of proven database objects may be provided with an NLI system, and modification and/or new database objects may be added at runtime. Next, the metadata set includes keywords to assist matching with database objects.

Keywords, according to one embodiment of the invention, may be identified, organized, and controlled based on an enterprise information model connected to database values. Each database object corresponds to a finite number of keywords, and keywords would generally not be derived from any enumeration of possible permutations of words in natural queries. Therefore, the information model-based subset of keywords would generally be a constant (or a linear growth with a small slope), leaving the database value-based subset to grow with new applications and attain sufficiency without infinite growth in the number of keywords as in conventional systems. According to one aspect of the invention, keyword growth would be a polynomial-type growth at most. Complexity of keywords in accordance with various embodiments of the invention is a significant improvement over that of a linguistic dictionary, whose growth is exponential by nature due to the number of keyword permutations. Further, cases of query processing may be created and integrated with other metadata. Thus, according to one aspect of the invention, there are four layers of enterprise metadata (resources of search) considered; i.e., cases, keywords, information models, and database values. According to one aspect of the invention, they are integrated in an extensible metadata representation method so that every resources item references all other related resources for query interpretation. A repository of metadata may be implemented as, for example, a reference dictionary. More particularly, a semantically based graphical abstraction of the reference dictionary may be used to define the logical structure of the enterprise database query for the application domain concerned. The core of the reference dictionary (information models and initial keywords) may be, for example, a design-time product, developed by analysts, designers, and users. Cases and additional keywords and other metadata (e.g., changes to the information models) and database values can be added during runtime as the

enterprise database system ages and evolves. Further, case-based reasoning may be used for producing richer keywords and cases.

General Purpose Computer System

Various aspects of the present invention may be described with reference to a general purpose computer system **101** such as that shown in FIG. **1**. The computer system **101** may include a processor **108** connected to one or more storage devices **103**, such as a disk drive through a communication device such as bus **107**. The computer system also includes one or snore output devices **104**, such as a monitor or graphic display, or printing device. The computer system **101** typically includes a memory **105** for storing programs and data during operation of the computer system **101**. In addition, the computer system may contain one or more communication devices that connect the computer system to a communication network **106**.

Computer system **101** may be a general purpose computer system that is programmable using a high level computer programming language. The computer system may also be implemented using specially programmed, special purpose hardware. In computer system **101**, the processor **108** is typically a commercially available processor, such as the PENTIUM, PENTIUM II, PENTIUM III, PENTIUM IV, or StrongARM microprocessors from the Intel Corporation (Santa Clara, Calif.) ATHLON or DURON processor available from Advanced Micro Devices, Inc. (Sunnyvale, Calif.), PowerPC microprocessor (available from IBM, Armonk, N.Y.), SPARC processor available from Sun Microsystems, Inc. (Santa Clara, Calif.), or 68000 series microprocessor available from Motorola Inc. (Schaumburg. Ill.). Many other processors are available. Such a processor usually executes an operating system which may be, for example, DOS, WINDOWS 95, WINDOWS NT, WINDOWS 2000, or WinCE available from the Microsoft Corporation (Redmond, Wash.), MAC OS SYSTEM 7 available from Apple Computer, (Cupertino, Calif.), SOLARIS available from Sun Microsystems, Inc. (Santa Clara, Calif.), NetWare available from Novell Incorporated (Bellevue, Wash.), PalmOS available from the 3COM corporation (Marlborough, Mass.), or UNIX-based operating systems (such as LINUX) available from various sources. Many other operating systems may be used.

The communication network **102** may be an ETHERNET network or other type of local or wide area network (LAN or WAN), a point-to-point network provided by telephone services, or other type of communication network. Information consumers and providers, also referred to in the art as client and server systems, respectively, communicate through the network **102** to exchange information.

Various aspects of the present invention may be performed by one or more computer systems, processors, or other computing entity. Various aspects of the present invention may be centralized or distributed among more than one system, and the invention is not limited to any particular implementation.

It should be understood that the invention is not limited to a particular computer system platform, processor, operating system, or network. Also, it should be apparent to those skilled in the art that the present invention is not limited to a specific programming language or computer system and that other appropriate programming languages and other appropriate computer systems could also be used.

A general-purpose computer system **101** may include a natural language user interface (NLUI or simply NLI), through which a user requests information and performs other transactions. For instance, the user may provide input and receive output from graphical user interfaces. In one case, the interface may prompt a user with a series of questions, to which the user may respond. The questions may be multiple choice question format, of which a single selection of the choices is an appropriate response. However, the system **101** may present a general query interface on graphical user interface, through which the user may pose natural language queries or responses to questions. For example, system **101** may prompt the user to "Please enter a search (natural language or keyword)." In response, the user may provide a natural language response, asking system **101** "Where is the Houston Field House at RPI located?" The natural language interface may have associated with it a natural language analyzer which determines the meaning of a provided input.

Natural Language Analyzer

According to one embodiment of the invention, a natural language analysis system is provided such as the system shown in FIG. **2** discussed in more detail below. The natural language analysis system finds the meaning of the request and determines the correct source of the information requested. For example, system **101** may format and send the request to a server-based system, and the server-based system may return the result to system **101**.

In general, a natural language analyzer that analyzes queries (hereinafter termed a "natural language query processor") may be part of computer system **101**. This query processor may perform one or more analyzing steps on a received query, which is generally a string of characters, numbers, or other items. A long-standing goal in the field of information technology is to allow humans to communicate with computer systems in the natural languages of humans. However, because of the various ambiguous and implicit meaning found in natural language, queries are difficult for a computer system to interpret precisely.

Most of the conventional methods for understanding natural language queries involve determining a method of understanding a user's natural language articulation by implementing various methods of artificial intelligence. There are several conventional approaches to limiting the naturalness of language so that the system could reduce the complexities of interpretation and successfully process queries. They include (1) template-based approach [e.g., Weizenbaum, J. 1966, *ELIZA-A Computer Program for the Study of Natural Language Communication between Man and Machine, Communications of the ACM* Vol. 9 No. 1: pp. 36–44], (2) syntax-based approach [e.g., Waltz, D. L. 1978, *An English Language Question Answering System for a Large Relational Database, Communications of the ACM* Vol. 21 No. 7: pp. 526–539, Codd, E. F, R. S. Arnold, J-M. Cadiou, C. L. Chang, and N. Roussopoulos, *RENDEZVOUS Vesion 1: An Experimental English Language Query Formulation System for Casual Users of Relational Data Bases*, IBM Research Report RJ2144, 1978., Codd, E. F., *How about Recently? (English Dialog with Relational Databases Using Rendezvous Version* 1). In B. Shneiderman (Eds). *Databases: Improving Usability and Responsiveness,* 1978, pp. 3–28.], (3) semantics-grammar-based approach [e.g., Hendrix, G. G. Sacerdoti, E. D. Sagalowicz, C. and Slocum, J. 1978, *Development a Natural Language Interface to Complex Data, ACM Trans. on Database Systems* Vol. 3. No. 2: pp. 105–147], (4) intermediate-representation-language-based approach [e.g., Gross, B. J. Appelt, D. E. Martin, P. A. and Pereira, F. C. N. 1987, *TEAM: an Experiment in Design of Transportable Natural-Language Inter-*

faces, ACM Transactions Vol. 32: pp. 173–243], and (5) semantics-model-based approach [e.g., Janus, J. M. 1986, *The Semantics-based Natural Language Interface to Relational Databases*, in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 143–187. New York: Springer-Verlag Janus, J. M. 1986, *The Semantics-based Natural Language Interface to Relational Databases*, in L. Bolc and M. Jarke (Eds). *Cooperative Interfaces to Information Systems*. pp. 143–187. New York: Springer-Verlag, Motro, A. 1990, *FLEX, A Tolerant and Cooperative User Interface to Databases, IEEE Transactions on Knowledge and Data Engineering*. Vol. 2 No. 2: pp. 231–246, Guida, G. and Tasso C. 1982, *NLI: A Robust Interface for Natural Language Person-Machine Communication, Int. J Man-Machine Studies* Vol. 17: pp. 417–433].

These conventional approaches differ in the way each controls the input and in the extent to which each exerts the control on the user. The first four approaches (1)–(4) require users to articulate only in the natural language forms that the system provides—or at least they assume that the user's articulation is consistent with these underlying forms. When this basic requirement or assumption does not hold in practice, the system would fail to function properly (e.g., with poor performance and low accuracy), or even fail altogether. These forms typically feature some generic, linguistic prototype consisting of only one single sentence per query. Thus, their advantage is that the resultant NLI is easily portable from one database system to another. The disadvantage is the restriction on naturalness of the input from the user. The last approach (5) essentially embraces a different priority, placing naturalness ahead of portability (i.e., coupling a particular NLI design with a particular domain of application, but allowing free-format text as input). If the first four approaches are top-down in their relying on the computer's direct understanding of the user's articulation, the last one could be considered as the computer's exhausting of all possible interpretations from the bottom up.

The basic strategy of system (1)–(5) is to provide a semantic model or a dictionary as the roadmap for generating possible interpretations. These systems assume that the users always query databases known to the system, thus the NLI could be tuned according to this known information. According to one embodiment of the invention, it is recognized that, under this assumption, users are bound to refer, either directly or indirectly, to these known database objects (types or semantic models, instances or values, and operators) in their natural queries. If they do not use directly these database objects, they have to articulate their query in terms of other significant words and phrases (hereinafter referred to as "keywords") that correspond to these objects. Thus, the domain of interpretation is finite, compared to natural language processing in general. Semantic model is a form of keywords and a dictionary is a more extensive collection of keywords beyond a usual semantic model. The critical success factor of this approach is clearly the semantic model-dictionary employed, which must be powerful enough to span efficiently the space of possible usage of natural language in the domain. Because database objects can only be a grossly simplistic portion of the natural vocabulary, keywords must shoulder the burden of representing naturalness. Their number could increase exponentially as the number of users and usage patterns increase.

There is a need to consider the logical structure paradigm in addition to the narrow application paradigms used above. This paradigm enumerates all concepts—i.e., the logical association of words—that users could use for articulation in the place of a linguistic structure to interpret natural languages. An example would be Microsoft's MindNet software technology developed by Microsoft Corporation of Redmond, Wash., which derives conceptual networks (webs of concepts) of, say, a standard dictionary and matches human queries against the structure. Here, matching is essentially the interpretation. This is obviously a monumental effort in the general case for general natural language processing. However, enterprise databases already represent a much focused application domain of querying; and more significantly, they also provide a much narrower range of concepts and their logical structures. In the database field, information models are webs of concepts for enterprise databases. The problem is, these webs of concepts are typically far from being sufficient for capturing all meaningful words that users could use naturally to articulate queries in the application domain. Thus, to create an efficient logical structure to support a truly natural query, one has to find a way to design a sufficient reference dictionary tat will not grow exponentially, and a way to optimally enumerate and evaluate all possible interpretations.

A new approach according to one embodiment of the invention recognizes implicit enumeration-evaluation as a basic solution paradigm to the problem of natural language queries. The new approach includes designing a reference dictionary of concepts that integrates enterprise information models and contextual knowledge with user-oriented keywords and past cases of usage, to provide the logical structure of natural queries for the enterprises databases concerned. The new design affects two vital functions: (1) generate all possible interpretations of a natural query suitable for evaluation, and (2) stem the complexity and growth of keywords. A new NLI method according to one embodiment of the invention uses a branch-and-bound algorithm to search for an optimal interpretation of the natural query based on the logical structure. Case-based reasoning adds to the search to achieve maximum naturalness with minimum enumeration. With this objective accomplished, truly natural database query would become possible for enterprise applications, which account for a significant portion of all natural language processing needs. In a broader sense, the proposed project formulates for the first time the NLI research as a search problem and relates it to the vast tradition of constrained optimization (e.g., scheduling and traveling salesman). Because, according to one embodiment of the invention, constrained optimization is used, solution to the natural language query problem is bounded. Drawing ideas and strengths from this tradition as well as from the literature of NLI, the new metadata search model according to various aspects of the invention is able to add to the paradigm of logical structures for natural language processing.

FIG. **2** shows a natural language query processor **201** according to one embodiment of the invention. Processor **202** receives a natural language query and a plurality of database objects **204**, and produces a query result. The natural language query may be, for example, a paragraph, a sentence, sentence fragment, or a plurality of keywords. The query result may be any information that is relevant to the combination of database objects **204**A and query **202**. According to one embodiment of the invention, the natural language query **202** is mapped to the plurality of database objects **204**A using a reference dictionary **208** comprising keywords **209**, case information **210**, information models **211**, and database object values **204**B. An advantage of this mapping is that less-capable processing hardware is needed to perform the mapping than traditional natural language processing algorithms such as those cited above, because the

number of keywords that needs to be recognized and searched by the system is reduced. This advantage enables, for example, use of such an NLI on a portable device such as PalmPilot, or other portable device that has limited storage capability. Also, because the portable device may be allocated to a single user, and processor **201** is capable of learning using case-based learning, processor **201** may become more accurate for the particular user.

Natural language processors are well-known, and functions they perform are described in more detail in the book entitled *Natural Language Understanding*, by James Allen, Benjamin/Cummings Publishing Company, Inc., Redwood City, Calif., 1994, herein incorporated by reference. Other natural language query processors are discussed in the journal articles and books cited above.

According to one embodiment of the invention as shown in FIG. **2**, query processor **201** includes a reference dictionary object identifier **205** that parses query **202** and generates one or more objects recognized in the reference dictionary **208**. Reference dictionary object identifier **205** also identifies words that are meaningful in the reference dictionary **1208** and eliminates useless or meaningless words. Processor **201** also accepts and processes a number of database objects **204**A. As discussed, processor **201** may have an associated reference dictionary **208** that includes keywords **209**, case information **210**, information models **211** and one or more database objects **204**B. Keywords **209** may be, for example, a set of keywords and their combinations generated from the plurality of database objects **204**A, which includes one or more objects **214**A–**214**ZZ. Keywords **209** may also be "learned" from a user through performing queries, or may be provided through a separate keyword administrator interface associated with query processor **201**.

Query processor **201** also includes an interpreter and dictionary processor **207** that receives objects identified by the reference dictionary object identifier **205** and determines an optimal interpretation of the received objects. More specifically, processor **207** determines optimal interpretations of the received objects, resolves ambiguities, updates information models **211**, and interacts with users to facilitate learning. Processor **207** utilizes rules **212** and heuristics **213** to resolve ambiguities in determining the optimal interpretation of query **202**. Rules **212** and heuristics **213** may relate to information models **211**, which are in turn related to keywords **209**, cases **210**, and database objects **204**B in a semantic manner. When there are ambiguities in the interpretation of objects, e.g. multiple possible interpretations, multiple permissible combinations of meaningful objects, etc., rules **212** and heuristics **213** related to these objects are used to reduce or resolve these ambiguities.

Mapping processor **206** performs a mapping between incoming objects and database objects **204**A. In particular, processor **206** may generate database queries from the objects and the interpretations provided by identifier **205** and processor **207**, respectively. Processor **206**, may, for example, generate SQL queries used to locate database objects **204**A. These queries may be executed by an SQL search engine, and processor **201** may provide query result **203** to user through, for example, a graphical user interface.

Literature on NLI clearly indicates that establishing a complete set of keywords is a key factor in handling ambiguity. However, according to one aspect of the invention, additional information beyond keywords are used to determine the meaning of an input query. This additional information makes it possible to use a collection of keywords far smaller than those required by conventional NL processing methods. Particularly, there are four layers of resources comprising a data dictionary that are used to relate an incoming query **202** to database object **204**A; i.e., cases **210**, keywords **209**, information models **211**, and database object values **208**B. These resources may be integrated through an extensible metadata representation method so that every piece of resources references to all other related resources in a semantically-based graphic. For instance, a keyword **209** points to the semantic subject(s) it refers to, which points in turn to entities, relationships, and items pertaining to the subject(s), and ultimately to database object values **204**B. The keywords **209** also connect to cases **210** involving them. The core of the reference dictionary (information model, initial keywords, and database structure) maybe, for example, a design-time product, developed by the analysts, designers, and users. Cases and additional keywords, metadata (e.g., changes to the information model) and database values may be added during operation of the system, and thus the system ages and evolves. A learning mechanism allows richer keywords and cases to provide more accurate performance. The reference dictionary enables a computer system to recognize a feasible region of interpretations of the input query **202** and evaluate them. The reference dictionary **208** also serves as the basis for interaction with the user (identifying needs and generating meaningful reference points) and acquisition of lessons (determining additional keywords and cases)—i.e., the reference dictionary may be used to assist the user in learning.

A reference dictionary according to one embodiment of the invention has four fundamental attributes, as compared to conventional systems: the reference dictionary according to one embodiment of the invention generates search-ready graphics-based representation of all four layers of resources; supports learning; simplifies keywords, and assures complete interpretations of natural queries. Regarding the last two points, the inclusion of information models **211** and case information **210** reduces the volume of keywords **209** needed to reduce the first two sources of ambiguity. For example, consider a natural articulation in the form of a short essay. If the essay consists of n words of which m are database objects or other recognized dictionary entries, there could be n/m words associated with each known term. These n/m words become the candidate keywords for the term. When including phrases (grouping of words), there could be, in theory, up to m*(n/m)! new keywords implied from the short essay. It is desired to increase the number m (hits) because the bigger m becomes, the fewer (exponentially) the possible groupings of words becomes, thus resulting in fewer new keywords to consider or to add to the dictionary.

Properly-developed information models **211** having rich semantics provide a large m for the initial design of keywords, and increase the chance of subsequent "hits" (their use in queries) in practice resulting in less ambiguity, less possible interpretations to search, and less new keywords needed. Case information **210** do not directly change m, but do help in resolving some ambiguity and hence still helps reducing the need for new keywords. Information models **211** and cases **210** represent a tightly structured, efficient kernel of meaning with which the users are familiar and tend to use more frequently in their articulation with respect to the particular databases. In addition, information models **211** and case information **210** also contribute to resolving another type of ambiguity. In particular, they identify the possible missing information for incomplete input, by examining the graphics of the reference dictionary. Therefore, a reference dictionary determines more accurately and quickly

than conventional systems a complete set of possible interpretations for queries articulated in a natural language format.

Search and Learn

Basic logic of a search and learn approach according to one embodiment of the invention will now be described. Given a reference dictionary R={C, K, M, D}, representing respectively the sets of cases C, keywords K, information models M, and database values D, a method for searching according to one embodiment of the invention is as follows:

Step 1: Identify all words and phrases in the input natural language query **202** that also belong to R. Denote this set of elements I (including possibly elements from K, M or D).

Step 2: Determine all possible, complete paths implied by I that span all input elements and query **202** and belong to the overall graphics of R. These paths might include additional elements inferred from the reference dictionary in order to complete the paths. A complete path includes elements (original or inferred) in M and D. Each path corresponds to a particular interpretation of the original query.

Step 3: Search for the best interpretation by using branch and bound methods when multiple possible solutions exist. If multiple possible solutions exist, the elements in C that are associated with elements of I are used to resolve the ambiguity.

Step 4: Map the result to the database query language. Obtain the results of query and confirm them with the user.

Note that a learning mechanism may be engaged to interact with the user whenever the result provided at each step is insufficient. The outcome of the learning is stored in the system **201** as new cases and keywords added to C and K, respectively. Also, note that each step allows for a wide range of possible strategies and algorithms to implement it.

Reference dictionary **208** may also be based on a metadatabase model described in more detail below with respect to FIG. **4**. In particular, a reference dictionary having a model that integrates four different types of enterprise metadata may be used. These metadata types include: database structure, semantic model, application, and software resource. The model may be used to form a core of the reference dictionary, and this core may be extended to include other three layers: keywords, cases and database values, and hence form the integrative (connected) structure of the reference dictionary. The other benefits of using this model includes its capability to incorporate rules and to support global query processing across multiple databases. A modeling system helps the development and creation of the metadatabase.

Reference Dictionary

A structure of an example reference dictionary **301** is shown in FIG. **3**. Each object in the figure represents either a table of metadata (in the case of square icon and diamond icon), or a particular type of integrity control rules (in the case of double diamond and broken diamond). These metadata include subjects and views, entity-relationship models, contextual knowledge in the form of rules, application and user definitions, database definitions and values, keywords, and cases.

Keywords, as noted above, are the natural words and phrases users use to refer to database objects and information model elements in natural articulation such as a natural language query. They could represent instances, operators, items (attributes), entities, relationships, subjects, and applications. A keyword according to one embodiment of the invention is defined as an ordered pair of (class, object). Classes include Application, Subject, EntRel (entity-relationship), Item, Value, and Operator; all of which are metadata tables shown in FIG. **3**. Objects are instances (contents) of these classes. Because a hierarchy of objects in the core structure of the reference dictionary is Item-EntRel-Subject-Application, an object can be identified by an ordered quadruple (Item name, EntRel name, Subject name, Application name). In the model, however, each object has a unique identifier, thus the ordered quadruple is not needed to uniquely identify each object. It should be understood that any method for identifying objects may be used.

Metadatabase Model

As discussed above with reference to FIG. **3**, system **101** may use a database to store keywords and other information. According to one embodiment of the invention, the database is metadatabase, which is well-known in the art of data and knowledge management tools. Metadatabase theory is described in more detail in a number of books and publications, including the book entitled *Enterprise Integration and Modeling: The Metadatabase Approach*, by Cheng Hsu, Kluwer Academic Publishers, Amsterdam, Holland and Boston, Mass., 1996. Also, metadatabase theory is described in the journal article by Hsu, C., et al. entitled *The Metadatabase Approach to Integrating and Managing Manufacturing Information Systems*, Journal of Intelligent Manufacturing, 1994, pp. 333–349. In conventional systems, metadatabase theory has traditionally been applied to manufacturing problems. A metadatabase contains information about enterprise data combined with knowledge of how the data is used. The metadatabase uses this knowledge to integrate data and support applications.

The metadatabase model as shown in FIG. **4** uses a structure that shows how a metadatabase system **402** provides an enterprice information model describing data resources of globally-distributed provider systems applications and their control strategy in the form of rules. These globally-distributed systems applications may be executed, for example, at one or more provider systems discussed above. The metadatabase system **402** communicates with service provider applications **409**, such as the three provider applications **409**A, **409**B, **409**C illustrated in FIG. **4**. The information model also includes knowledge regarding dynamics of information transfer such as "what and how" information is shared among local systems and under what circumstances it is used. The information model may be in the form of a metadatabase **401** having data items **404**, models **405**, rules **406**, software resources **407** and application and user information **408**.

Case-based Reasoning

A case in a case-based reasoning paradigm typically includes three components: problem definition, solution, and its outcome. New problems would use the problem definition to find the (best) matching cases and apply the associated solutions to them. The third component is useful when the domain knowledge is incomplete or unpredictable. In this research, the reference dictionary contains complete domain knowledge needed, thus, we expand the problem definition but drop outcome. The system uses cases to resolve ambiguity in the recognition of meaningful terms (i.e., user's natural terms that are included in the reference dictionary) in the input and to help determine the solution among multiple possible interpretations. Thus, the case structure includes case-id, case-type, choices, context, and solution. For the type of resolving term ambiguities, a set of known terms describes the context (for problem definition).

User's selection among possible choices of the meaningful term defines the solution. For the interpretation ambiguities type, a set of known elements of the information model describes the context, possible paths in the information model define the choices, and user's selection solution.

The resources (entries) of the reference model are connected in two ways. Recall that the structure shown in FIG. **3** may be a meta-schema representing the types and organization of all enterprise metadata. Thus, the elements of information models are metadata instances stored in some of the meta-entities (squares) and meta-relationships (diamonds) of the structure. These model elements are themselves connected internally in terms of their entity-relationship semantics. They are also connected externally to other types of resources including database values, keywords, and cases through the meta-schema. Keywords and cases are connected to information models and database values through particular meta-relationships. In other words, elements of information models (subjects, entities, relationships, and items) and keywords are linked to the database objects they represent. Therefore, the reference dictionary contains sufficient knowledge to determine the database objects involved and required for all queries defined sufficiently in information model elements or keywords.

Each sufficient statement corresponds to a complete and unique path (connection) of these elements and their corresponding database objects. (An SQL-like style database called MSQL may determine the shortest path when alternative paths exist. MSQL is discussed further in the journal entitled *The Model-Assisted Global Query System for Multiple Databases in Distributed Enterprises*, ACM Trans. Information Systems, 14:4, October 1996, pp. 421–470.) These complete paths represent the system's interpretations of users' queries. Ambiguity exists when a statement is insufficient such that there are conflicting interpretations—multiple paths leading to different database objects—for the query. These multiple paths could be the result either from providing incomplete elements or from providing conflicting elements implied in the input, or both. Such are the cases easily taking place with truly natural articulation of database queries.

There are several different ways to handle ambiguity. First, the system employs a rich information model to maximize the chance with which the users would naturally choose its elements in their articulation. Second, the system uses keywords to capture the words in the natural articulation that the information model misses. It is worthwhile to reiterate that the information model is the roadmap (together with database values) for developing keywords at design time. These keywords represent multiple natural equivalents of terms used in the information model (and database values). As explained above, a rich information model not only lessens the burden of "scoring hit" on the keywords, it also greatly reduces the complexity of adding new keywords at the run time. Third, it accumulates cases of usage from actual operation and applies them to resolve remaining ambiguity when both information model and keywords are insufficient for a query. Interaction with the users is the last measure to sufficiently close the loop and finish the job. The NLI systematically involves users to provide the final resolution of ambiguity and confirmation of the result if needed. This learning also generates new cases and keywords and enhances the old cases.

The reference dictionary contains different interpretations related to the query; interpretations being represented by multiple paths leading to different database objects for the query. Thus, the reference dictionary allows the system to definitively measure and identify ambiguities, based on the following graphical model of the logical structure of the metadata.

Added to this basic logic is particular search strategies and additional search knowledge. Search includes the identification of all possible paths-interpretations (when ambiguity exists) and the evaluation of them. A search algorithm could follow a branch-and-bound strategy to minimize the space of search (limiting the number of possible paths to search). The development of bounds and branching rules would require a way to evaluate a given path with respect to the original natural query. A method for eliminating paths may also be used; that is, the system could infer contradiction based on the information model and perhaps operational rules (contextual knowledge) the reference dictionary contains. A method of optimization—inferring goodness of fit for the user—could be performed. Information about user's profile, concerned applications, and past cases are among the metadata that could be used form a basis to identify the most probable interpretations. Elimination is more conservative, but robust, than optimization because elimination places safety (correctness) first.

Three progressive levels of search and learn capabilities may be used. First, the system develops the most efficient way to enumerate all possible interpretations for a natural query (i.e., design a powerful reference dictionary). Second, it will develop evaluation methods to improve the performance of search (i.e., eliminate more paths early in the search process) over the basic method. Finally, the system also proactively suggests the best interpretation for the user (i.e., develop case-based reasoning and other heuristics). Learning methods may accompany these search strategies at all levels. The above ideas are illustrated below with a brief example.

Logical Structure: A Graphical Representation

The logical structure of the reference dictionary may be represented graphically according to one embodiment of the invention. Interpretations of a natural language query are defined on a graph G (Definition 1 below), abstracted from the (content of) reference dictionary. Given a natural language query Q (Definition 2), the natural language interface performs interpretation in several steps. It first determines all recognized terms (Definition 3) contained in Q and their corresponding recognized vertex sets (Definition 4) in G. It then identifies all query images (Definition 5) of Q on G. Since Q may be ambiguous (e.g., incomplete and multi-valued mapping) to G, each of its recognized terms could correspond to multiple recognized vertices, resulting in multiple query images. Further, a recognized vertex may not always connect to other recognized vertices in a way covering a complete range of data semantics (database value, attribute, entity, and relationship) with a unique path. Therefore, it could have multiple semantic paths (Definition 6) covering multiple semantic domains (Definition 7). Taking these data semantics into account results in all possible query graphs for a query image, called feasible graphs (Definition 8). The refinement of feasible graphs leads to connected feasible graphs (Definition 9) and complete query graphs (Definition 10). A complete query graph represents an interpretation of the natural language query Q according to the logical structure G. The branch and bound algorithm searches implicitly all possible interpretations to determine the final query graph for execution.

Definition 1: The logical structure G is a graph <V, E>, where sets V and E are defined on the reference dictionary. In particular, V is a set of vertices of five types: subjects, entities, relationships, attributes, and values; and E is a set of their connection constraints (owner-member and peer-peer associations). Owner-member constraints belong to two types: subject-(sub)subject-entity/relationship-attribute-value and subject-attribute. Peer-peer constraints belong to three types: entity-entity, entity-relationship, and relationship-relationship.

Definition 2: A natural language query Q is a string of characters segmented by spaces.

Definition 3: A recognized term t_i of Q is a segment of Q matching some keywords in the reference dictionary or some vertices in graph G.

Definition 4: A recognized vertex set of a recognized term t_i, V_{t_i}, is a set of vertices of G that matches t_i. A member of V_{t_i} is a recognized vertex.

Definition 5: Given an n-recognized-term query where n< >0, a query image in graph G is a set of recognized vertices v_i where i=1, . . . , n and v_i V_{t_i}, respectively.

Definition 6: A semantic path of a recognized vertex v_i is a minimum set of vertices in G containing v_i that satisfies the following conditions: it contains a subject vertex and its vertices are connected. The vertices it contains, other than v_i, are all implied vertices by v_i to provide an interpretation (semantics) of v_i.

Definition 7: A semantic domain of a semantic path in graph G is a sub-graph of G that includes the semantic path, its member vertices, and edges connecting them.

Definition 8: A feasible graph of an n-recognized-vertex query image, where n< >0, is a collection of semantic domains sd_i where i=1, . .. , n and sd_i is a semantic domain implied by a semantic path of recognized vertex v_i of the query image.

Definition 9: A connected feasible graph is a connected sub-graph of G containing the feasible graph and a collection of entity/relationship vertices and edges in this graph. This collection is minimally sufficient to connect entity/relationship vertices of the feasible graph.

Definition 10: A query graph is a connected feasible graph. It represents an interpretation of a query that is definitive for a database query such as QBE and SQL to process the query.

Consider a Computer-Integrated Manufacturing system including three databases: order processing, process planning, and shop floor control. Suppose the system used the well-known TSER—Two Stage Entity-Relationship method (whose constructs include Application-Subject-Context-Entity-Relationship-Item described more in detail in the journal article entitled *Paradigm Translations in Integrating Manufacturing Engineering Using a Meta-Model: the TSER Approach*, by Cheng Hsu et al., J. Information Systems Engineering, 1:1, September 1993, pp. 325–352) to develop their information models and created a reference dictionary. These models became metadata instances stored in certain meta-entities and meta-relationships according to FIG. **1**. The system had also included proper keywords and cases and stored them along with all database objects. Now, suppose a user made the following sample request, a problem of NLI: An enterprise user of a Computer-Integrated Manufacturing database could query the system such as: "I have asked you many times and I still have not heard a clear answer. Listen, I would like to know the status of my order. I want to know what models you have started working onand how many of them you have completed. I placed this order, I believe, sometime around the 20th of last December. Could you please help? Thanks."

A text scanning and information retrieval algorithm (described further below) may generate the result shown in Table 1 below.

TABLE 1

Recognized Terms and Possible Interpretations

Terms	Possible Interpretations
status	(Item, SHOP_FLOOR.STATUS)
order	(Subject, ORDER)
	(EntRel, ORDER_PROCESSING.ORDER)
models	(EntRel, ORDER_PROCESSING.PART)
	(EntRel, SHOP_FLOOR.PART)
	(EntRel, PROCESS_PLAN.PART)
	(Subject, PART)
started working on	(Value, SHOP_FLOOR.STATUS)
completed	(Item, SHOP_FLOOR.NUM_COMPLETED)
20th of last December	(Value, ORDER_PROCESSING.DATE_DESIRED)

Ambiguity exists at terms "Order" and "Models" because each has multiple interpretations identified from the reference model. Otherwise, the result is definitive (forming a unique overall path) and could be mapped to the Metadatabase Query Language (MQL).

In this particular query, the information model would be sufficient to sort out the ambiguity and suggest a unique, optimal interpretation for these terms, and hence for the natural query. Still, cases could also be used either to confirm or to assist the resolution of ambiguity. However, there may be another kind of ambiguity in the input; the user indicated "around" 20th of last December in the original natural query. Because of this ambiguity, the user may find the final answer less than satisfactory. The system generally would have no method for interpreting correctly this piece of input since the user herself was ambivalent about it. There may be, in this instance, no proper solution other than to leaving the interpretation to the user. The final answer (based on 12/20/1999) may represent the best point estimation for the user's fuzzy interval of possibilities. The user would be presented with a chance to comment on this estimate and to request new dates for another query if wanted. Interaction with the user concerning this estimate, in its own right, could add valuable cases to the system so that it could provide more help for uncertain users when the term "around" is encountered in a following query, which is also a part of learning.

The following is an example algorithm for identifying meaningful terms:

```
Let m = number of words of input string
Let OrderedTerm = ( )
Let n = 1
Do while (n <= m)
Search in OrderedTerm for Input SubString starts at n-th position
If (Term is found with length 1),
Then
Let n = 1 + 1
Else
Search in the dictionary for Input SubString starts at n-th position
If (Term is found with length 1),
Then
Let Term = 1-words string
Retrieve meaning sets of the Term
```

-continued

```
Insert OrderedTerm with the Term
Let n = 1 + 1
Else
Let n = n + 1
End if
End if
End do while
```

The application of cases—i.e., matching a query with a case—is based on the vector space model as is known in the art. Two binary vectors represent a case (C) and a query (Q); and their COSINE measure indicates the goodness of fit. Below is an algorithm applying cases to the resolution of ambiguity in terms.

Retrieve cases containing the similar situation

If there are retrieved cases,

```
Then
Let similar_value = 0
For each retrieved case
Let the base set for meaning space = set of meaning of terms of
query
For each meaning of case
Update the base set
End for each
Form a term space as an ordered n-tuples of terms /* n is a number
of
terms in the base set */
Form a binary vector for query (Q) corresponding to the meaning
space
Form a binary vector for case (C) corresponding to the meaning space
Compute COSINE similarity: COSINE (Q, C) = (Q.C)/(|Q|)(|C|)
If COSINE (Q, C) > similar_value,
Then
Let the solution case = the current case
Let similar_value = COSINE (Q,C)
End if
End for each
If similar_value > accepted_value,
Then
Determine the meaning of the ambiguous term from the solution case
End if
End if
```

Finally, when ambiguities are resolved, the complete list of path information becomes the sufficient input for the underlying database query language. In this example, we show the MQL statements, which may function on top of a standard SQL facility.

FROM OE/PR WO_SEQ GET STATUS

FROM SUBJECT ORDER GET CUST_ORDER_ID, DATE_DESIRED,

OD_STATUS, CUST_ID, ORDER_LINE_ID, $PART_{13}$ID, QUANTITY,

DATE_SCHED, OI_STATUS, DESCRIPTION, COST

FROM OE/PR WK_ORDER GET NUM_COMPLETED

FOR STATUS='START WORKING ON'

AND DATE_DESIRED='12/20/1999';

The mapping performs processing in order to determine the GET lists and some conditions (such as AND/OR). However, at this point, the reference model would have all information needed to perform the query.

A complete sequence of graphs generated from this list is shown in FIGS. **7**–**17** and is described in more detail below. Query images and feasible graphs are intermediate vertices while query graphs are terminal vertices.

According to one embodiment of the invention, a branch and bound algorithm is provided, including the optimal search strategies and evaluation rules. A reliable idea is to use a method of elimination; that is, the system could infer contradiction based on the information model and operational rules (contextual knowledge) to remove certain interpretations. The method of optimization—inferring goodness of fit for the user—could be another possibility. Information such as user profile, concerned applications, and past cases is among the metadata that could help identify the most probable interpretations. A method is provided to enumerate all possible interpretations for a natural query, the optimal evaluation methods to improve the performance of search, and case-based reasoning and other heuristics to enhance user-feedback and assure closure.

Branch and Bound Algorithm

Consider the basic algorithm shown below. An interpretation problem is formulated as an optimization problem with objective function z(t) where t is a terminal vertex. A goal is to minimize z(t) with respect to graph G. An evaluation function LB(v) finds the lower bound for an intermediate vertex v, so that the search could either fathom all paths starting with v, or pick the most promising v to explore in more detail. According to one embodiment of the invention, the evaluation embodies operating rules and heuristics. Note that the search is complete; i.e. there always exists an optimal solution in the search space and the search will find it.

Below is an example branch and bound algorithm that may be used according to one embodiment of the invention:

```
    Let current_best_LB = MAX_NUMBER
Let current_LB = 1
Insert root into priority queue//the least lower bound element is always in
the first position
While (current_best_LB >= current_LB) {
    Let current_vertex = the first element of priority queue
    Remove the first element from priority queue
    Let min_successor_LB = MAX_NUMBER
    Let successor_set = branch (current_vetrex)
    //branch( ) return with successor's LB or z( )
    For each successor in successor_set {
        If (successor is not a terminal vertex) {
            Insert successor into priority queue}
        If (LB(successor) < min_successor_LB) {
            Let min_successor_LB = LB(successor)}
    }
    For each successor in successor_set {
        If (min_successor_LB = LB(successor) {
            If (successor is not a terminal vertex) {
                Let current_LB = min_successor_LB }
            else {
                If (min_successor_LB <= current_best_LB ) {
                    If (min_successor_LB < current_best_LB ) {
                        Remove all elements of solution set }
                    Insert successor into solution set
                    Let current_best_LB = min_successor_LB }
            }
        }
    }
}
```

Case-Based Reasoning

The basic design of cases is discussed above with respect to the reference dictionary. The application of cases—i.e., matching a query with a case—is based on the vector space model. Two binary vectors represent a case (C) and a query (Q); and their COSINE measure indicates the goodness of fit.

Below is an example procedure for measuring Cosine similarity:

```
Procedure cosine_simialarity (Q, C) {
/* Q and C are in terms of recognized vertices */
    Let base_set = Q ∪ C
    Let U=binary_vector(Q)
    /* function binary_vector (Q) returns a binary vector based on
    base_set*/
    Let V=binary_vector(C)
    /* function binary_vector (C) returns a binary vector based on
    base_set*/
        Return (U.V)/(|U|)(|V|)
    }
```

According to another embodiment of the invention, an NL system processes an input according to a series of steps. For example, the NL system may accept an input (such as a query string), determine recognized terms and their vertices, and determine minimal recognized terms and their minimal recognized vertices. The NL system may then search for minimum cost query graphs, eliminate redundant solutions and determine complete solutions, and translate these complete solutions to query language statements. The following are a series of steps and algorithms that may be used to interpret an input database query.

Main Algorithm:

Step 1: Determine recognized terms and their recognized vertices (Algorithm 1)

Step 2: Determine minimal recognized terms and their minimal recognized vertices (Algorithm 2)

Step 3: Search for minimum cost query graphs (Algorithm 3)

Step 4: Remove redundant solutions (Algorithm 4)

Step 5: Determine complete solutions (Algorithm 5)

Step 6: Translate to SQL statement

Algorithm 1: Determine recognized terms and their recognized vertices

```
Determine from input string, an ordered N-tuple I = (w1, w2, . . . , wN)
        where w is a word and N is a number of words
wordIndex = 1
        While (wordIndex < N) {
    Search in list of existing recognized terms
            for a string constructed from (wwordIndex , wwordIndex+1 , . . .
            ,
    wwordIndex+L−1 ) with one space as a delimiter where L is the length
    of maximum-length matching string
If (substring is found) {
    wordIndex = wordIndex + L}
Else {
    Search in the dictionary
        for a string constructed from (wwordIndex , wwordIndex+1 , . . . ,
                wwordIndex+L−1 ) with one space as a delimiter where
        L is the length of maximum-length matching string
    If (substring is found) {
        Insert founded substring into term list
        Insert meanings of found substring into meaning list
        wordIndex = wordIndex + L }
    Else {Let wordIndex = wordIndex + 1}
}
    }
```

Algorithm 2: determine minimal recognized terms and their minimal recognized vertices

```
For each recognized term {
    If (number of its recognized vertices) > 1 {
        For each recognized vertex {
        If (the recognized vertex belongs to a recognized vertex of the
        term)
            {Eliminate the recognized vertex}
    }
}
}
For each recognized term {
    If (the number of recognized vertices > 1){
        For each recognized vertex {
            If (the recognized vertex is not attribute or value vertex
            and it belongs to a recognized vertex of other terms) {
                Eliminate the recognized vertex
        }
    }
}
If (all recognized vertices of that term are eliminated) {
    Eliminate this term
}
}
```

FIG. **5** shows an example complete search graph having a number of query images, query graphs, and feasible graphs arranged in a hierarchical structure.

Algorithm 3: Search for minimum cost query graphs (query graph containing minimum vertices)

```
    Let current_best_LB = MAX_NUMBER
Insert root into priority queue//the least lower bound element is always in
the first position
Let current_vertex = the first element of priority queue
Remove the first element from priority queue
Let current_LB = LB(the first element of priority queue)
While (current_best_LB >= current_LB) {
    Let min_successor_LB = MAX_NUMBER
    Let successor_set = branch (current_vetrex) //Algorithm 3.1
    //branch( ) return with successor's LB or z( )
    For each successor in successor_set {
        If (successor is not a terminal vertex) {
            Insert successor into priority queue}
            If (LB(successor) < min_successor_LB) {
                Let min_successor_LB = LB(successor)}
    }
    For each successor in successor_set {
        If (min_successor_LB = LB(successor) {
            If (successor is not a terminal vertex) {
                Let current_LB = min_successor_LB
            } else {
                If (min_successor_LB <= current_best_LB ) {
                    If (min_successor_LB < current_best_LB ) {
                        Remove all elements of solution set }
```

-continued

```
                    Insert successor into solution set
                    Let current_best_LB = min_successor_LB }
                }
            }
        }
        if (the priority queue is not empty) {
            Let current_vertex = the first element of priority queue
            Remove the first element from priority queue
        } else {
            current_lb = MAX_NUMBER
        }
}
```

Algorithm 3.1: Branching procedure

```
Procedure branch (current vertex) {
    Let successor set is an empty set
    Case: the current vertex is the root {
        Determine query images
        For each query image {
            Determine LB of query image
        }
        Add all query images to successor set
    }
    Case: the current vertex is a query image {
        Determine feasible graphs
        For each feasible graph {
            Determine LB of feasible graph
        }
        Add all feasible graphs to successor set
    }
    Case: the current vertex is a feasible graph {
        Determine query graphs
        For each query graph {
            Determine cost of query graph
        }
        Add all query graphs to successor set
    }
    Return successor set
}
```

Algorithm 5: Determine complete solutions

```
            for each solution {
                determine target attribute list
                determine er set
                determine join condition
                determine selection condition
            }
```

Application of the above algorithms to an input query is more clearly illustrated by the following example:

Example Query: Get order_id, model, wo_quan, and num_completed of John Smith's orders.

First, the NL system determines recognized terms and their vertices. Below are example recognized terms and vertices corresponding to the example query above.

Term	Vertex
ORDER_ID	I sfcl_17
MODEL	I opsl_100
	I ppsl_54
	I sfcl_5
WO_QUAN	I sfcl_6
NUM_COMPLETED	I sfcl_20
JOHN SMITH	V opsl_90|John Smith
ORDERS	S ORDER

From the recognized terms and vertex sets are determined minimal recognized terms and corresponding minimal vertex sets. Below is an example set of minimal recognized terms and minimal vertex sets according to the example.

Term	Vertex
ORDER_ID	I sfcl_17
MODEL	I opsl_100
	I ppsl_54
	I sfcl_5
WO_QUAN	I sfcl_6
NUM_COMPLETED	I sfcl_20
JOHN SMITH	V opsl_90|John Smith
ORDERS	S ORDER

Applying algorithm 3, above, the query graph having the minimum vertices is located. A search graph corresponding to the example appears in FIG. **6**.

Result from Branching Procedure

Query Images and their lower bounds:

Query Image 1: {I sfcl_**17**, I opsl_**100**, I sfcl_**6**, I sfcl_**20**, V opsl_**90**\|John Smith, S ORDER}

LB=10

A corresponding query image is shown in FIG. **7**.

Query Image 2: {I sfcl_**17**, I ppsl_**54**, I sfcl_**6**, I sfcl_**20**, V opsl_**90**\|John Smith, S ORDER}

LB=11

A corresponding query image is shown in FIG. **8**.

Query Image 3: {I sfcl_**17**, I sfcl_**5**, I sfcl_**6**, I sfcl_**20**, V opsl_**90**\|John Smith, S ORDER}

LB=10

A corresponding query image is shown in FIG. **9**.

Feasible Graphs and their lower bounds:

Feasible Graph 1of Query Image 1:

```
        ErSet =
{ WORK _ORDER, PART, ORDER_ITEM, ORDER_HEADER, CUSTOMER }
        Item_Er_set = {( I sfcl_6, WORK_ORDER), (I opsl_100,
ORDER_ITEM), (I sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
LB = 10
```

A corresponding feasible graph is shown in FIG. **10**.
Feasible Graph 2of Query Image 1:

```
ErSet =
{ WORK_ORDER, PART, ORDER_ITEM, ORDER_HEADER,
CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I opsl_100, PART), (I
sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
LB = 10
```

A corresponding feasible graph is shown in FIG. **11**.
Feasible Graph 1of Query Image 3:

```
ErSet =
{ WORK_ORDER, PART, ORDER_ITEM, ORDER_HEADER,
CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I sfcl_5, OPERATOR), (I
sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
LB = 11
```

A corresponding feasible graph is shown in FIG. **12**.
Feasible Graph 2of Query Image 3:

```
ErSet =
{ WORK_ORDER, PART, ORDER_ITEM, ORDER_HEADER,
CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I sfcl_5, PARTS_AVAIL), (I
sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
LB = 11
```

A corresponding feasible graph is shown in FIG. **13**.
Feasible Graph 3of Query Image 3:

```
ErSet =
{ WORK_ORDER, PART, ORDER_ITEM, ORDER_HEADER,
CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I sfcl_5, WORK_ORDER),
(I sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
LB = 10
```

A corresponding feasible graph is shown in FIG. **14**.
Query Graphs and their minimum vertices:

```
ErSpSet = { WORK_ORDER, PART, ORDER_ITEM,
ORDER_HEADER, CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I opsl_100,
ORDER_ITEM), (I sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
Z = 10
```

A corresponding query graph is shown in FIG. **15**. Query Graph 1of FG 12:

```
ErSpSet = { WORK_ORDER, PART, ORDER_ITEM,
ORDER_HEADER, CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I opsl_100, PART), (I sfcl_17,
WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
Z = 10
```

A corresponding query graph is shown in FIG. **16**. Query Graph 1of FG 33:

```
ErSpSet = { WORK_ORDER, PART, ORDER_ITEM,
ORDER_HEADER, CUSTOMER }
Item_Er_set = {( I sfcl_6, WORK_ORDER), (I sfcl_5, WORK_ORDER),
(I sfcl_17, WORK_ORDER), (I sfcl_20, WORK_ORDER) }
Value_Er_set = {( V opsl_90|John Smith, CUSTOMER)}
Z = 10
```

A corresponding query graph is shown in FIG. **17**.
Solution Set:
Solution 1: Query graph 1of FG33
Solution 2: Query graph 1of FG12
Solution 3: Query graph 1of FG11
Solution Set after removing redundancy:
Solution 1: Query graph 1of FG33
The Complete solution is:

```
Attribute list = { WORK_ORDER.ORDER_ID,
WORK_ORDER.PART_ID, WORK_ORDER.WO_QUAN,
WORK_ORDER.NUM_COMPLETED}
ER list = { WORK_ORDER, PART, ORDER_ITEM,
ORDER_HEADER, CUSTOMER }
JC = "PART.PART_ID =WORK_ORDER.PART_ID and
WORK_ORDER.WO_ID=WO_SEQ.WO_ID and
WORK_ORDER.ORDER_ID=ORDER_ITEM.ORDER_LINE_ID and
ORDER_ITEM.CUST_ORDER_ID=
ORDER_HEADER.CUST_ORDER_ID"
SC = "(CUSTOMER.CUST_NAME = 'John Smith')"
```

An SQL statement corresponding to the complete solution above is:

```
SELECT WORK_ORDER.ORDER_ID,
WORK_ORDER.PART_ID, WORK_ORDER.WO_QUAN,
WORK_ORDER.NUM_COMPLETED
FROM WORK_ORDER, PART, ORDER_ITEM, ORDER_HEADER,
CUSTOMER
WHERE ORDER_ITEM.ORDER_LINE_ID=WORK_ORDER.ORDER_ID and
ORDER_ITEM.PART_ID=PART.PART_ID and
ORDER_ITEM.CUST_ORDER_ID =
ORDER_HEADER.CUST_ORDER_ID and
ORDER_HEADER.CUST_ID=CUSTOMER.CUST_ID and
(CUSTOMER.CUST_NAME = 'John Smith');
```

According to one embodiment of the invention, a search-and-learn approach is described in more detail below. A particular enterprise database (or distribution of databases) necessarily has only a finite number of database objects, including data types, data instances, and database computational operators (also referred to as database values). They give rise to only a finite number of possible permutations. All meaningful queries must ultimately refer to these database objects and the articulation of any queries must correspond to some of these permutations. When the system identifies the permutations implied, it has interpreted the query. Then, it has to be able to map the query to the underlying database query language (such as SQL) without having to understand linguistically the human meaning of

the words and phrases used. Generic natural language processing is fundamentally different since the possible permutations in the latter case are virtually infinite. Of course, even the relatively small, finite number of database objects could in theory generate a large number of possible permutations. Moreover, truly natural articulation could plausibly defy the pre-determined conventions of the databases in the following basic ways:

(1) Use other words and phrases than those captured in the database to refer to these basic objects (e.g., 4^{th} of July instead of 7/4 and "buyer" rather than "customer");
(2) Leave its reference to these objects undefined (e.g., the precise nature and condition of the reference or objects); and
(3) Be implicit in the articulation (e.g., incomplete or convoluted input).

All these cause ambiguity, and the system might not be able to identify any database object or permutation at all. In the worst case, the system would have to search the entire space of possible permutations and evaluate each of them for the user to determine which one is correct. From this point on, every (additional) database object the system recognized would serve to eliminate certain space from the search and narrow down the remaining possible interpretations. In the best case, the system would quickly identify a single complete permutation and determine immediately a unique, correct interpretation for the user. In this context, one could consider the understanding-based approaches as the best case all the time. But the implicit enumeration-evaluation approach assumes the more realistic middle road, aiming to avoid the worse case and approaches as closely as possible the best case. The task, rather, is one of developing good search methods to represent and characterize distinct possibilities and to optimize interpretations. The challenge is to analyze ambiguity and turn it into a problem that can be handled systematically.

Various concepts related to search and learn is described in more detail below. Section 1 introduces a basic model of the reference dictionary as a graph of complete interpretations of natural language queries for use in enterprise databases and as a resource for resolving ambiguity. The graph incorporates four resource layers (database values, information models, keywords, and cases). Section 2 describes a search and learn process that identifies a single and complete interpretation (path) for a query by identifying all possible complete paths. This spans all recognized elements of the query as well as intermediate elements in this graph to complete the path, and finally evaluates them. When ambiguity (multiple paths) exists, users are engaged in dialogue in determining the correct path. Moreover, users are engaged in dialogue in evaluating the result of the query and acquiring new keywords. These interactions with users are denoted as learning because their outcomes become new cases and keywords added to the reference dictionary to further improve effectiveness and efficiency of the system.

1. A Basic Graph Model of the Reference Dictionary

A basic graph model of the reference dictionary is designed to serve three purposes: (1) to include all possible interpretations for a natural language query suitable for evaluation, (2) to provide a basis for evaluation (resolution of ambiguity), and (3) to support learning. It integrates four resource layers (database values, information models, keywords, and cases) into a single graph model. In this graph, therefore, every element can refer to all related elements.

The design adopts the entity-relationship data model as a conceptual model because it is both sufficient for modeling the semantics of database objects and other database models can be translated into it. Therefore, in this graph, vertices constitute an entity or a relationship (an associative integrity constraint) and edges are connections based on their relationship constraints.

The design follows three steps. The first step models database objects as a database graph. The second step models indirect references to these objects. The final step models additional resources for evaluation.

1.1 A Database graph

Given S, R, A, and D respectively are the finite sets of entities, relationships, attributes, and values; and $S \cup R \cup A \cup D$ is a set of enterprise database objects (O). A database graph is a graph <V, E> where $V=O$, $X=S \cup R$, $Y=S \cup R \cup A$ and $E \subseteq (X \times Y) \cup (A \times D)$. E satisfies one of the following properties:

(1) For all $a \in A$, if a is an attribute of $x \in X$, then edge $\{x, a\} \in E$.
(2) For all $r \in R$, if r represents a referential or an existence integrity constraint between s_1 and $s_2 \in S$, then edge $\{s_1, s_2\} \in E$.
(3) For all $r \in R$, if r represents an associative integrity constraint between s_1 and $s_2 \in S$, then edges $\{r, s_1\}$ and $\{r, s_2\} \in E$.
(4) For all $d \in D$, if d is a value of $a \in A$, then edge $\{a, d\} \in E$.

1.2 Indirect References to Database Objects

Indirect references to database objects are three extensions of the database graph as follows.

(1) Given H is an entity set of high-level hierarchical objects, $V=O \cup H$, and $E \subseteq (X \times Y) \cup (A \times D) \cup (H \times (H \cup Y))$. Edges $(H \times (H \cup Y))$ satisfy the following properties:
  (1.1.) For all h_1 and $h_2 \in H$, if h_1 is part of h_2, then edge $\{h_1, h_2\} \in E$.
  (1.2.) For all $h \in H$, if h is described by $\{a_1, a_2, \ldots, a_n\}$ for $a_i \in A$, $i=1, \ldots, n$; then edges $\{h, a_1\}, \{h, a_2\}, \ldots, \{h, a_n\} \in E$;
  (1.3.) For all $h \in H$, if h is described by $\{x_1, x_2, \ldots, x_n\}$ for $x_i \in X$, $i=1, \ldots, n$; then edges $\{h, x_1\}, \{h, x_2\}, \ldots, \{h, x_n\} \in E$.
(2) Given F is an entity set of operations (such as formulas and graphic presentations) upon a set of attributes, $M=O \cup H \cup F$, $V=M$, and $E \subseteq (X \times Y) \cup (A \times D) \cup (H \times (H \cup Y)) \cup (A \times F)$. Edges $(A \times F)$ satisfy the property: for all $f \in F$, if f is computed from $\{a_1, a_2, \ldots, a_n\}$ for $a_i \in A$, $i=1, \ldots, n$; then edges $\{f, a_1\}, \{f, a_2\}, \ldots, \{f, a_n\} \in E$
(3) Given K is an entity set of keywords that refers to an element of M, $V=M \cup K$, and $E \subseteq (X \times Y) \cup (A \times D) \cup (H \times (H \cup Y)) \cup (A \times F) \cup (K \times M)$. Edges $(K \times M)$ satisfy the property: for all $m \in M$, if $k_1, k_2, \ldots, k_n$ for $k_i \in K$, $i=1, \ldots, n$ refer to m; then edges $\{m, k_1\}, \{m, k_2\}, \ldots, \{m, k_n\} \in E$.

1.3 Additional Resources for Evaluation

Additional resources for evaluation are three extensions of the graph in Section 1.2 above as follows.

(1) Given T is an associative relationship set of contextual knowledge among elements of H, $V=M \cup K \cup T$, and $E \subseteq (X \times Y) \cup (A \times D) \cup ((H \cup T) \times (H \cup Y \cup T)) \cup (A \times F) \cup (K \times M)$. Edges $((H \cup T) \times (H \cup Y \cup T))$ satisfy the property: for all $t \in T$, if t represents associations among $h_1, h_2, \ldots, h_n$ for $h_i \in H$, $i=1, \ldots, n$; then edges $\{t, h_1\}, \{t, h_2\}, \ldots, \{t, h_n\} \in E$.

(2) Given U is an entity set of user profiles, P is an associative relationship set of elements of U and H, $V=M\cup K\cup T\cup U\cup P$, and $E\subseteq(X\times Y)\cup(A\times D)\cup((H\cup T)\times(H\cup Y\cup T))\cup(A\times F)\cup(K\times M)\cup((H\cup P)\times(U\cup P))$. Edges $((H\cup P)\times(U\cup P))$ satisfy the property: for all $p\in P$, if p represents an association between $h\in H$ and $u\in U$, then edges $\{h, p\}$ and $\{u, p\}\in E$.

(3) Given C is a entity set of cases and Z is an associative relationship set of elements of C and M, $V=M\cup K\cup T\cup U\cup P\cup C\cup Z$ and $E\subseteq(X\times Y)\cup(A\times D)\cup((H\cup T)\times(H\cup Y\cup T))\cup(A\times F)\cup(K\times M)\cup((H\cup P)\times(U\cup P))\cup((M\cup Z)\times(C\cup Z))$. Edges $((M\cup Z)\times(C\cup Z))$ satisfy the property: for all $z\in Z$, if z represents association between $c\in C$ and $\{m_1, m_2, \ldots, m_n\}$ for $m_i\in M$, $i=1, \ldots, n$; then edges $\{c, m_1\}$, $\{c, m_2\}$, . . . , $\{c, m_n\}\in E$

2. A Search and Learn Process

An ambiguous (incomplete) input is an input that corresponds to more than one interpretation (represented by more than one path in the graph defined in Section 1.1 and Section 1.2(1), which this graph denotes as G). The ambiguity stems from a meaningful term (a word or phrase) in the input which refers to more than one element in the graph G. Therefore, there exists more than one target graph for each ambiguous input. A target graph is a subset of the set of vertices in the graph G and its vertices corresponds to each recognized term of the input. Each of these target graphs may determine more than one query graph. A query graph is the connected graph of a target graph. It represents the complete path for each interpretation. A complete path is the path that spans all vertices of its target graph and may include intermediate vertices in the graph G to complete it. An unambiguous (complete) input is the input that corresponds to a unique complete path in the graph G.

The basic idea of search is to enumerate all possible paths (query graphs) and to evaluate (rank) them to determine the best path (interpretation) for a query. In the above analysis, however, there are two sources of ambiguity: multiple meanings of a term in the input and multiple paths determined from a target graph. These sources can lead to a huge number of complete paths in the graph G. So, it is inefficient to generate all of these possible paths. The primary goal is to place correctness first while generating the minimal set of possible complete paths (i.e., correctness and efficiency are our goals). In particular, the goal is to determine the minimal set of candidate interpretations (minimize number of possible complete paths) without excluding any meanings of terms from the input and suggesting the best interpretation for the user.

To minimize a number of possible paths under the correctness constraint, we have to minimize a number of terms representing the input and minimize a number of enumerated target graphs. Domain-independent heuristics based on object hierarchy in G are used to minimize the number of terms. The number of target graphs can be minimized in two steps. In the first step, consistency constraints (such as functional dependencies, and business rules) in the graph G and its extension defined in Section 1.3(1) are used to eliminate the target graphs with a conflicting selection condition. The second step minimizes the number of vertices for each target graph and eliminates the redundant target graphs by using domain-independent heuristics based on the object hierarchy defined in the graph G.

The next task is to obtain a minimal set of query graphs corresponding to the obtained set of target graphs. The query graph of a target graph is the connected graph that spans all vertices of the target graph and intermediate vertices of G to connect them. In addition, the number of connections is minimal. This bases on the observation that the semantically related terms tend to be near each other in G. Note that this problem can be modeled as a discrete optimization problem to determine the shortest path connecting all elements of a target graph. If there exists more than one target graph corresponding to a query, the next query graph will be generated based on the accumulated knowledge from the generation of the past query graphs. However, the new set of query graphs may contain identical elements. These can be reduced to obtain a minimal set of query graphs.

The last task of the search process is to evaluate candidate query graphs (interpretations) to suggest the best interpretation for the user. This task follows three steps. The first step uses heuristics to measure the semantic relatedness of term meanings based on the same observations used in generating query graphs. Thus, the query graphs with the shortest path length (minimum number of edges) have the greatest semantic relatedness. The next two steps are employed if the result of the previous step is ambiguous. The second step identifies the solution based on high-level object-usage history of users (the extension of graph G defined in 1.3(2)). The last step identifies the solution based on the past cases (the extension of graph G defined in 1.3(3)).

However, the above process may still lead to three results: no interpretation, one interpretation, and multiple interpretations. In case of multiple interpretations, the system will resolve this ambiguity prior to presenting candidate interpretations back to the user to choose the correct one. The result of the best interpretation (either an interpretation suggested by the system or user's own chosen interpretation) is that which is evaluated by the user. If the user does not accept the result or there is no interpretation of the query, the system will engage the user in a dialogue relying on the learning mechanism.

A detailed exemplary algorithm following the above basic logic is outlined below.

Given an input (I_0) is an ordered n-tuple $(w_1, w_2, \ldots, w_n)$ of words, i.e., $I_0=(w_1, w_2, \ldots, w_n)$, where n is the number of words in a natural language query.

Step 1: Identify all meaningful words or phrases in the input and their meanings (elements in G).

Step 1.1: Determine a set of all meaningful words and phrases in the input (we denote this set as I_1) such that these words and phrases are possibly elements of the graph G and its extensions defined in Section 3.1.2(2) and (3). Therefore, we have $I_1=(t_1, t_2, \ldots, t_k)$ such that $k\leqq n$, where k is the number of terms formed from n words of I_0. If there is no meaningful word or phrase in the query (k=0), then go to the learning mechanism to acquire new keywords and redo this step.

Step 1.2: Determine an ordered k-tuple $M_0=(S_1, S_2, \ldots S_k)$ such that S_i is a set of set of meanings of its correspondent $t_i\in I_1$ where $i=1, \ldots, k$; and S_i is determined based on how they are used in G (i.e., all elements I_1 that refer to elements of the graph defined in Section 3.1.2(2) and (3) are mapped into G).

Step 1.3: Determine M_1 and I_2 such that M_1 represents the minimal meaning of M_0 (M_1 contains no redundant sets of set of meanings) and 12 corresponds to M_1 (i.e., $M_1=(S_1, S_2, \ldots, S_{k^*})$ and $I_2=(t_1, t_2, \ldots, t_{k^*})$ where k^* is a minimal k and $k^*\leqq k$).

Step 2: Determine all possible and complete paths (interpretations) implied by I_2.

Step 2.1: Determine the set of ordered pairs (TG_i, $<SC>_{TGi}$), we denote this set as TG_{M1}, such that TG_i is a target graph, i=1, . . . , n, n is the number of enumerated target graphs determined from M_1, and $<SC>_{TGi}$ is a selection condition of TG_i; such that for all elements of TG_i in vertices D of the graph G:

If there exist d_1, . . . , d_m in the same element of A (denoted by a) in the graph G (defined in Section 3.1.1), then add $a=d_1 \vee . . . \vee a=d_m$ to $<SC>_{TGi}$.

If there exist d_1, . . . , d_k in the different elements of A (a_1, . . . , a_k) in the graph G, then add $a_1=d_1 \wedge . . . \wedge a_m=d_m$ to $<SC>_{TGi}$.

Step 2.2: Determine the set of consistent target graphs (CTG_{M1}) such that for each $TG_i \in CTG_{M1}$, its selection condition $<SC>_{TGi}$ does not conflict any of consistency constraints in defined in the graph of G and its extension in Section 3.1.3(1), i=1, . . . , n*, $n^* \leqq n$, and n is defined in Step 2.1. If there is no consistent target graph ($CTG_{M1}=\{\}$), then go to the learning mechanism.

Step 2.3: Determine the minimal elements for each $TG \in CTG_{M1}$ such that for all elements v_i and v_j in TG where i, j=1, . . . , k*, $i \neq j$ and k* is defined in step 1.3; there does not exist v_i such that $v_i \in v_j$.

Step 2.4: Determine the minimal set for CTG_{M1} (we denote this set as MTG_{M1}) such that for all TG_i and TG_j in MTG_{M1} that $<SC>_{TGi}=<SC>_{TGj}$ where i, j=1, . . . , n**, $i \neq j$, $n^{**} \leqq n$, and n* is defined in step 2.2; there does not exist TGi such that $TG_i \subseteq TG_j$.

Step 2.5: Determine the set of query graphs for MTG_{M1} (we denote this set as QG_{M1}) such that the query graph for each target graph TG in MTG_{M1} QG(TG)=<VQ, EQ> is the connected graph that $TG \subseteq VQ$, edges EQ are elements of edges G, and the number of EQ is minimal. Note that the QG(TG) can be obtained by using an optimization method. If there exists one query graph, then go to Step 4.

Step 2.6: Determine the minimal set of query graphs for QG_{M1} (we denote this set as MQG_{M1}) such that for all QG_i and QG_j in QG_{M1} that $<SC>_{QGi}=<SC>_{QGj}$ where i, j=1, . . . , n***, $i \neq j$, $n^{***} \leqq n^{**}$, and n** is defined in step 2.4; there does not exist QG_j such that $QG_i=QG_j$. If there exists one interpretation (n***=1) in MQG_{M1}, then go to Step 4.

Step 3: Evaluate candidate interpretations (query graphs).

Step 3.1: Identify the set of minimum-length query graphs (we denote this set as LQG_{M1}) such that every QG in LQG_{M1} is an element of MQG_{M1} and the QG is the minimum path-length query graph among a query graph $QG_i \in MQG_{M1}$ where i=1, . . . , n***, $n^{***} \leqq n^{**}$, and n** is defined in step 2.4. If there exists one interpretation, then go to Step 4.

Step 3.2: Determine the set of the most likely interpretation (when ambiguity exists as the result of Step 3.1) such that for all QG in this set its measure of similarity between its set of high-level objects (HQG) of QG in LQG_{M1} and a set of high-level object-usage of the user (H_U) is maximal. If there exists one best interpretation based on this measure, then go to Step 4.

Step 3.3: Determine the set of the most likely interpretation (when ambiguity exists as the result of Step 3.2) such that for all QG in this set its measure of similarity between the I_2 and a set of terms described cases is maximal and the case solution is the most similar to the query graph QG. If there exists multiple interpretations, then ask users to choose the correct interpretation.

Step 4: Map the query graph to the database query language, resolve semantic ambiguity result from the query processing, obtain the results of query and confirm them with the user.

Step 4.1: Determine the database query language (DBL) such that the DBL represents the final interpretation (QG, $<SC>_{QG}$).

Step 4.2: Determine the best semantics of DBL (if there exists semantic ambiguity from the query processing) such that the measure of similarity between its correspondent database graph and the case is maximal and the best semantics of DBL is the solution of the case. If using cases cannot resolve this problem, then ask users to choose the correct one.

Step 4.3: Obtain the results of query and confirm them with the user. If the user accepts them, the learned lessons will be appropriately updated as new keywords (the extension of graph G defined in Section 3.1.2(3)) and cases (the extension of graph G defined in Section 1.3(3)). Otherwise, the system will engage the user in dialogue in the learning mechanism to acquire new keywords and then go to Step 1.

Note that this analysis applies to the problem of natural language interfaces for enterprise databases in general. Therefore, each step allows for a wide range of possible strategies and algorithms to implement it.

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention are not limited by any of the above exemplary embodiments, but are defined only in accordance with the following claims and their equivalents.

The invention claimed is:

1. A method for processing a natural language input provided by a user, the method comprising:

providing a natural language query input by the user;

performing, based on the input, without augmentation, a search of one or more language-based databases including at least one metadata database comprising at least one of a group of information types comprising:

case information;

keywords;

information models; and

database values;

providing, through a user interface, a result of the search to the user;

identifying, for the one or more language-based databases, a finite number of database objects; and

determining a plurality of combinations of the finite number of database objects.

2. The method according to claim **1**, further comprising a step of mapping the natural language query to the plurality of combinations.

3. The method according to claim **2**, wherein the step of mapping comprises steps of:

identifying keywords in the natural language query; and

relating the keywords to the plurality of combinations.

4. The method according to claim **2**, further comprising a step of determining a reference dictionary comprising:

case information;

keywords;

information models; and

database values.

5. The method according to claim 2, wherein the step of mapping further comprises resolving ambiguity between the keywords and the plurality of combinations.

6. The method according to claim 5, wherein the step of resolving includes determining an optimal interpretation of the natural language query using at least one of a group comprising rules and heuristics.

7. The method of claim 1 wherein providing a natural language query input by the user includes providing a natural language query input by the user without constraining said query grammatically or structurally.

8. The method of claim 7 wherein the natural language query comprises a sentence, multiple sentences, a non-sentence word group or a combination thereof.

9. A computer-implemented method for processing a natural language input comprising:

receiving a natural language input;

providing from said natural language input a plurality of language-based database objects;

identifying a finite number of permutations of the plurality of database objects, the database objects being stored in a metadata database comprising at least one of a group of information comprising

case information,

keywords,

information models, and

database values; and

interpreting at least one of the permutations to provide determination of a result of the natural language input.

10. The method according to claim 9, wherein the step of providing a plurality of database objects includes providing at least one of the group comprising data types, data instances, and database computational operators.

11. The method according to claim 10, wherein the step of interpreting includes mapping the at least one of the permutations to a database query.

12. The method according to claim 11, wherein the database query is formulated in a structured query language (SQL) query.

13. The method according to claim 9, wherein one or more permutations are eliminated.

14. The method according to claim 9, further comprising:

providing a reference dictionary comprising cases, keywords, information models, graphics, and database values;

identifying, in the natural language input, a plurality of elements that belong to the reference dictionary;

determining complete paths that are implied by the plurality of elements that span elements of the natural language input and which belong to the graphics of the reference dictionary.

15. The method according to claim 14, including determining a path that includes elements of at least the informational models and database values.

16. The method according to claim 14, further comprising:

providing rules and heuristics for searching; and

determining an optimum permutation based on the rules and heuristics.

17. The method according to claim 14, further comprising adding, as a result of user input, new cases and keywords to the reference dictionary.

18. The method according to claim 9, further comprising performing a search of a group of possible interpretations to determine an optimal interpretation of the natural language input.

19. The method according to claim 9, further comprising accepting the natural language input, wherein the action of accepting the natural language input accepts a free-format input.

20. The method according to claim 9, wherein of interpretation of natural language input is based on other than the results of analyzing syntax of the natural language query input.

21. The method according to claim 9, further comprising representing graphically elements of the metadata database.

* * * * *

EXHIBIT B

Learn more about Siri. Beta

What is Siri?

Siri is the intelligent personal assistant that helps you get things done just by asking. It allows you to use your voice to send messages, schedule meetings, place phone calls, and more. But Siri isn't like traditional voice recognition software that requires you to remember keywords and speak specific commands. Siri understands your natural speech, and it asks you questions if it needs more information to complete a task.

Siri is available for iPhone 5, iPhone 4S, iPad (3rd generation), and iPod touch (5th generation).

Using Siri

How do I ask Siri something?

To talk to Siri, hold down the Home button and you'll hear two quick beeps and see "What can I help you with?" on the screen. Just begin speaking. The microphone icon lights up to let you know that Siri hears you talking. Once you've started a dialogue with Siri, tap the microphone icon to talk to it again.

There's more than one way to talk to Siri. Siri works with headphones and Bluetooth headsets. When you're using headphones with a remote and microphone, you can press and hold the center button to talk to Siri. With a Bluetooth headset, press and hold the call button to bring up Siri.

On iPhone 4S and iPhone 5, simply bring iPhone up to your ear when the screen is on. You'll hear two quick beeps to indicate that Siri is listening to you.

In a car that supports Eyes Free, you can also start a conversation with Siri just by pressing a button on your steering wheel.

Siri waits for you to stop talking, but you can also tap the microphone icon to tell Siri you're done talking. This is useful when there's a lot of background noise.

What happens after I ask Siri a question or ask it to do something?

When you finish speaking, Siri displays the text of what you said and provides a response. If Siri needs more information to complete a request, it will ask you a question. For example, if you say "Remind me to call my mom," Siri will ask "What time would you like me to remind you?"

When you use earphones or a headset, Siri reads back text messages and email messages that

you've dictated before you send them, and it reads back the subjects of reminders before you create them. This is especially helpful when you're driving and can't see the screen.

Do I have to say things a certain way to get Siri to respond?

No. You can speak to Siri as you would to a person — in a natural voice with a conversational tone. If you want to know what the weather will be like tomorrow, simply say "What will the weather be like tomorrow?" Or "Does it look like rain tomorrow?" Or even "Will I need an umbrella tomorrow?" No matter how you ask, Siri will tell you the forecast.

Does Siri work out of the box, or do I have to teach it?

Siri works right out of the box, without any work on your part. And the more you use Siri, the better it will understand you. It does this by learning about your accent and other characteristics of your voice. Siri uses voice recognition algorithms to categorize your voice into one of the dialects or accents it understands. As more people use Siri and it's exposed to more variations of a language, its overall recognition of dialects and accents will continue to improve, and Siri will work even better.

Siri also uses information from your contacts, music library, calendars, and reminders to better understand what you say. So it responds more accurately when you ask to make a phone call, play music, or create an appointment or reminder.

If you like, you can reset what Siri has learned about your voice by turning Siri off and then back on in Settings > General > Siri.

What Siri Can Do For You

What types of things can I ask Siri about or ask it to do?

You can ask Siri to make a call, find a business and get directions, schedule reminders and meetings, search the web, and more. You can even ask Siri "What can you do for me?" or tap the "i" in the right corner of the screen when you bring Siri up. You'll see examples of things Siri can do, along with ways you can ask for things.

Which apps does Siri work with?

Siri works with almost all your built-in apps. And it's smart enough to figure out which apps to use to provide you with answers. It also uses Search and Location Services to help you with your requests. Here's a list of apps and services that Siri works with worldwide:

- Maps
- Sports
- Movies
- Local Search
- Post on Facebook
- FaceTime
- Phone
- Mail
- Web Search
- Send a Tweet
- App Launch
- Find My Friends
- Music
- Messages
- Calendar
- Reminders
- Notes
- Contacts
- Weather
- Stocks
- Wikipedia search
- Alarms, World Clock and Timer
- Wolfram|Alpha (English only)

How does Siri learn who I am?

If Siri knows who you are, it can use your information to help you. To make sure Siri knows who you are, select your contact information in Settings > General > Siri > My Info.

Your information is used for questions like "How do I get home?" or "What good restaurants are near work?"

How does Siri learn about my key relationships?

Siri helps you by learning about the key people in your life. The first time you ask Siri to call your

sister, it will ask you who your sister is. That information is stored in Contacts along with other relationship information like “mom,” “husband,” and “grandma.”

How do location-based reminders work?

Because Siri knows your current location and other locations like “home” and “work,” it can remind you to do a certain task when you leave a location or arrive at a location. So if you tell Siri,“Remind me to call my wife when I leave the office,” Siri does just that.

To turn off the ability for Siri to use your location, go to Settings > Location Services and set the switch for Siri to Off. Regardless of how Locations Services is set for Siri, information about your location is not tracked or stored outside your device.

Does iOS take dictation?

Yes. iPhone 5, iPhone 4S, iPad (3rd generation), and iPod touch (5th generation) support dictation in any app that has a keyboard. So instead of typing, you can speak and your words will be entered as text.

To start dictation, tap the microphone button on your keyboard and start talking. When you’re finished, tap Done and your words will be turned into text. Dictation for each language is built into the keyboard for that language.

Is Siri accessible to blind and visually impaired users?

Yes. VoiceOver, the screen reader built into iOS, can speak any text that’s displayed in responses from Siri. You can navigate through the responses and have each one read to you. This includes the days of a weather forecast, the body of an email, the details of an answer from Wolfram|Alpha, and more.

Language Support and Availability

Siri is available for iPhone 5, iPhone 4S, iPad (3rd generation), and iPod touch (5th generation). Siri understands and can speak the following languages:

- United States (English, Spanish)
- United Kingdom (English)
- Australia (English)
- France (French)
- Germany (German)
- Japan (Japanese)
- Canada (English, Canadian French)
- China (Mandarin)
- Hong Kong (Cantonese)
- Italy (Italian)
- Korea (Korean)
- Mexico (Spanish)
- Spain (Spanish)
- Switzerland (Italian, French, German)
- Taiwan (Mandarin)

Can I use Siri in any of these languages in other countries?

Yes. Siri can be enabled in any country, and you can choose to speak to it in any of the languages that Siri supports. However, Siri is designed to recognize the specific accents and dialects of the supported countries listed above. Since every language has its own accents and dialects, the accuracy rate will be higher for native speakers.

Siri is available in Beta only on iPhone 4S, iPhone 5, iPad (3rd generation), and iPod touch (5th generation) and requires Internet access. Siri may not be available in all languages or in all areas, and features may vary by area. Cellular data charges may apply.

EXHIBIT C

US 20120016678A1

(19) **United States**

(12) **Patent Application Publication**

Gruber et al.

(10) **Pub. No.: US 2012/0016678 A1**

(43) **Pub. Date: Jan. 19, 2012**

(54) **INTELLIGENT AUTOMATED ASSISTANT**

(75) Inventors: **Thomas Robert Gruber**, Emerald Hills, CA (US); **Adam John Cheyer**, Oakland, CA (US); **Dag Kittlaus**, San Jose, CA (US); **Didier Rene Guzzoni**, Mont-sur-Rolle (CH); **Christopher Dean Brigham**, San Jose, CA (US); **Richard Donald Giuli**, Arroyo Grande, CA (US); **Marcello Bastea-Forte**, New York, NY (US); **Harry Joseph Saddler**, Berkeley, CA (US)

(73) Assignee: **APPLE INC.**, Cupertino, CA (US)

(21) Appl. No.: **12/987,982**

(22) Filed: **Jan. 10, 2011**

Related U.S. Application Data

(60) Provisional application No. 61/295,774, filed on Jan. 18, 2010.

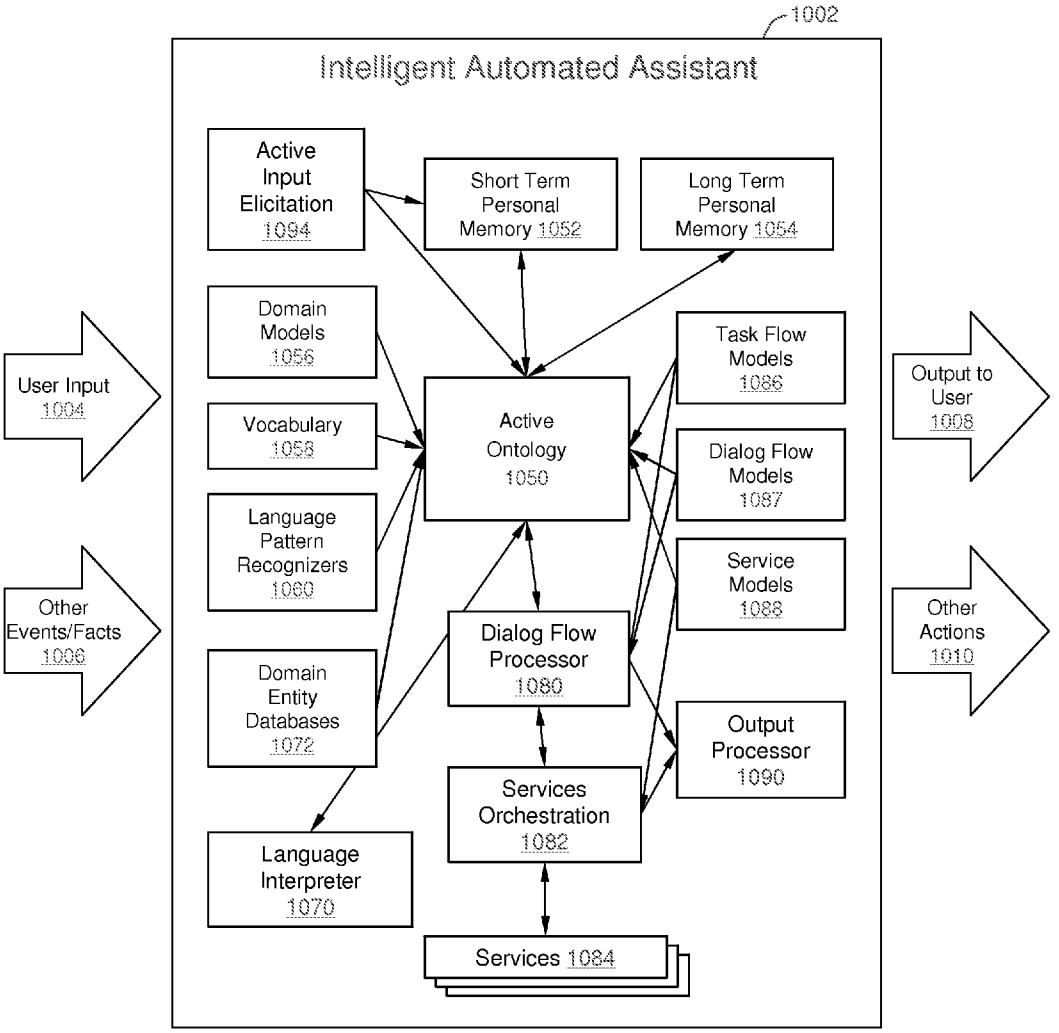
Publication Classification

(51) **Int. Cl.**
G10L 21/00 (2006.01)

(52) **U.S. Cl.** **704/275**; 704/E21.001

(57) **ABSTRACT**

An intelligent automated assistant system engages with the user in an integrated, conversational manner using natural language dialog, and invokes external services when appropriate to obtain information or perform various actions. The system can be implemented using any of a number of different platforms, such as the web, email, smartphone, and the like, or any combination thereof. In one embodiment, the system is based on sets of interrelated domains and tasks, and employs additional functionally powered by external services with which the system can interact.



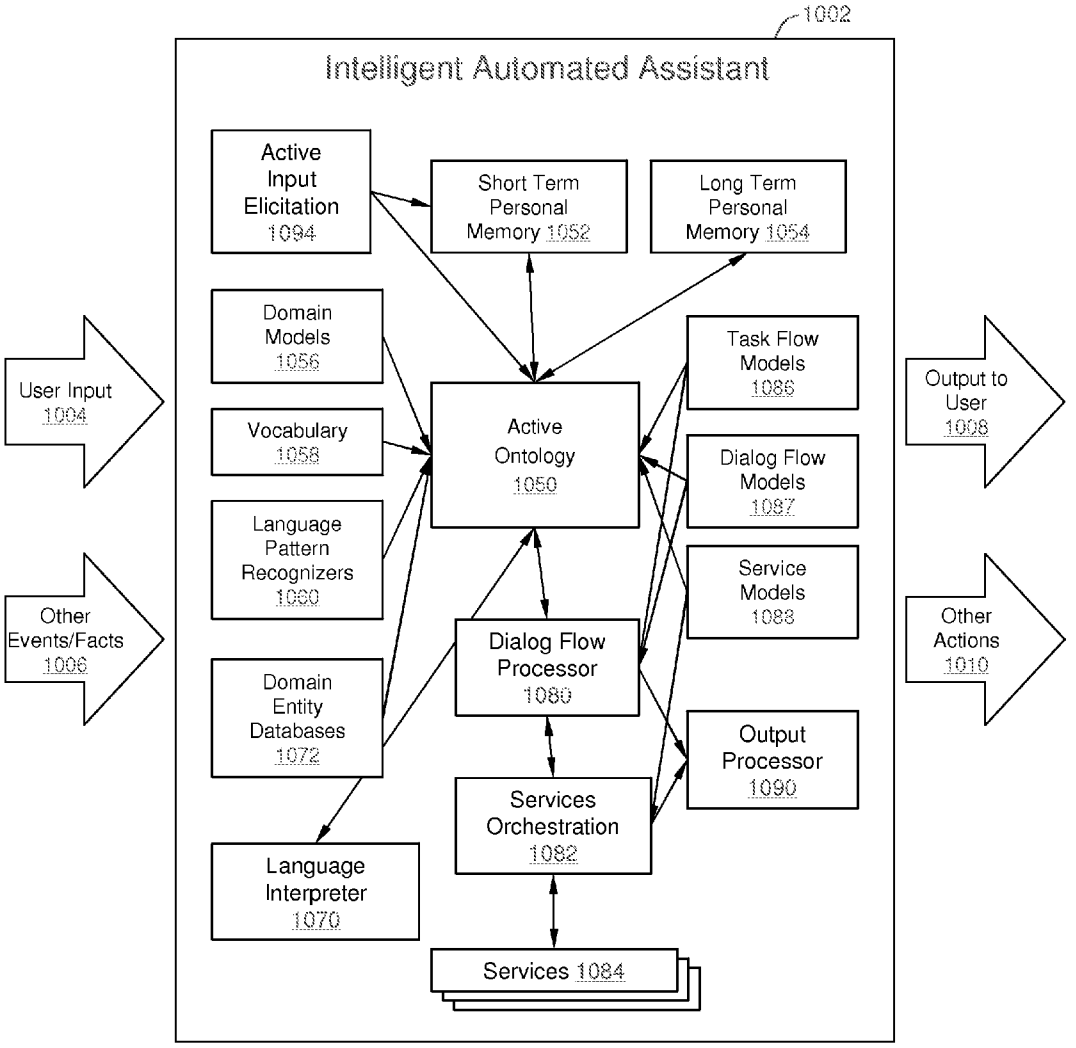

1002
Intelligent Automated Assistant
Active Input Elicitation 1094
Short Term Personal Memory 1052
Long Term Personal Memory 1054
Domain Models 1056
Vocabulary 1058
Language Pattern Recognizers 1060
Domain Entity Databases 1072
Language Interpreter 1070
Active Ontology 1050
Dialog Flow Processor 1080
Services Orchestration 1082
Services 1084
Task Flow Models 1086
Dialog Flow Models 1087
Service Models 1088
Output Processor 1090
User Input 1004
Other Events/Facts 1006
Output to User 1008
Other Actions 1010

FIG. 1

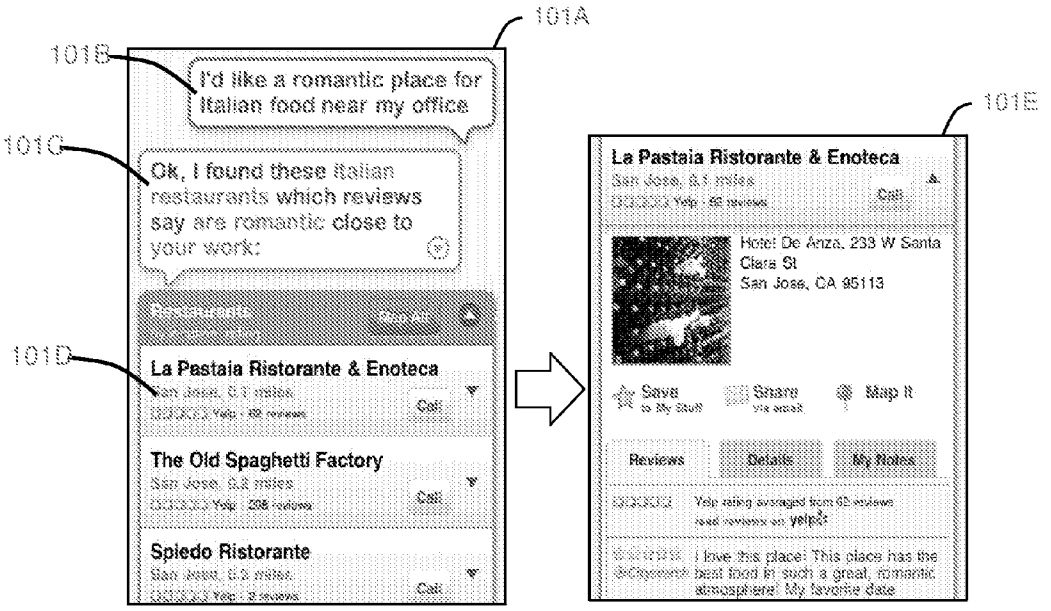

101A
101B
I'd like a romantic place for Italian food near my office
101C
Ok, I found these Italian restaurants which reviews say are romantic close to your work:
101D
La Pastaia Ristorante & Enoteca
Call
The Old Spaghetti Factory
Call
Spiedo Ristorante
Call
101E
La Pastaia Ristorante & Enoteca
Call
Hotel De Anza, 233 W Santa Clara St
San Jose, CA 95113
Save
Share
Map It
Reviews
Details
My Notes

FIG. 2

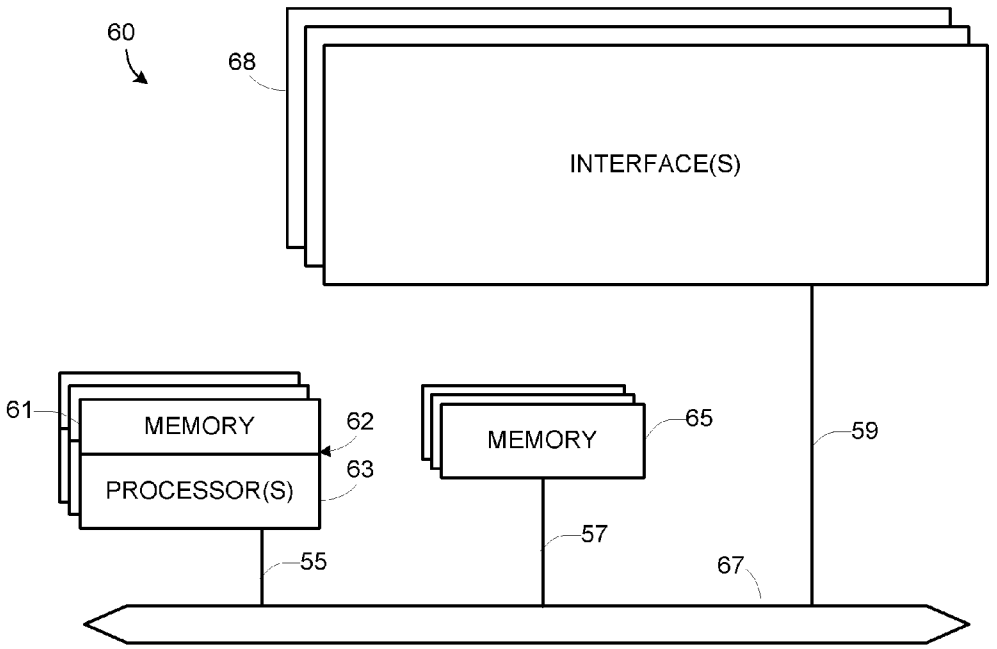

60
68
INTERFACE(S)
61
MEMORY
62
PROCESSOR(S)
63
MEMORY
65
59
57
55
67

FIG. 3

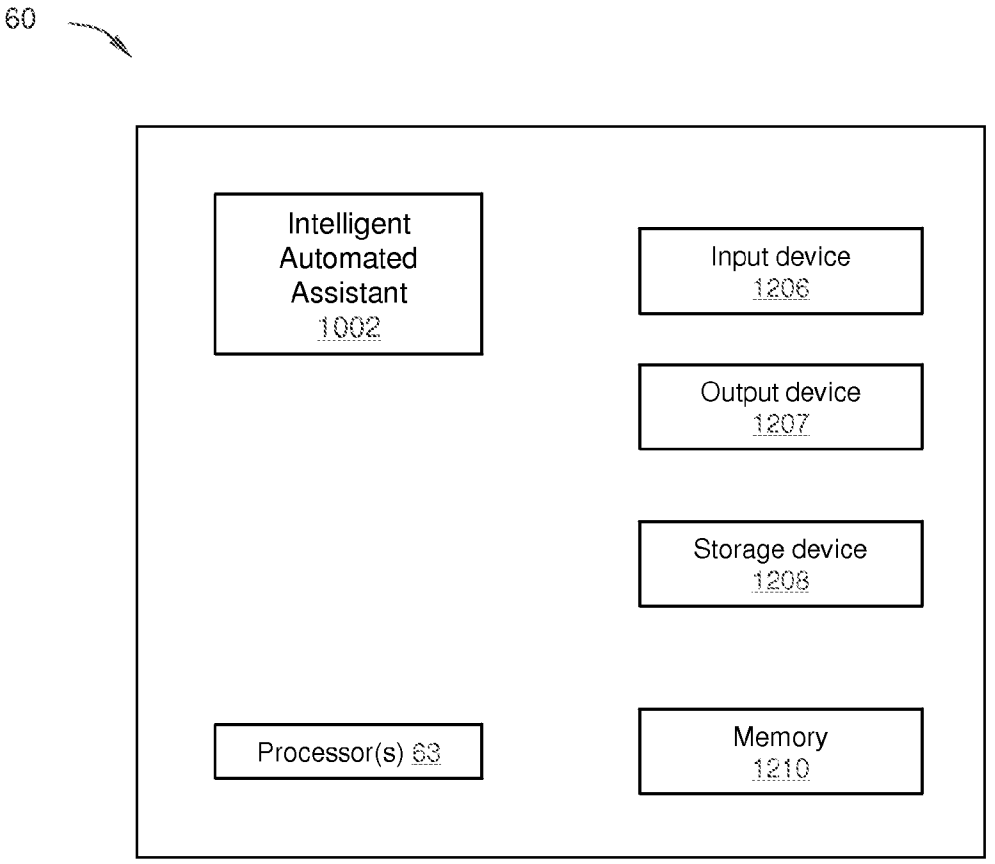

60
Intelligent Automated Assistant 1002
Input device 1206
Output device 1207
Storage device 1208
Processor(s) 63
Memory 1210

FIG. 4

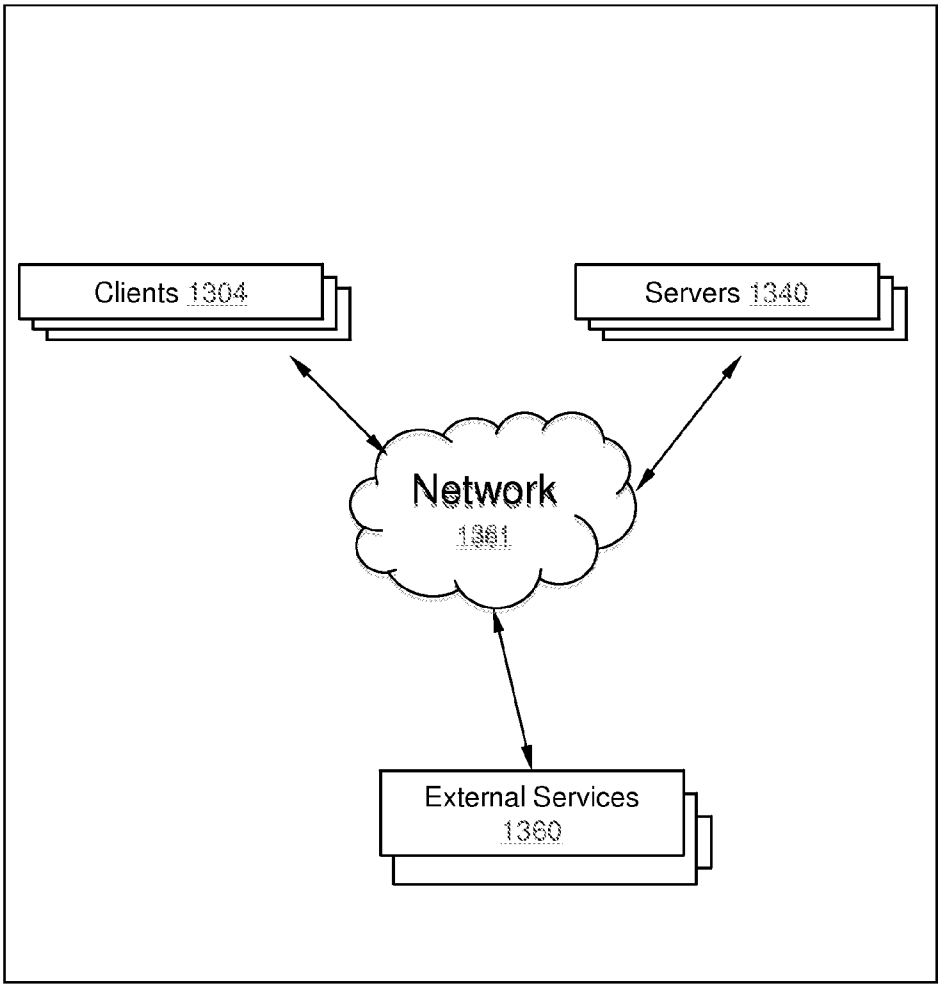

Clients 1304
Servers 1340
Network
1361
External Services
1360

FIG. 5

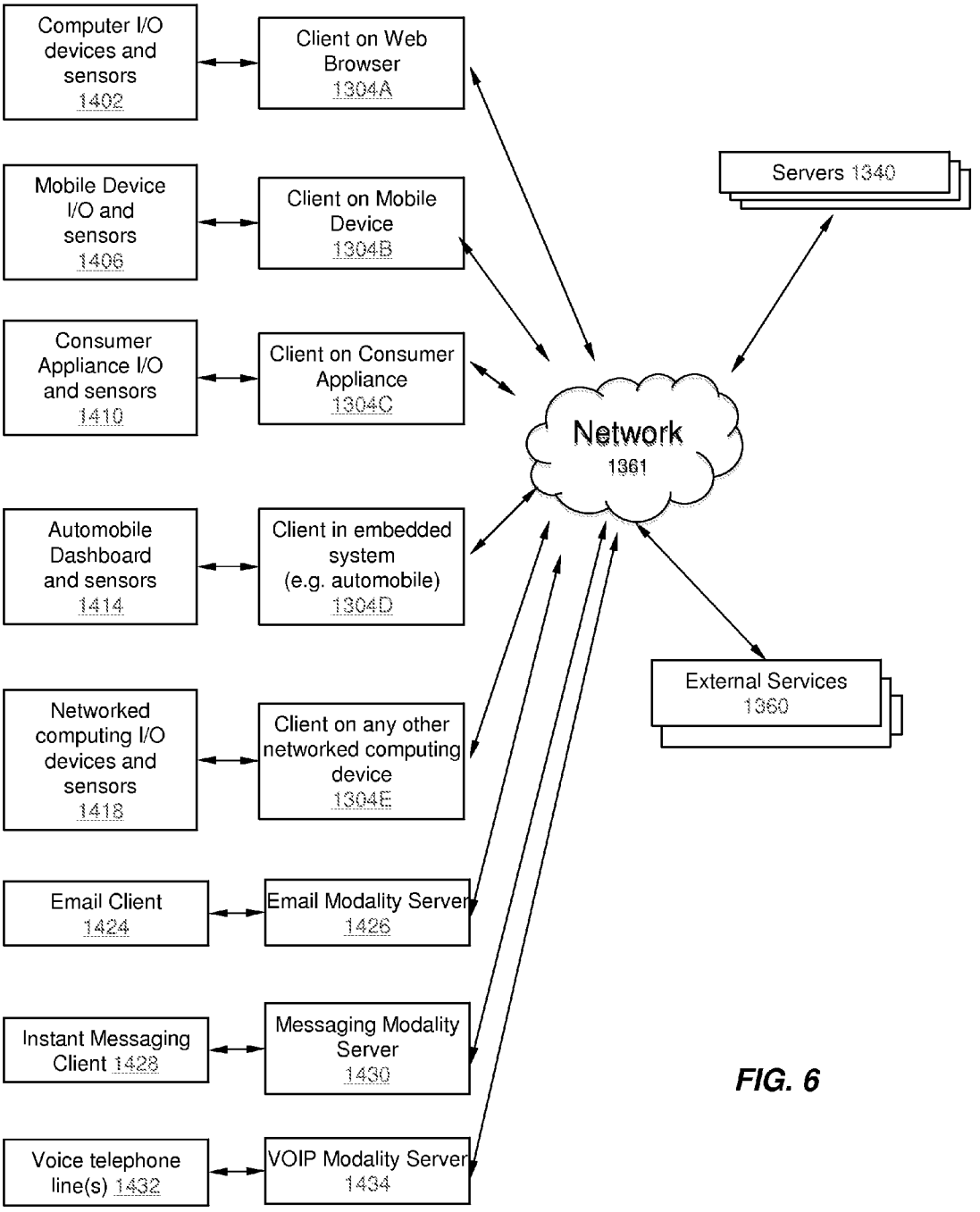

Computer I/O devices and sensors 1402
Client on Web Browser 1304A
Mobile Device I/O and sensors 1406
Client on Mobile Device 1304B
Consumer Appliance I/O and sensors 1410
Client on Consumer Appliance 1304C
Automobile Dashboard and sensors 1414
Client in embedded system (e.g. automobile) 1304D
Networked computing I/O devices and sensors 1418
Client on any other networked computing device 1304E
Email Client 1424
Email Modality Server 1426
Instant Messaging Client 1428
Messaging Modality Server 1430
Voice telephone line(s) 1432
VOIP Modality Server 1434
Network 1361
Servers 1340
External Services 1360

FIG. 6

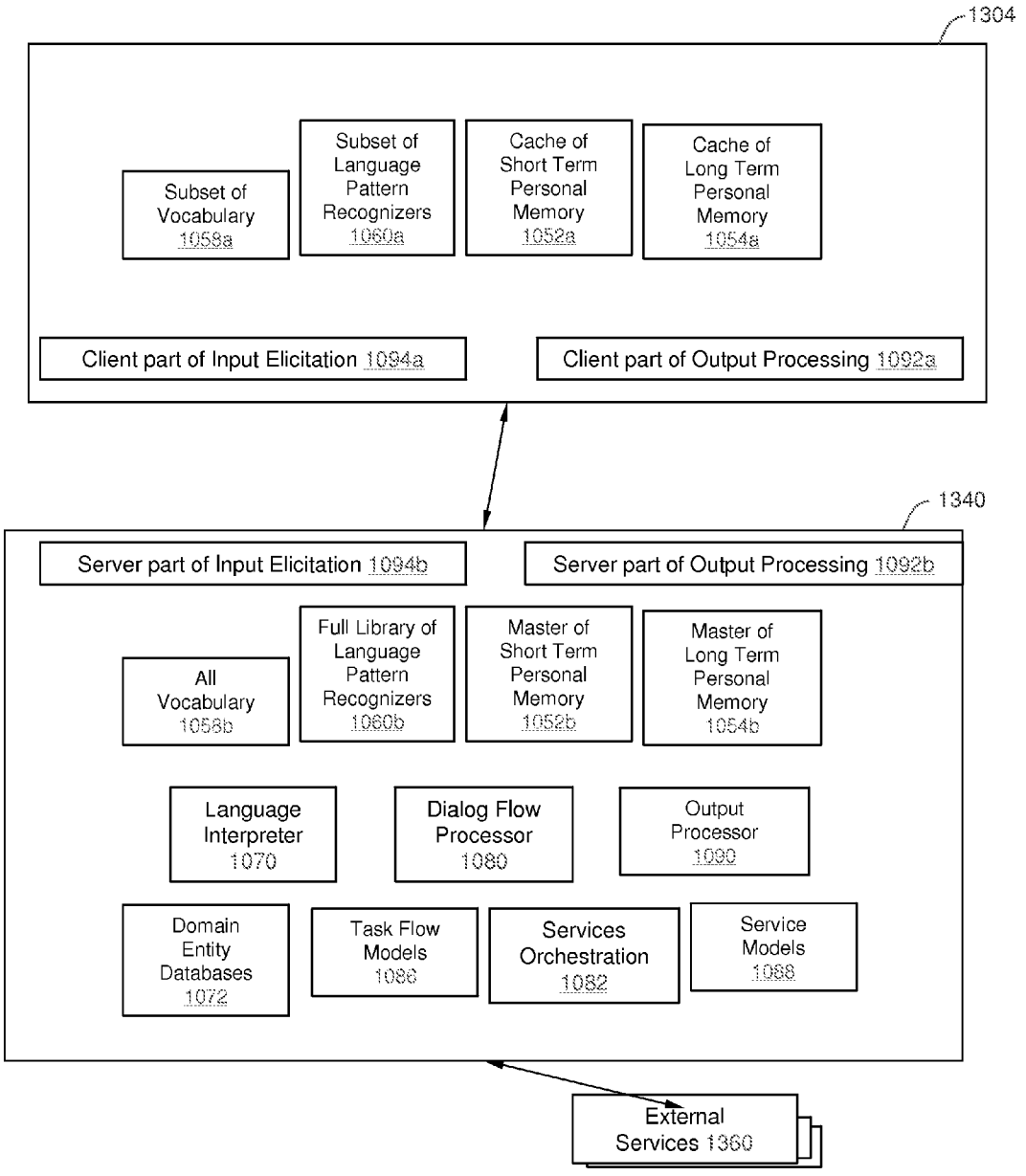

1304
Subset of Vocabulary 1058a
Subset of Language Pattern Recognizers 1060a
Cache of Short Term Personal Memory 1052a
Cache of Long Term Personal Memory 1054a
Client part of Input Elicitation 1094a
Client part of Output Processing 1092a
1340
Server part of Input Elicitation 1094b
Server part of Output Processing 1092b
All Vocabulary 1058b
Full Library of Language Pattern Recognizers 1060b
Master of Short Term Personal Memory 1052b
Master of Long Term Personal Memory 1054b
Language Interpreter 1070
Dialog Flow Processor 1080
Output Processor 1090
Domain Entity Databases 1072
Task Flow Models 1086
Services Orchestration 1082
Service Models 1088
External Services 1360

FIG. 7

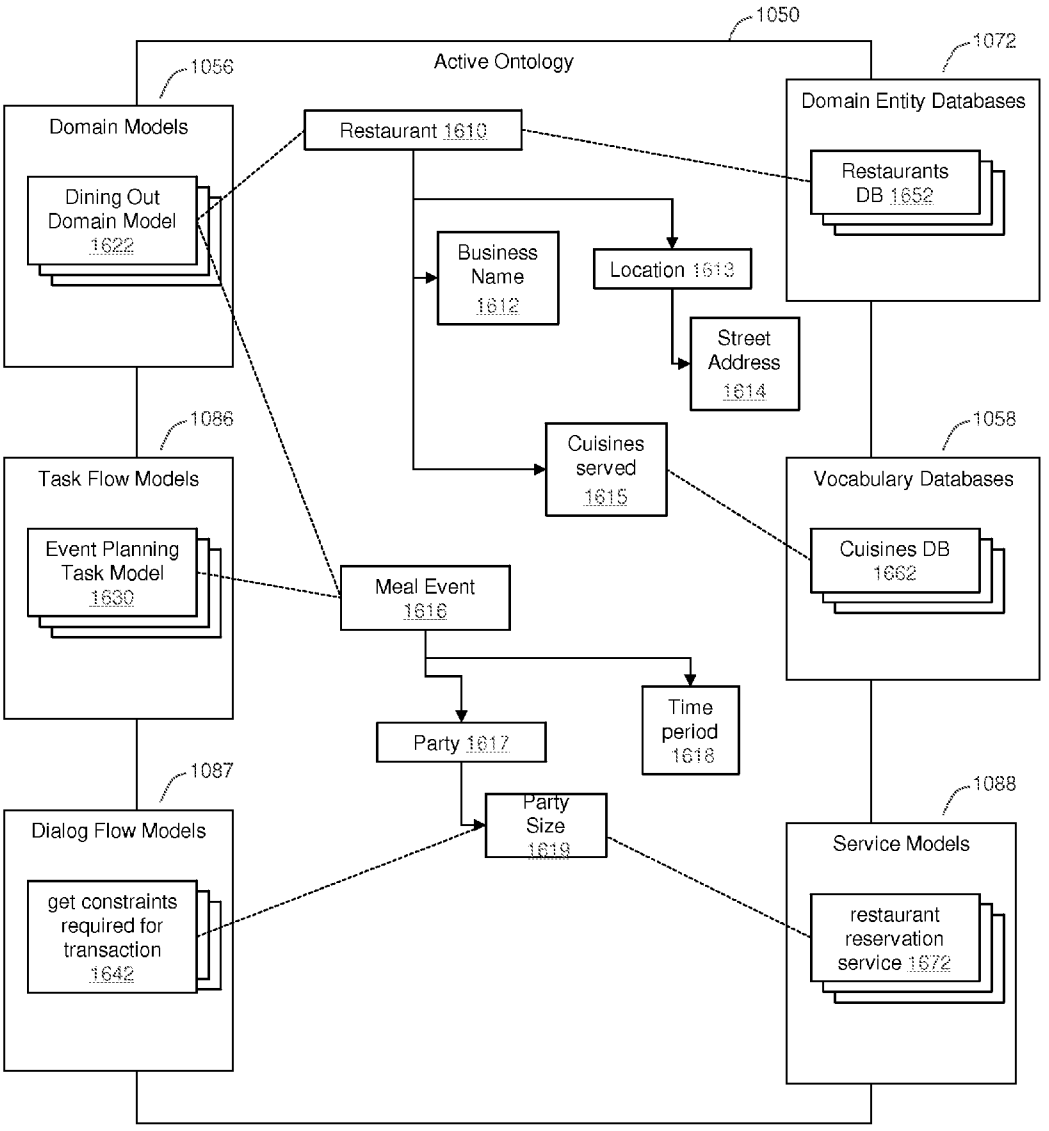

1050
Active Ontology
1056
Domain Models
Dining Out Domain Model 1622
Restaurant 1610
Business Name 1612
Location 1613
Street Address 1614
1072
Domain Entity Databases
Restaurants DB 1652
1086
Task Flow Models
Event Planning Task Model 1630
Meal Event 1616
Cuisines served 1615
Party 1617
Time period 1618
Party Size 1619
1058
Vocabulary Databases
Cuisines DB 1662
1087
Dialog Flow Models
get constraints required for transaction 1642
1088
Service Models
restaurant reservation service 1672

FIG. 8

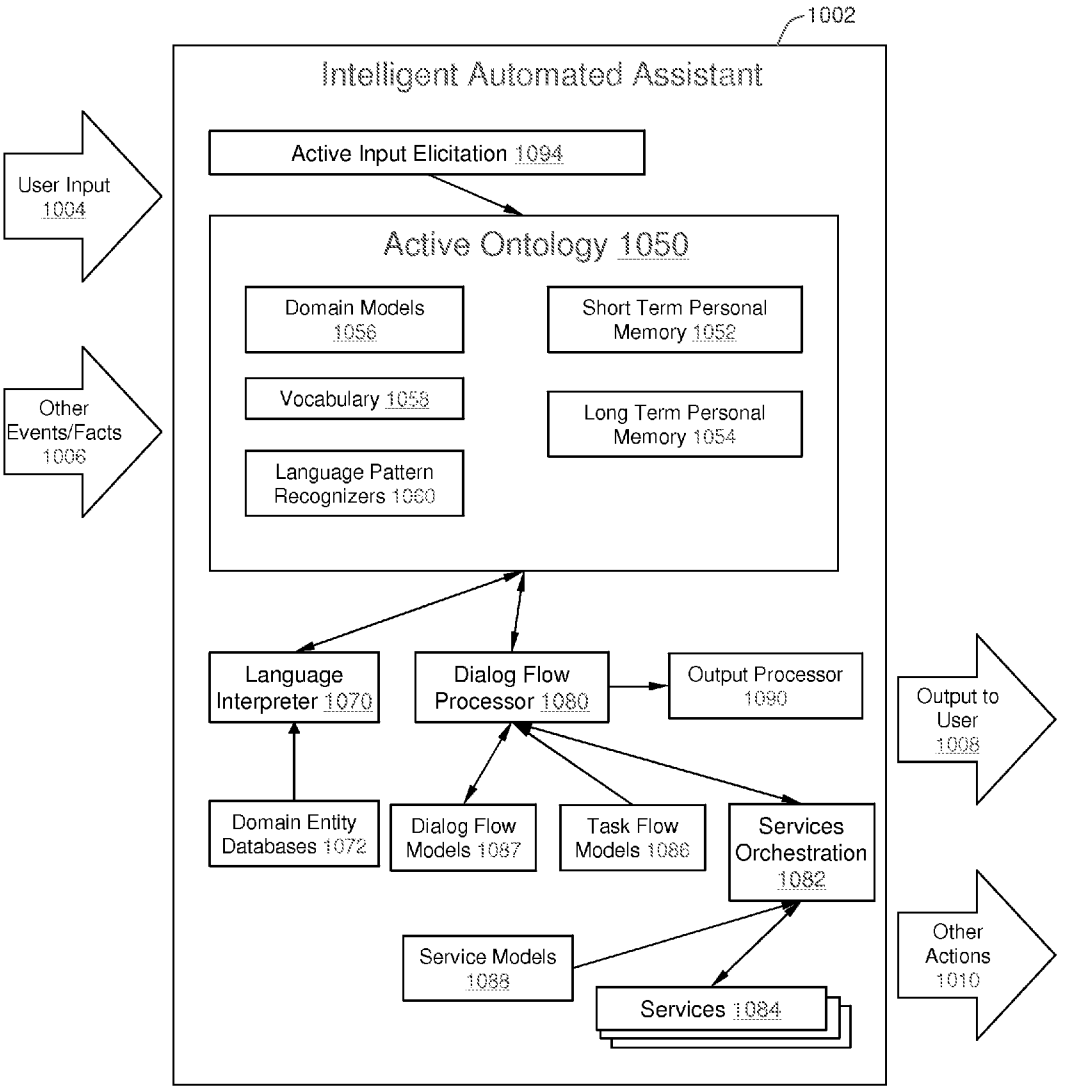

1002
Intelligent Automated Assistant
User Input 1004
Other Events/Facts 1006
Active Input Elicitation 1094
Active Ontology 1050
Domain Models 1056
Short Term Personal Memory 1052
Vocabulary 1058
Long Term Personal Memory 1054
Language Pattern Recognizers 1060
Language Interpreter 1070
Dialog Flow Processor 1080
Output Processor 1090
Output to User 1008
Domain Entity Databases 1072
Dialog Flow Models 1087
Task Flow Models 1086
Services Orchestration 1082
Service Models 1088
Services 1084
Other Actions 1010

FIG. 9

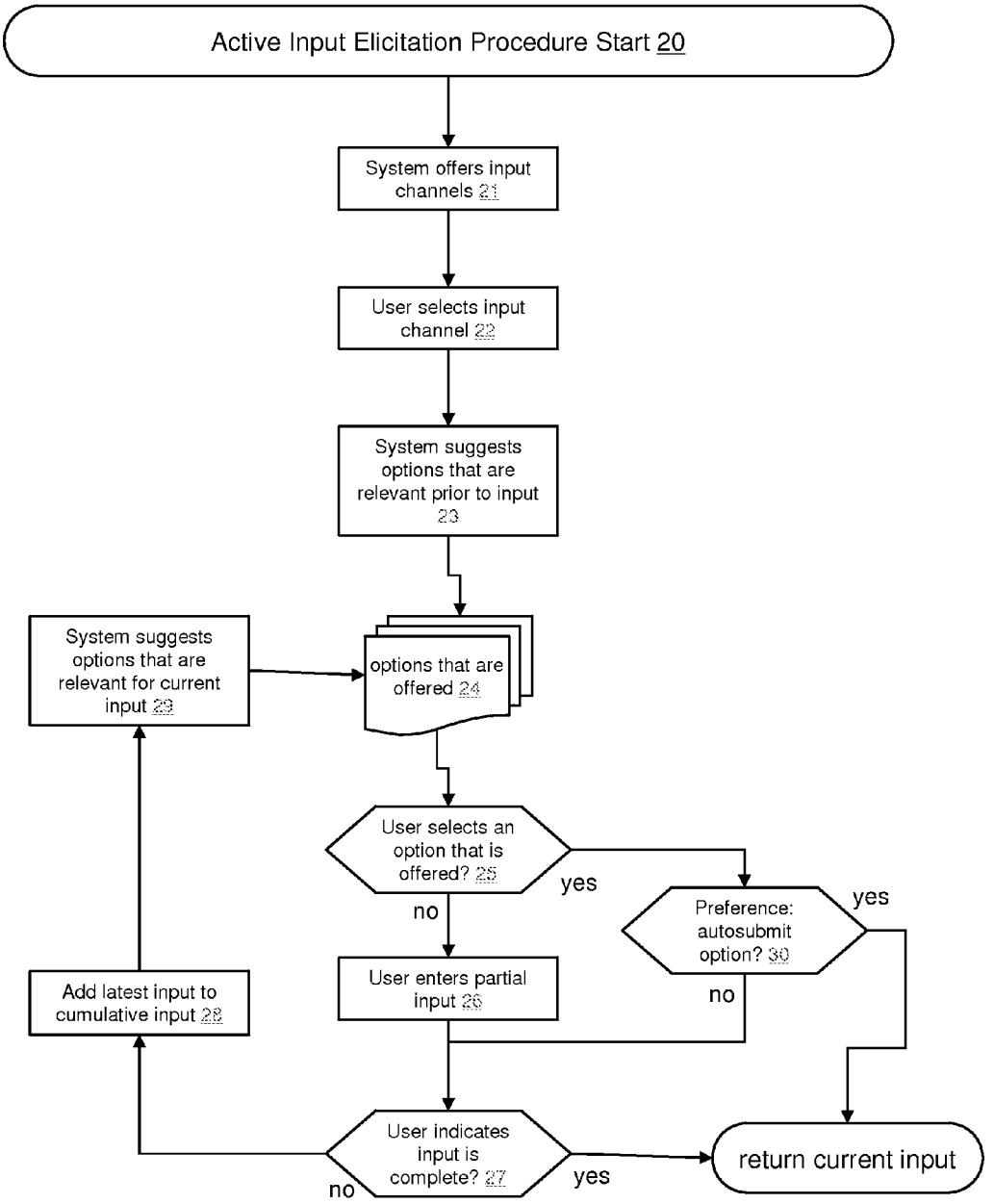

Active Input Elicitation Procedure Start 20
System offers input channels 21
User selects input channel 22
System suggests options that are relevant prior to input 23
System suggests options that are relevant for current input 29
options that are offered 24
User selects an option that is offered? 25
yes
no
Preference: autosubmit option? 30
yes
no
User enters partial input 26
Add latest input to cumulative input 28
User indicates input is complete? 27
no
yes
return current input

FIG. 10

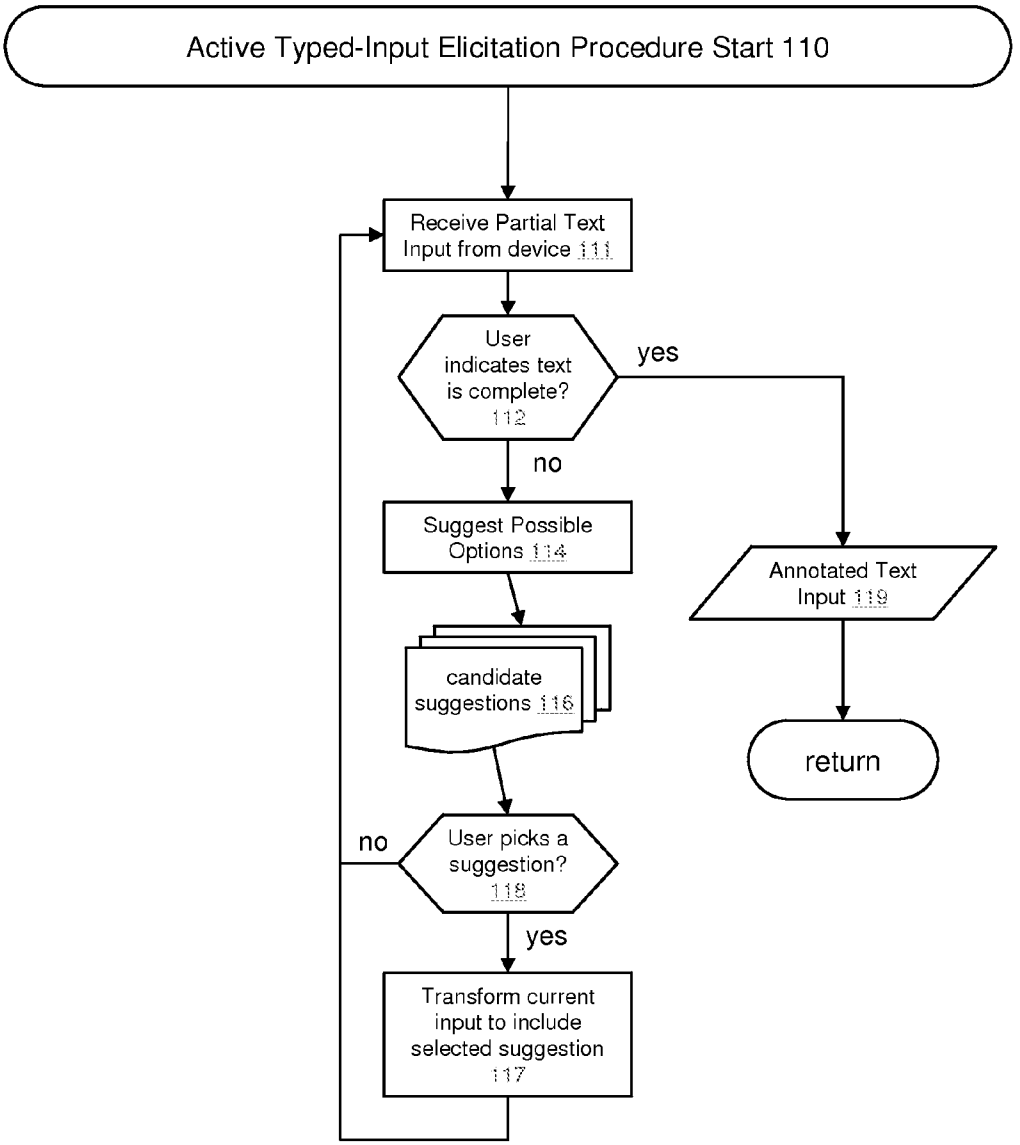

Active Typed-Input Elicitation Procedure Start 110
Receive Partial Text Input from device 111
User indicates text is complete? 112
yes
no
Suggest Possible Options 114
candidate suggestions 116
User picks a suggestion? 118
no
yes
Transform current input to include selected suggestion 117
Annotated Text Input 119
return

FIG. 11

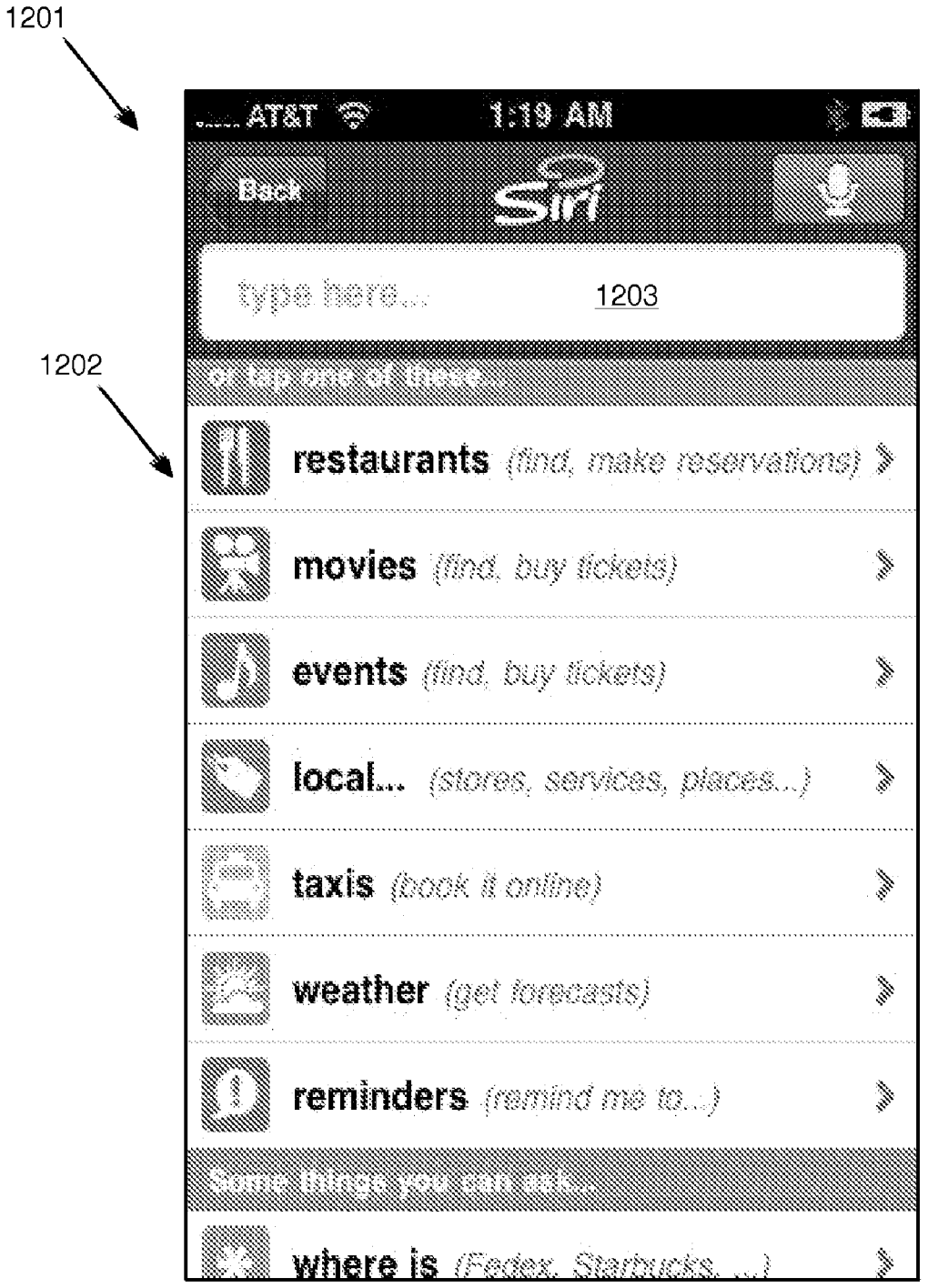

1201
1202
AT&T
1:19 AM
Back
Siri
type here...
1203
restaurants (find, make reservations)
movies (find, buy tickets)
events (find, buy tickets)
local... (stores, services, places...)
taxis (book it online)
weather (get forecasts)
reminders (remind me to...)

FIG. 12

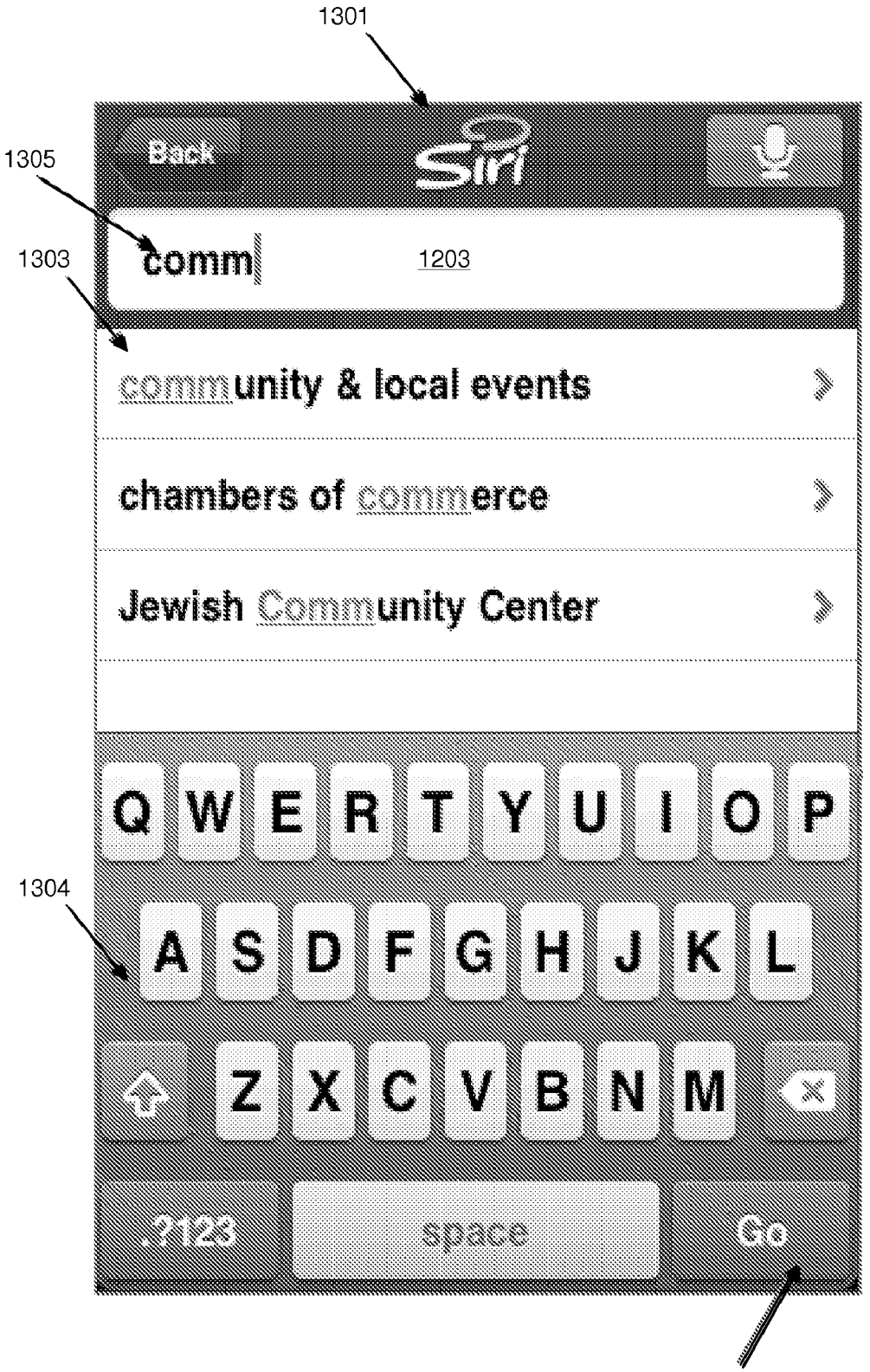

1301
Back
Siri
1305
comm
1203
1303
community & local events
chambers of commerce
Jewish Community Center
Q W E R T Y U I O P
1304
A S D F G H J K L
Z X C V B N M
.?123
space
Go
1306

FIG. 13

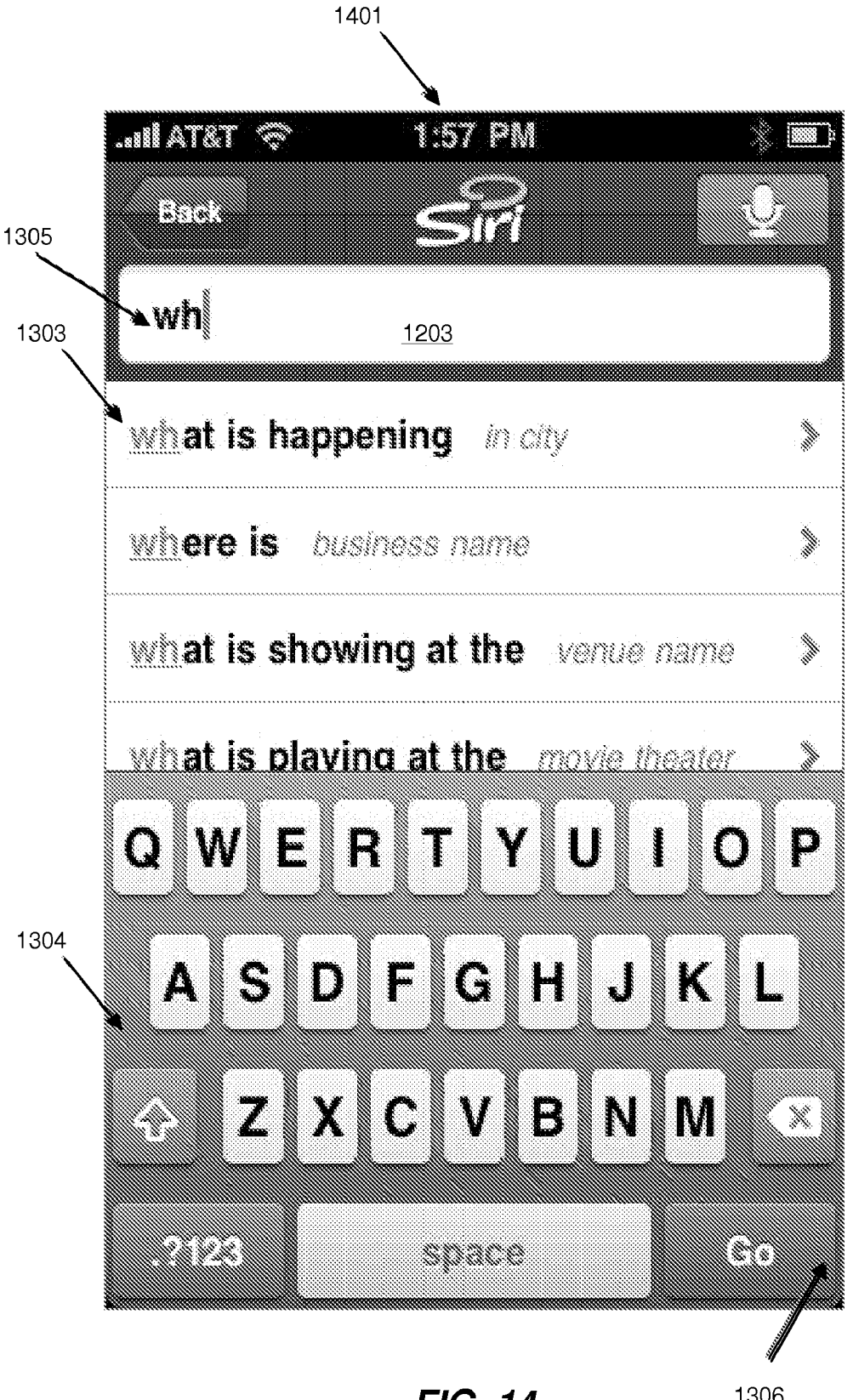

1401
AT&T
1:57 PM
Back
Siri
1305
wh
1203
1303
what is happening in city
where is business name
what is showing at the venue name
what is playing at the movie theater
Q W E R T Y U I O P
1304
A S D F G H J K L
Z X C V B N M
.?123
space
Go
1306

FIG. 14

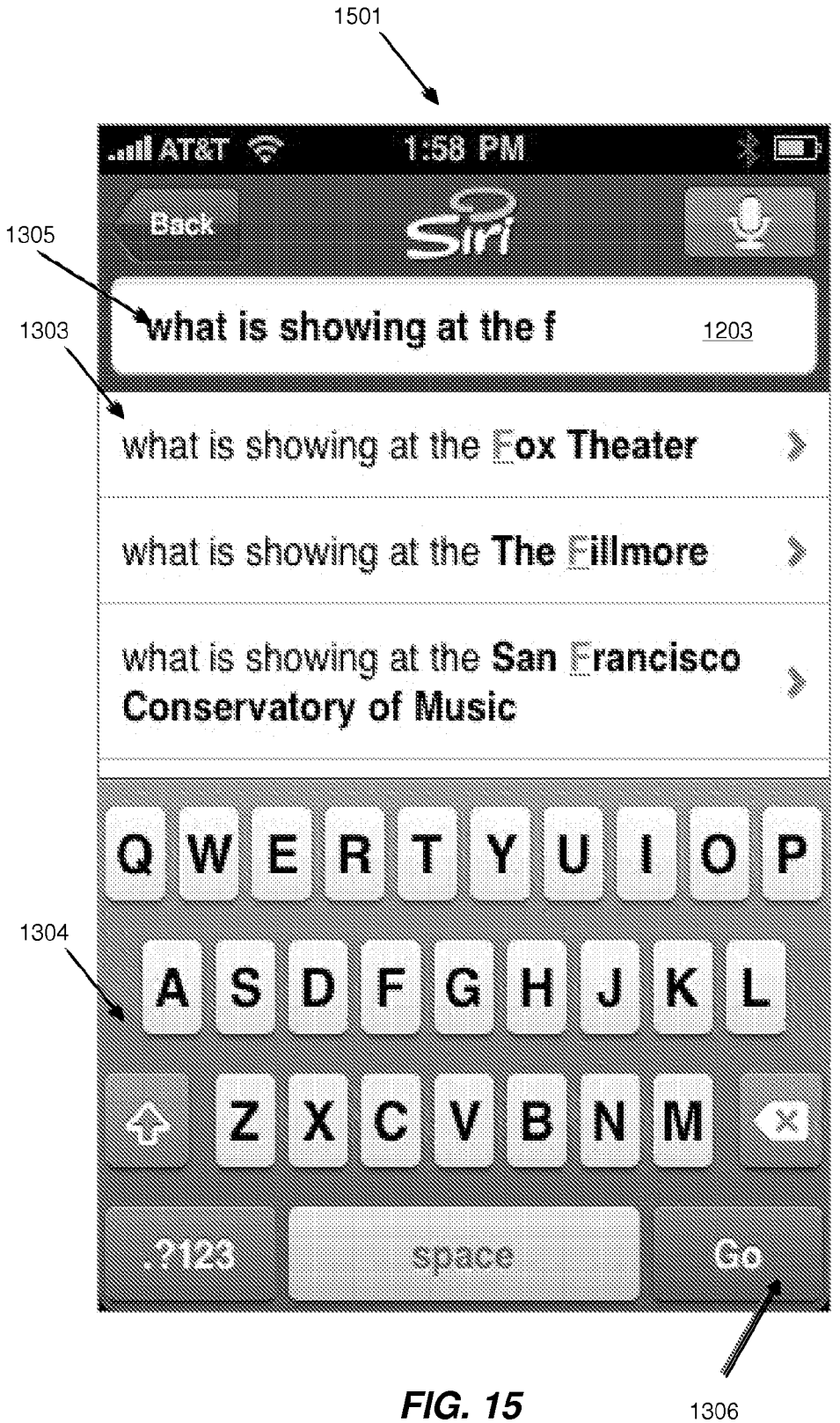

1501
AT&T
1:58 PM
Back
Siri
1305
what is showing at the f
1203
1303
what is showing at the Fox Theater
what is showing at the The Fillmore
what is showing at the San Francisco Conservatory of Music
Q W E R T Y U I O P
1304
A S D F G H J K L
Z X C V B N M
.?123
space
Go
1306

FIG. 15

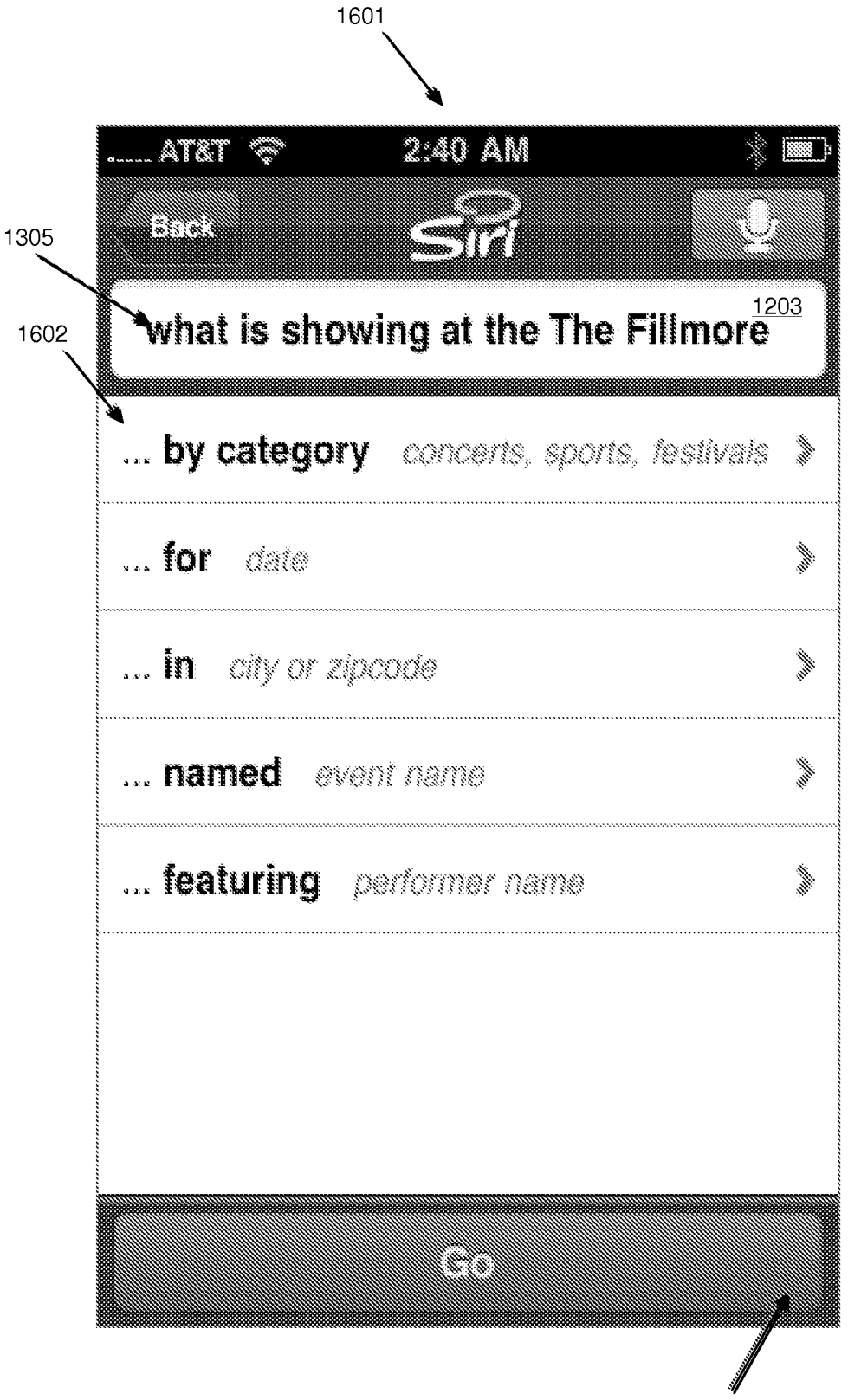

1601
AT&T
2:40 AM
Back
Siri
1305
1602
what is showing at the The Fillmore
1203
... by category concerts, sports, festivals
... for date
... in city or zipcode
... named event name
... featuring performer name
Go
1306

FIG. 16

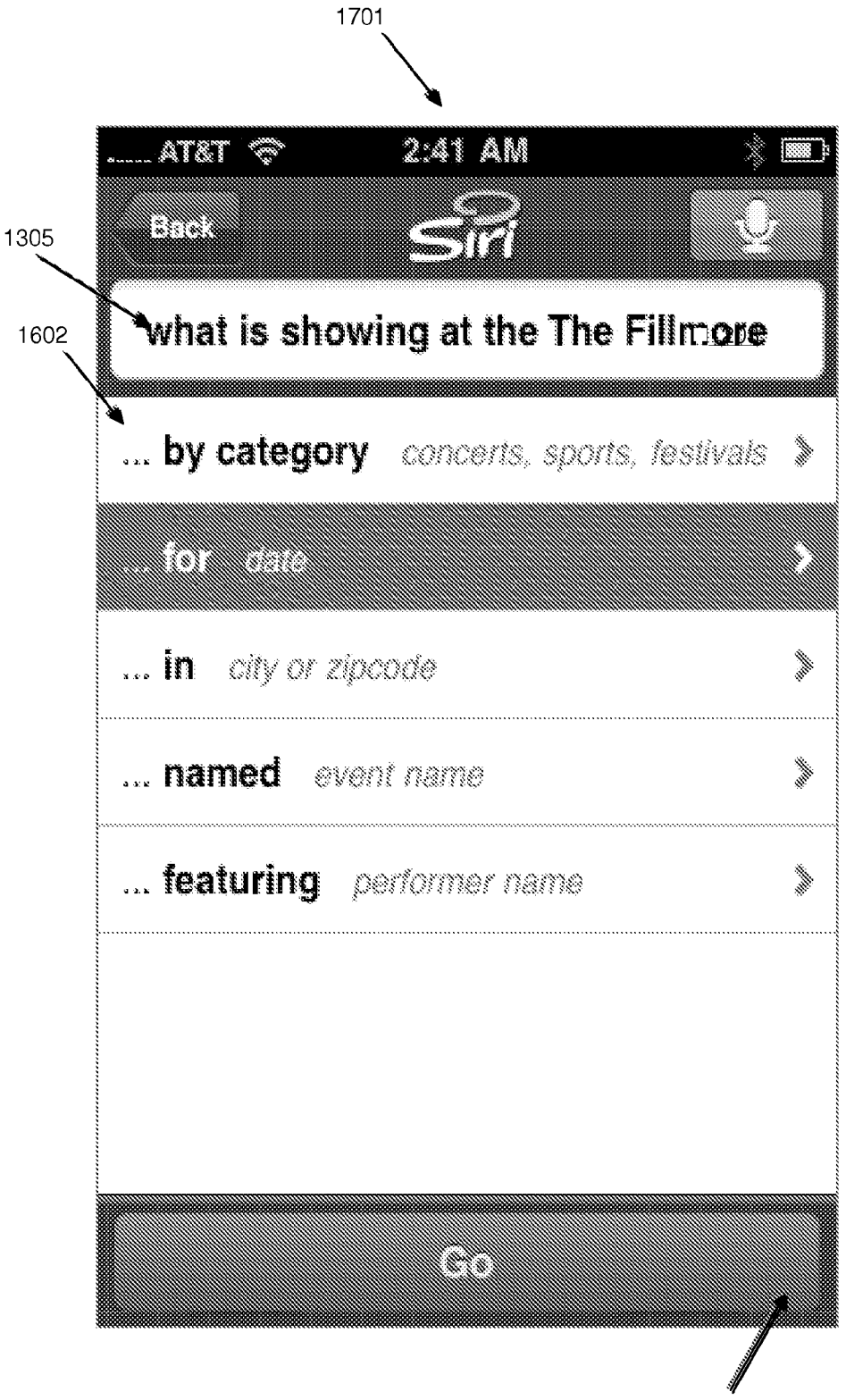

1701
AT&T
2:41 AM
Back
Siri
1305
what is showing at the The Fillmore
1602
... by category concerts, sports, festivals
... for date
... in city or zipcode
... named event name
... featuring performer name
Go
1306

FIG. 17

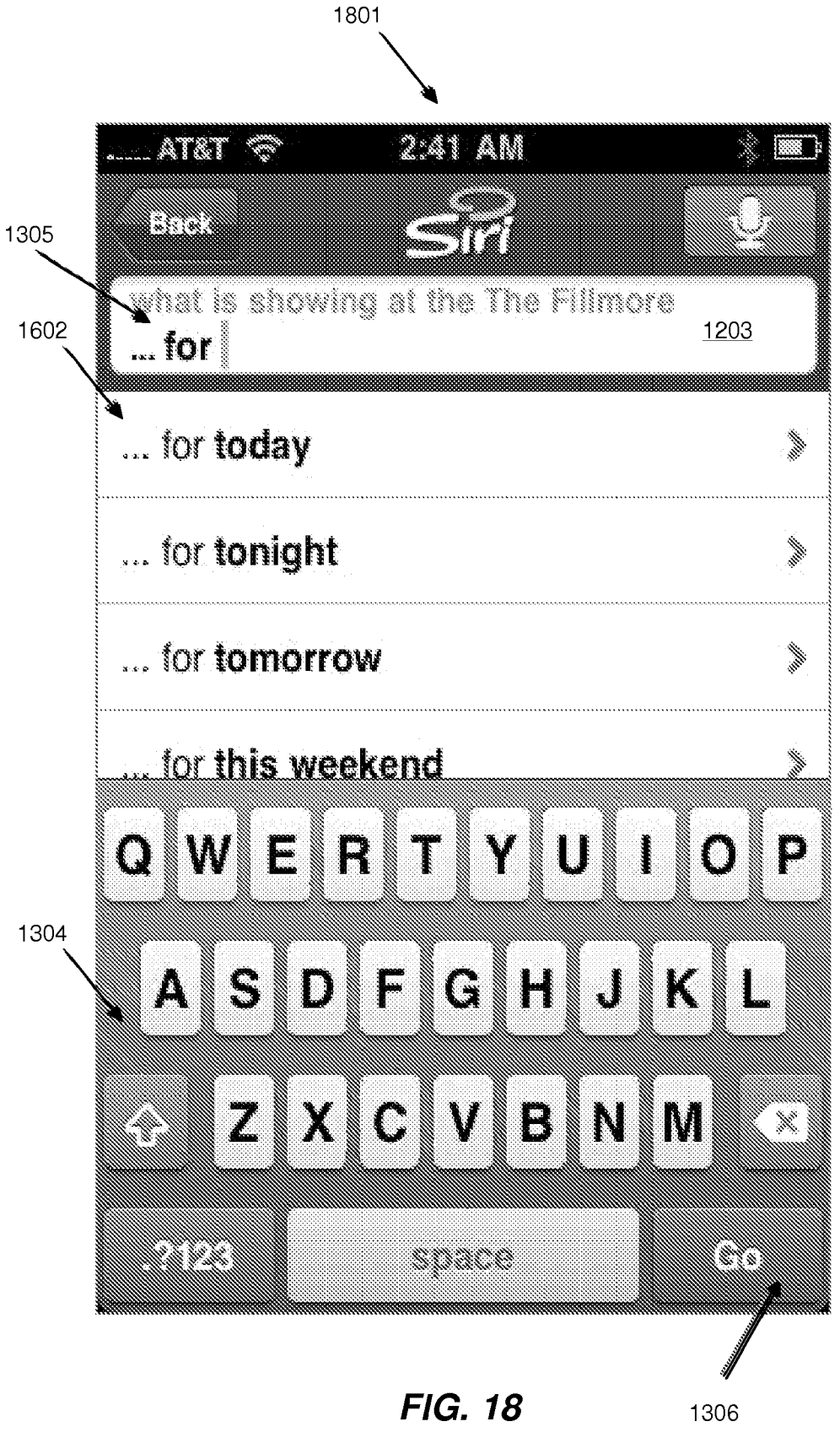

1801
AT&T
2:41 AM
Back
Siri
1305
what is showing at the The Fillmore
1602
... for
1203
... for today
... for tonight
... for tomorrow
... for this weekend
Q W E R T Y U I O P
1304
A S D F G H J K L
Z X C V B N M
.?123
space
Go
1306

FIG. 18

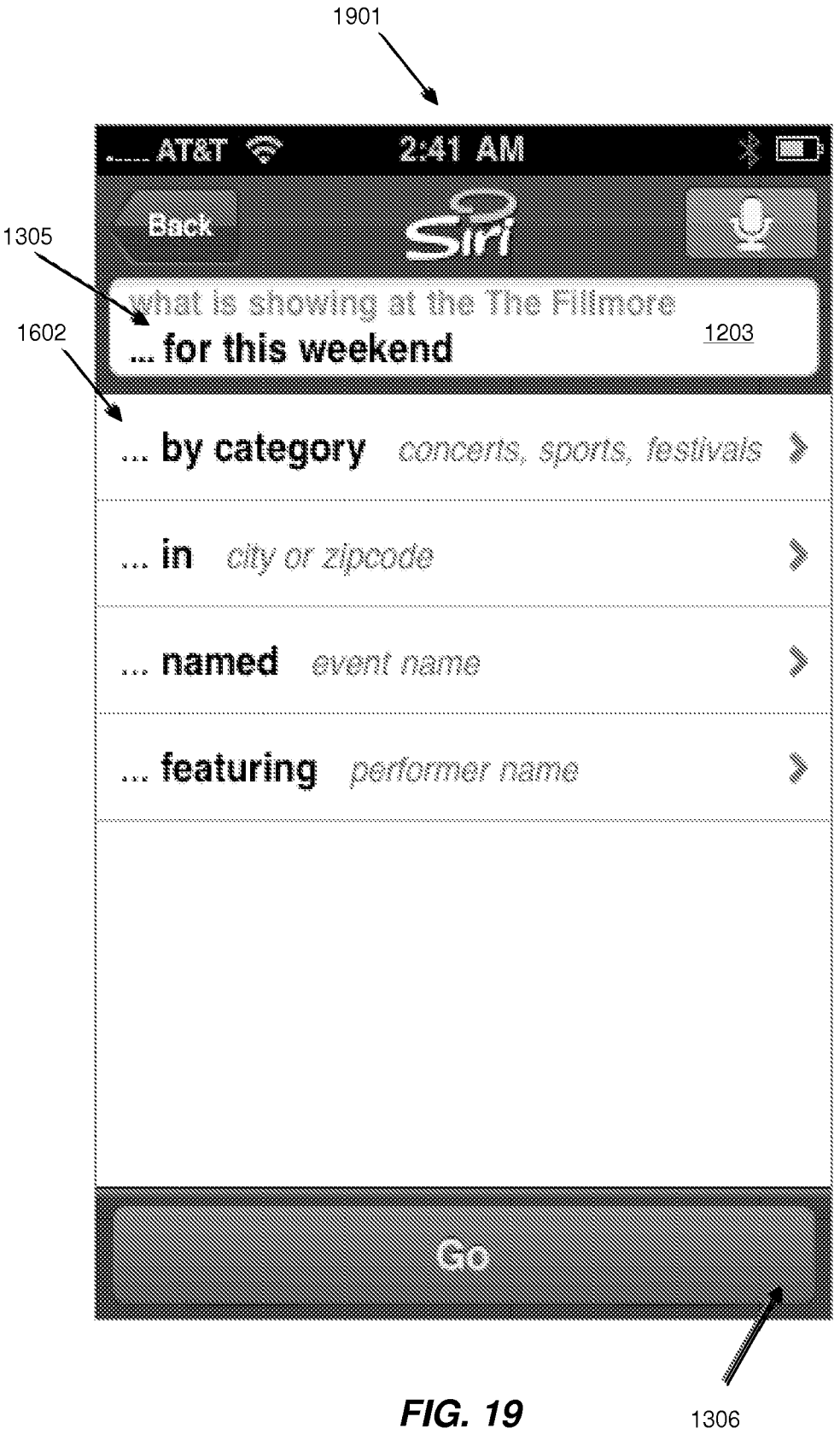

1901
AT&T
2:41 AM
Back
Siri
1305
what is showing at the The Fillmore
... for this weekend
1203
1602
... by category concerts, sports, festivals
... in city or zipcode
... named event name
... featuring performer name
Go
1306

FIG. 19

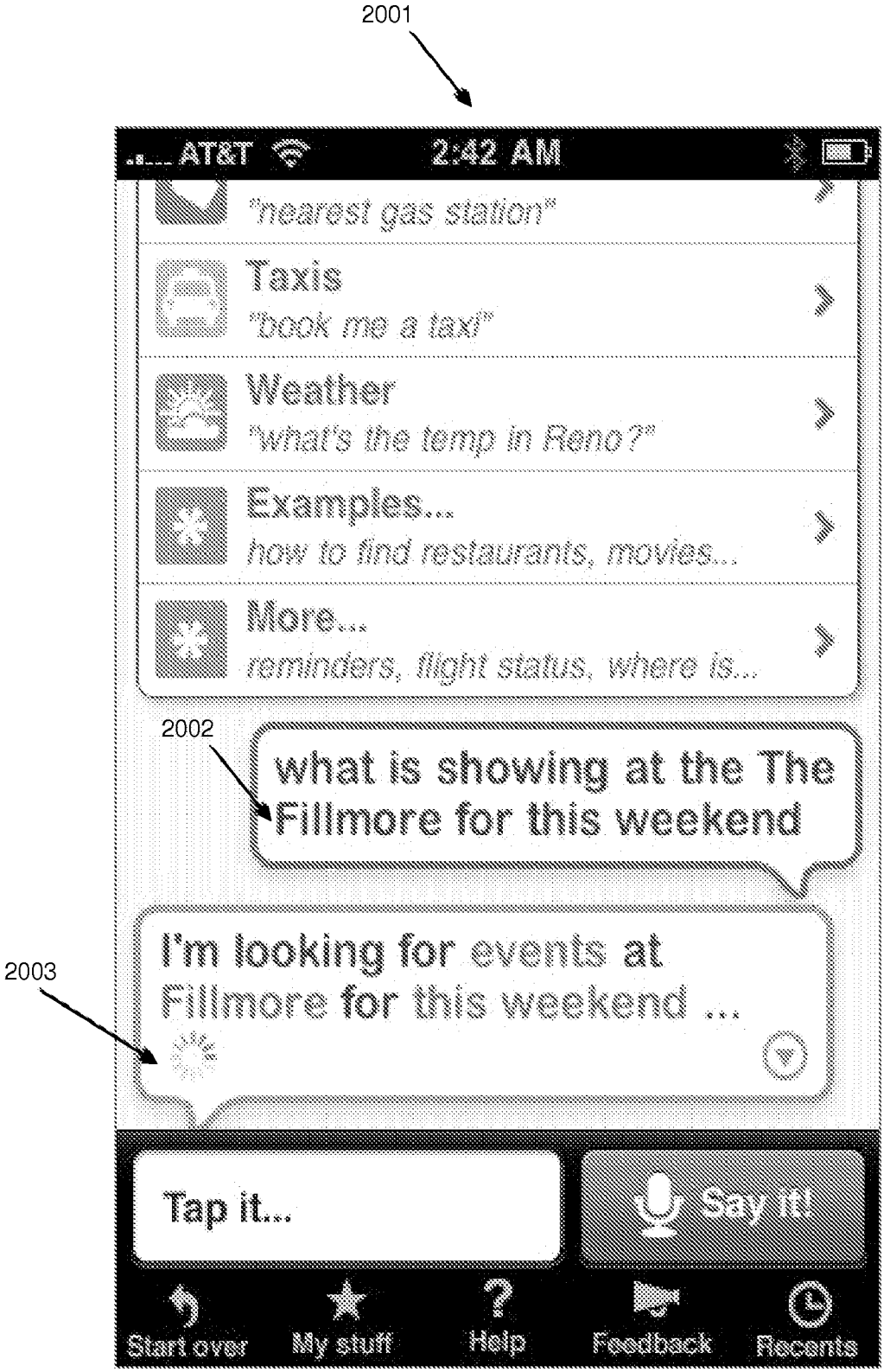

2001
AT&T
2:42 AM
"nearest gas station"
Taxis
"book me a taxi"
Weather
"what's the temp in Reno?"
Examples...
how to find restaurants, movies...
More...
reminders, flight status, where is...
2002
what is showing at the The Fillmore for this weekend
2003
I'm looking for events at Fillmore for this weekend ...
Tap it...
Say it!
Start over
My stuff
Help
Feedback
Recents

FIG. 20


2101
2102
2103
AT&T
2:42 AM
I looked for events at Fillmore for this weekend ... here's what I found:
The Fillmore
San Francisco, 23.1
1 upcoming event
Call
Save
to My Stuff
Share
via email
Map It
Events (1)
Details
My Notes
STATE RADIO
Jan 16, (Saturday) 09:00pm
concert, reggae concert, rock
$20
Siri data from
Tap it...
Say it!
Start over
My stuff
Help
Feedback
Recents

FIG. 21

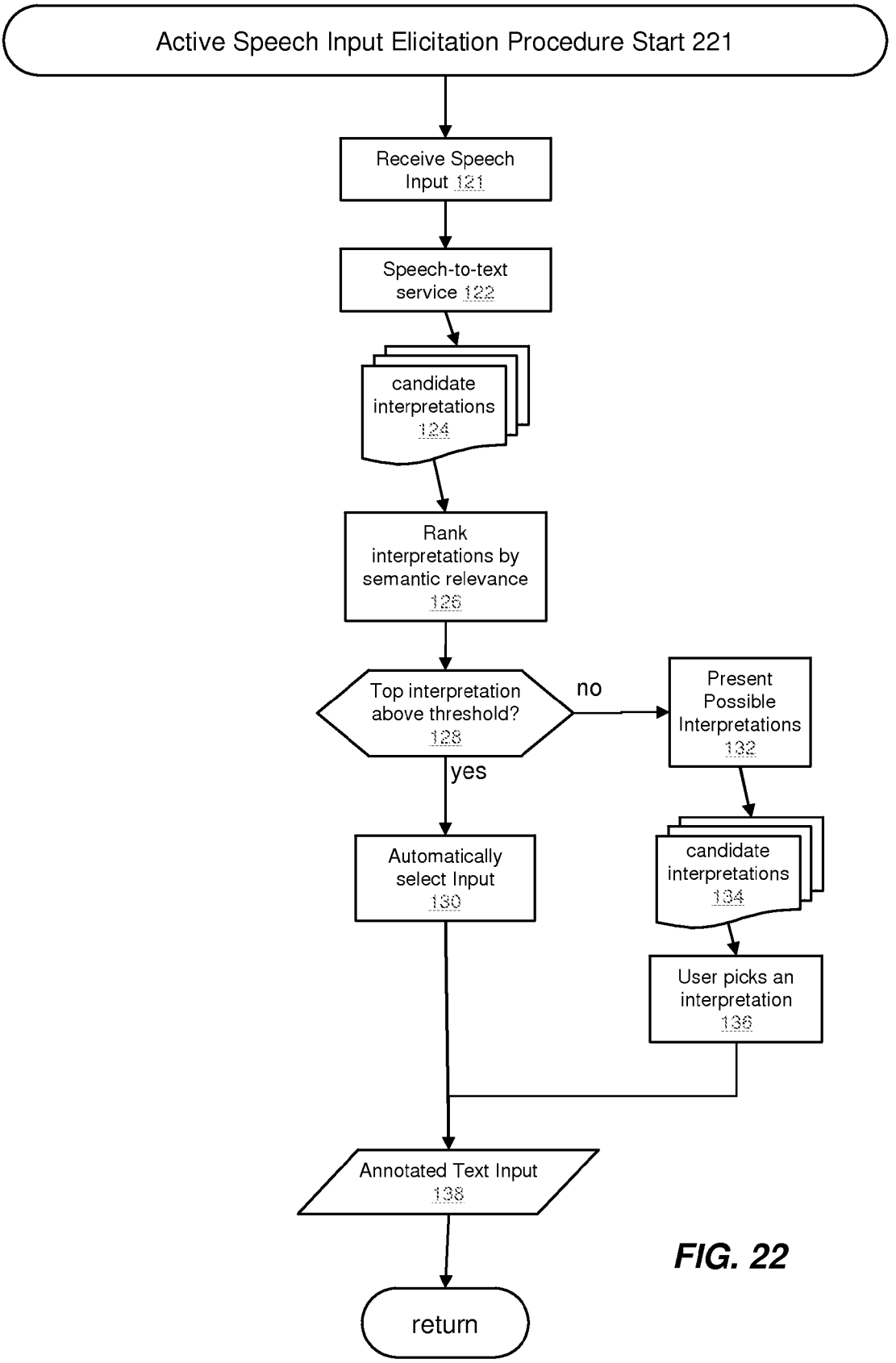

Active Speech Input Elicitation Procedure Start 221
Receive Speech Input 121
Speech-to-text service 122
candidate interpretations 124
Rank interpretations by semantic relevance 126
Top interpretation above threshold? 128
no
yes
Present Possible Interpretations 132
Automatically select Input 130
candidate interpretations 134
User picks an interpretation 136
Annotated Text Input 138
return

FIG. 22

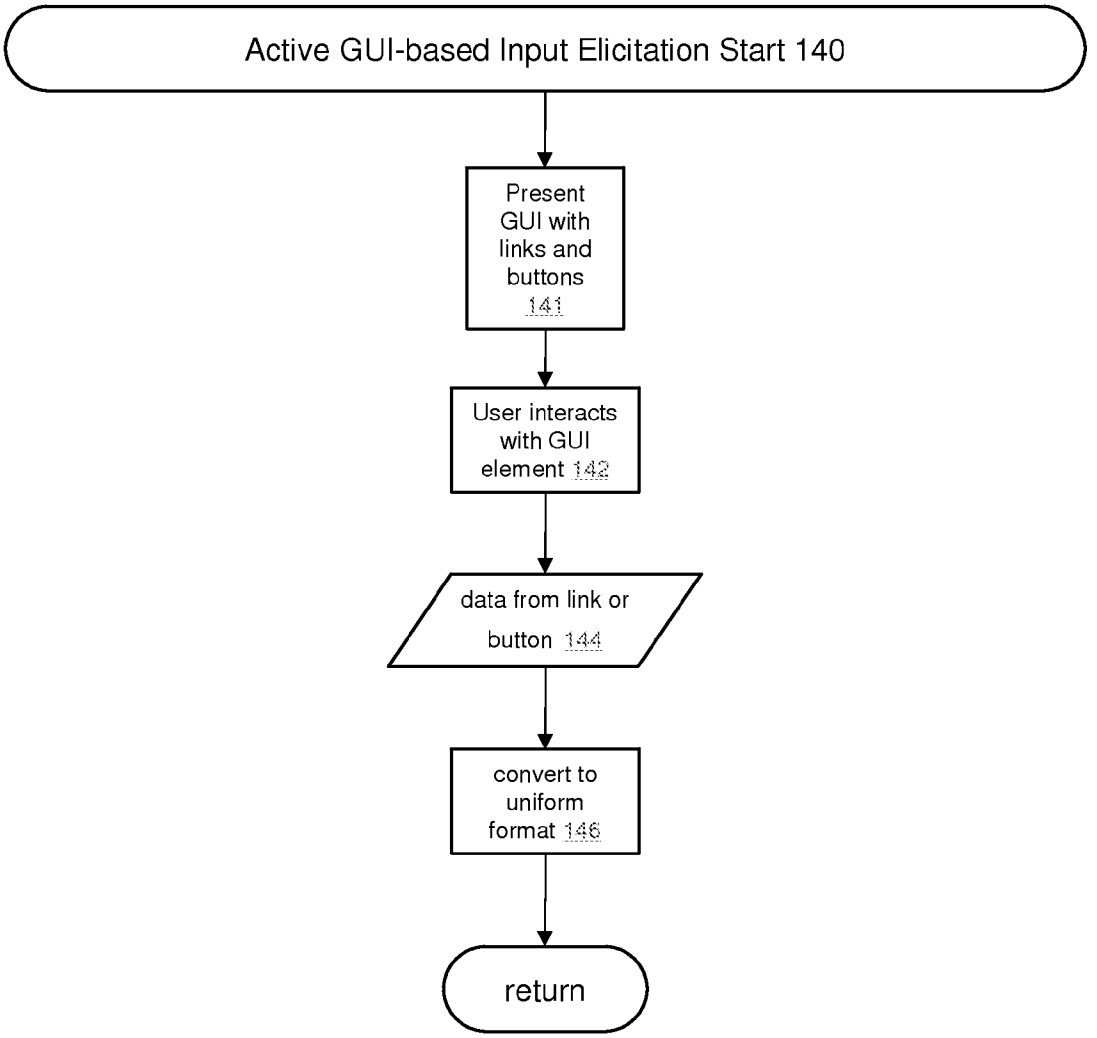

Active GUI-based Input Elicitation Start 140
Present GUI with links and buttons 141
User interacts with GUI element 142
data from link or button 144
convert to uniform format 146
return

FIG. 23

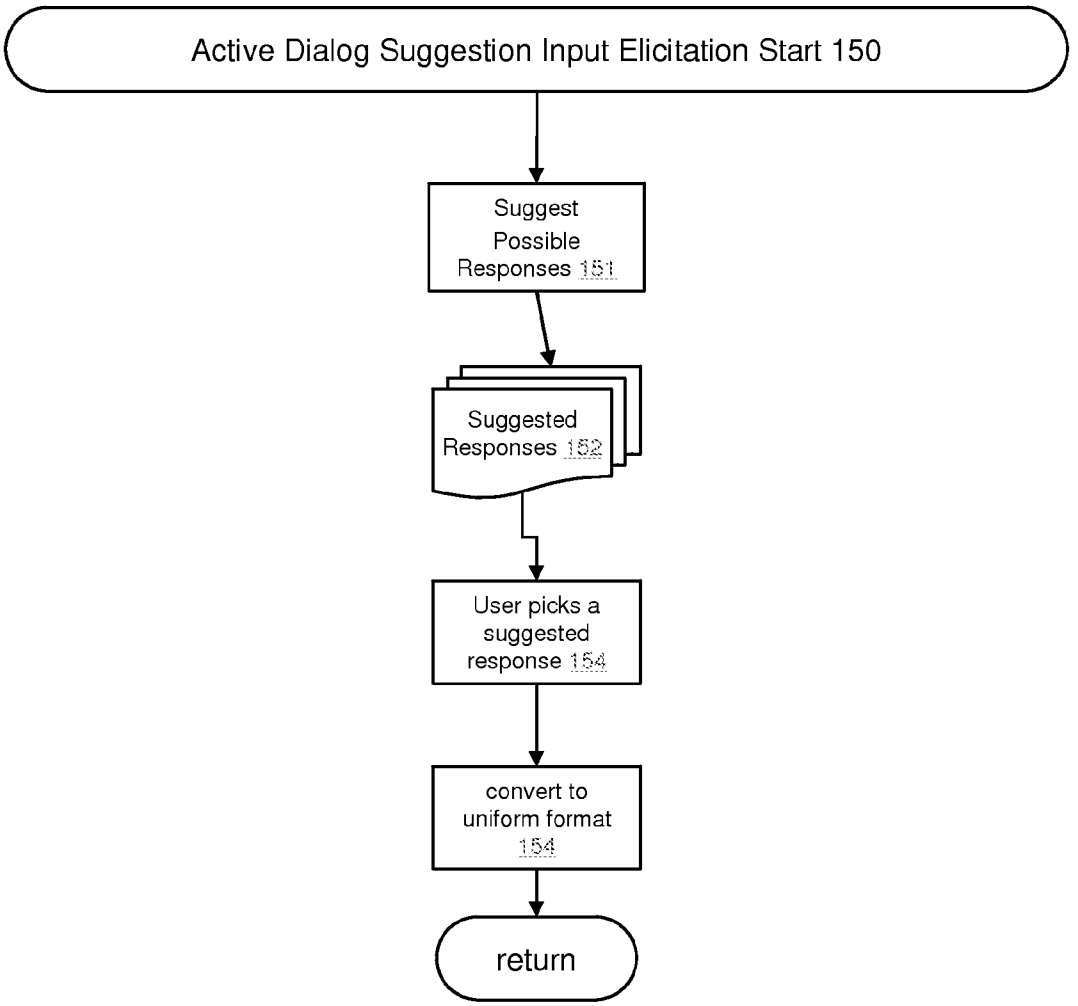

Active Dialog Suggestion Input Elicitation Start 150
Suggest Possible Responses 151
Suggested Responses 152
User picks a suggested response 154
convert to uniform format 154
return

FIG. 24

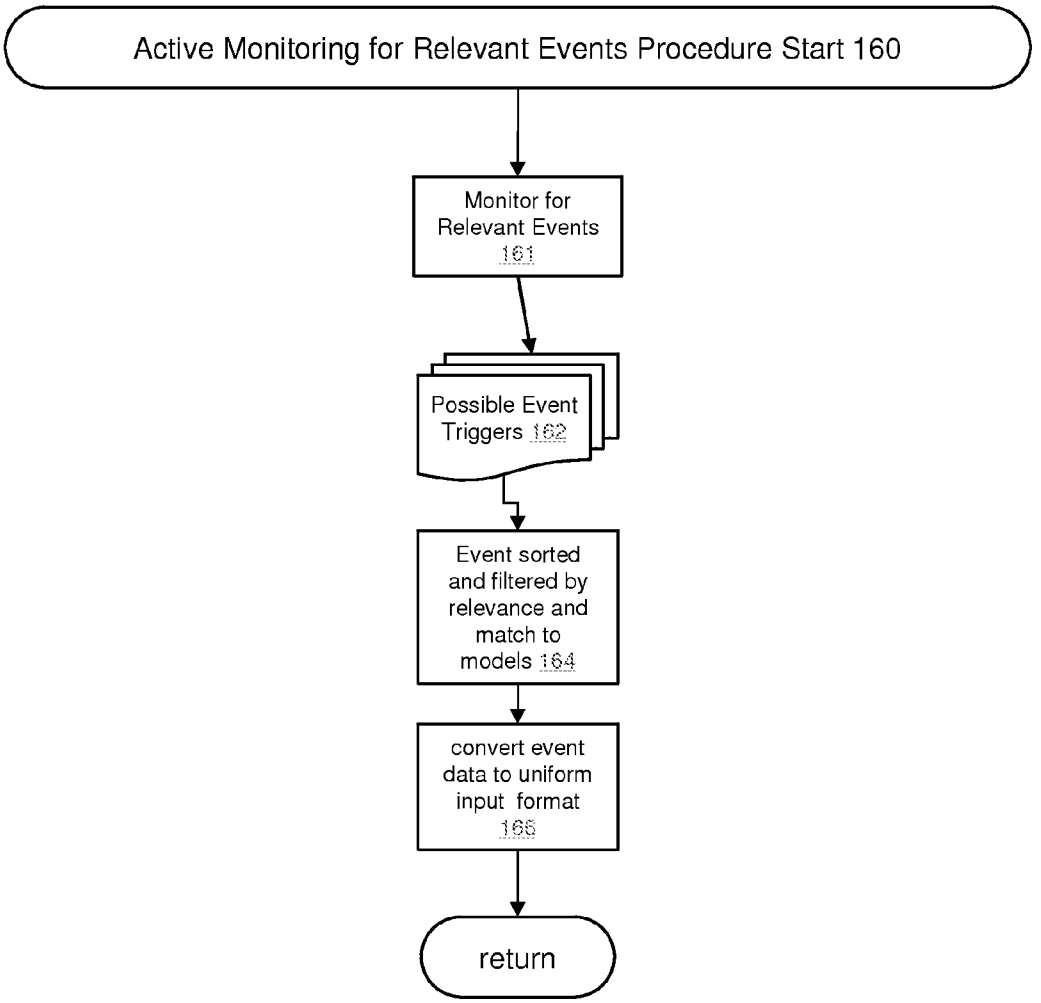

Active Monitoring for Relevant Events Procedure Start 160
Monitor for Relevant Events 161
Possible Event Triggers 162
Event sorted and filtered by relevance and match to models 164
convert event data to uniform input format 166
return

FIG. 25

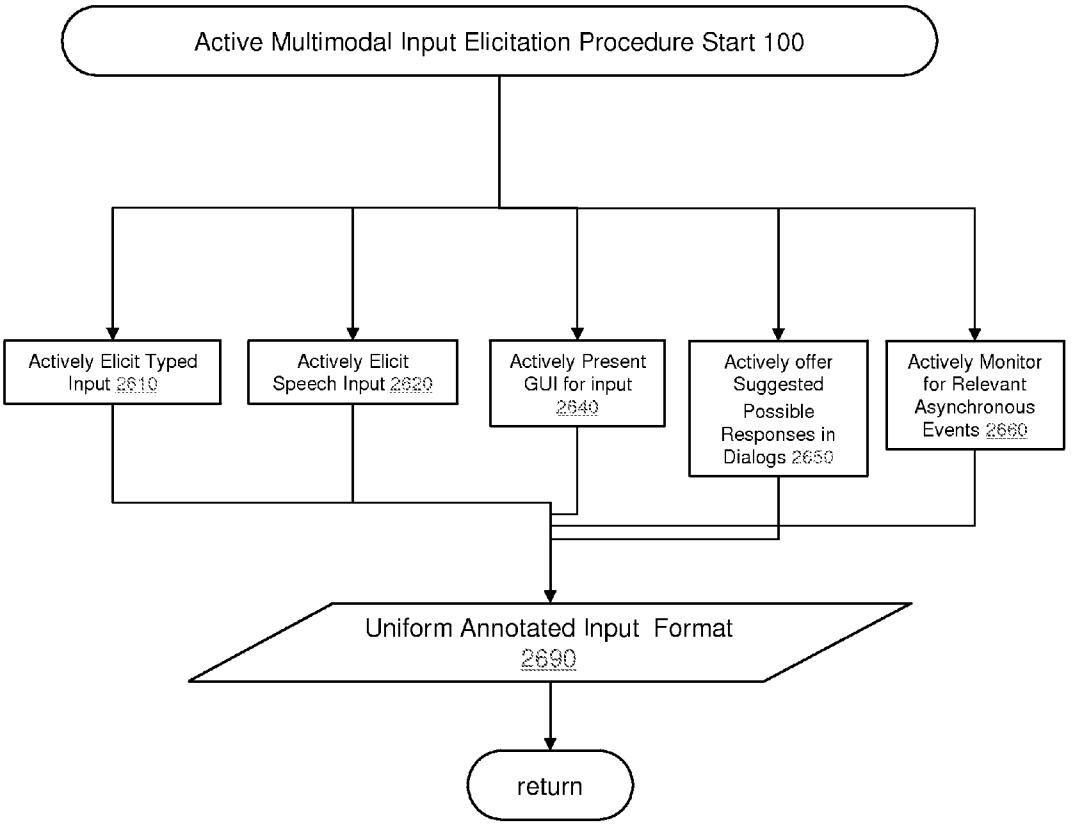

Active Multimodal Input Elicitation Procedure Start 100
Actively Elicit Typed Input 2610
Actively Elicit Speech Input 2620
Actively Present GUI for input 2640
Actively offer Suggested Possible Responses in Dialogs 2650
Actively Monitor for Relevant Asynchronous Events 2660
Uniform Annotated Input Format 2690
return

FIG. 26

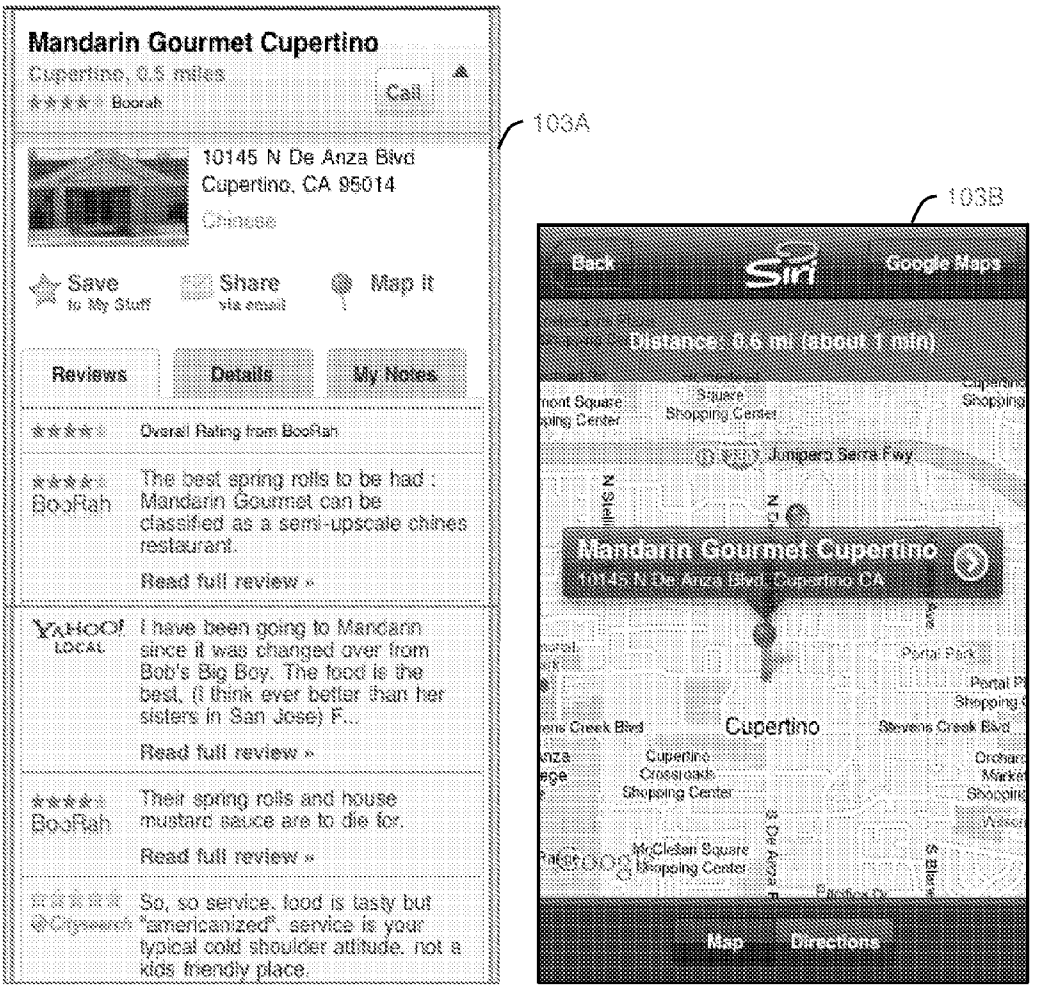

Mandarin Gourmet Cupertino
Cupertino, 0.5 miles
Call
103A
10145 N De Anza Blvd
Cupertino, CA 95014
Chinese
Save
to My Stuff
Share
via email
Map it
Reviews
Details
My Notes
Overall Rating from BooRah
The best spring rolls to be had : Mandarin Gourmet can be classified as a semi-upscale chines restaurant.
Read full review »
I have been going to Mandarin since it was changed over from Bob's Big Boy. The food is the best, (I think ever better than her sisters in San Jose) F...
Read full review »
Their spring rolls and house mustard sauce are to die for.
Read full review »
So, so service, food is tasty but "americanized", service is your typical cold shoulder attitude, not a kids friendly place.
103B
Back
Siri
Google Maps
Distance: 0.6 mi (about 1 min)
Square Shopping Center
Jumipero Serra Fwy
N Stelling
N De Anza
Mandarin Gourmet Cupertino
10145 N De Anza Blvd Cupertino CA
Portal Park
Cupertino
Stevens Creek Blvd
Cupertino Crossroads Shopping Center
McClellan Square Shopping Center
S De Anza Blvd
Map
Directions

FIG. 27

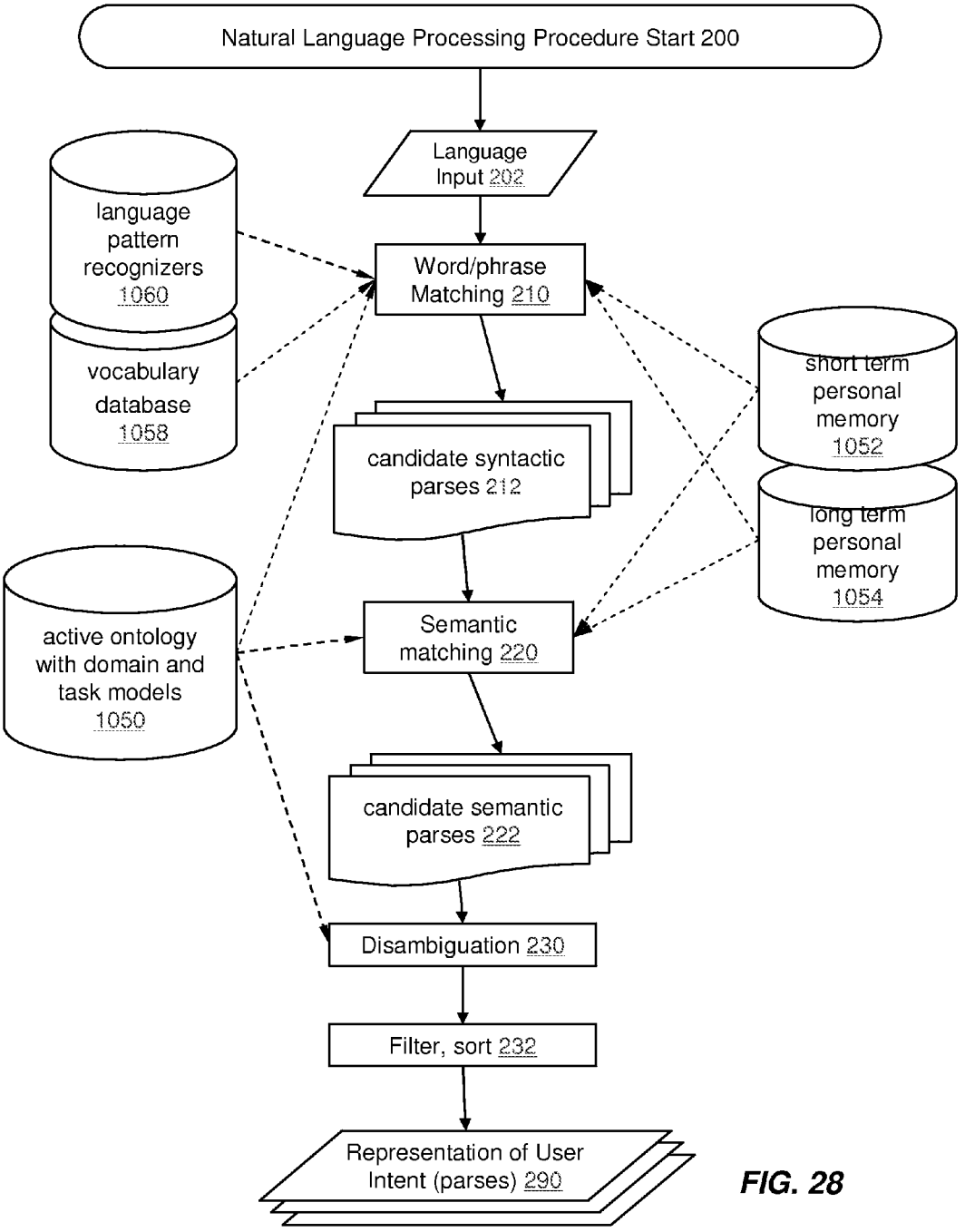

Natural Language Processing Procedure Start 200
Language Input 202
language pattern recognizers 1060
vocabulary database 1058
Word/phrase Matching 210
candidate syntactic parses 212
short term personal memory 1052
long term personal memory 1054
active ontology with domain and task models 1050
Semantic matching 220
candidate semantic parses 222
Disambiguation 230
Filter, sort 232
Representation of User Intent (parses) 290

FIG. 28

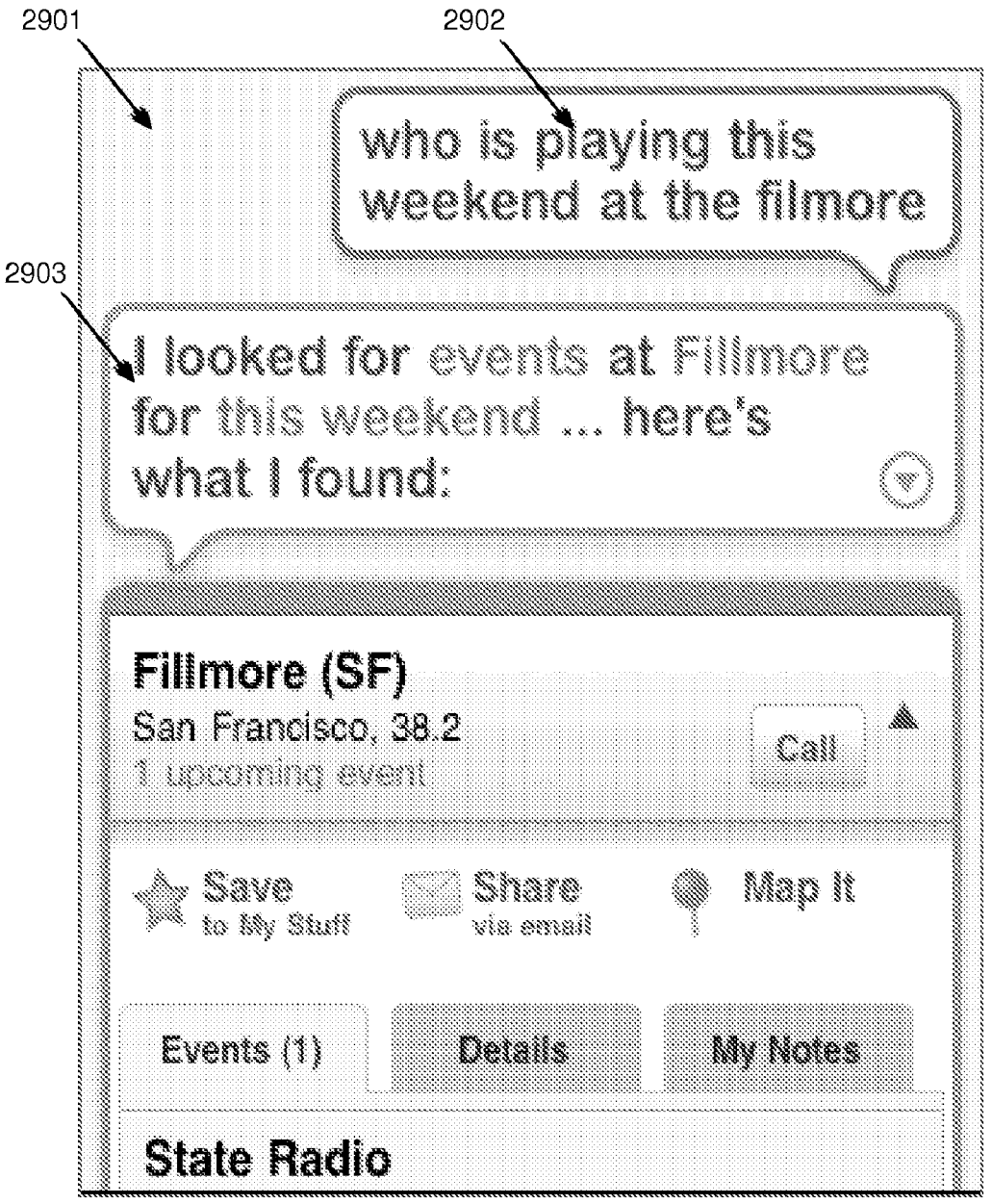

2901
2902
who is playing this weekend at the filmore
2903
I looked for events at Fillmore for this weekend ... here's what I found:
Fillmore (SF)
San Francisco, 38.2
1 upcoming event
Call
Save
to My Stuff
Share
via email
Map It
Events (1)
Details
My Notes
State Radio

FIG. 29

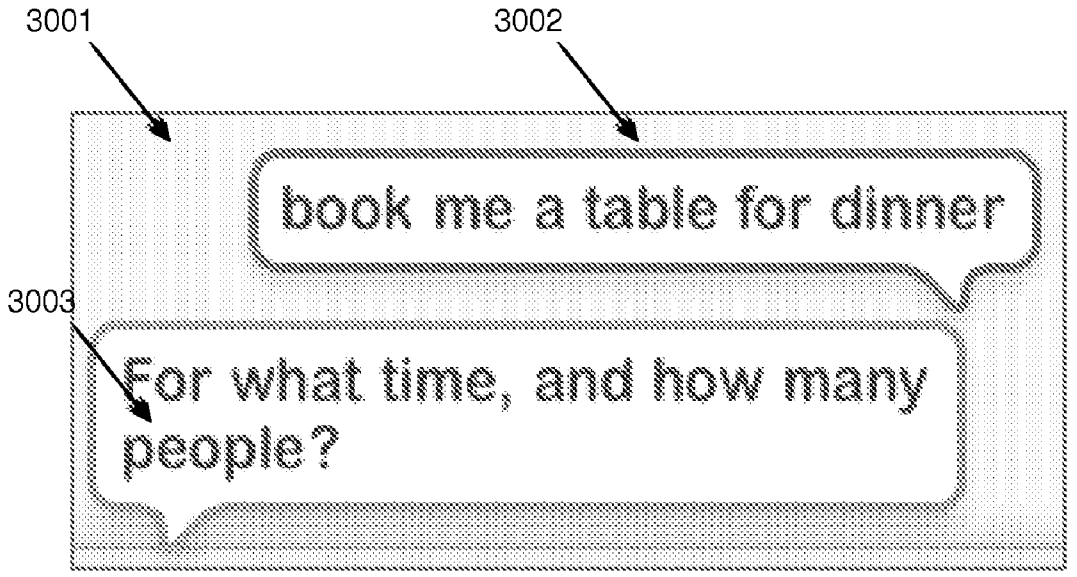

3001
3002
book me a table for dinner
3003
For what time, and how many people?

FIG. 30

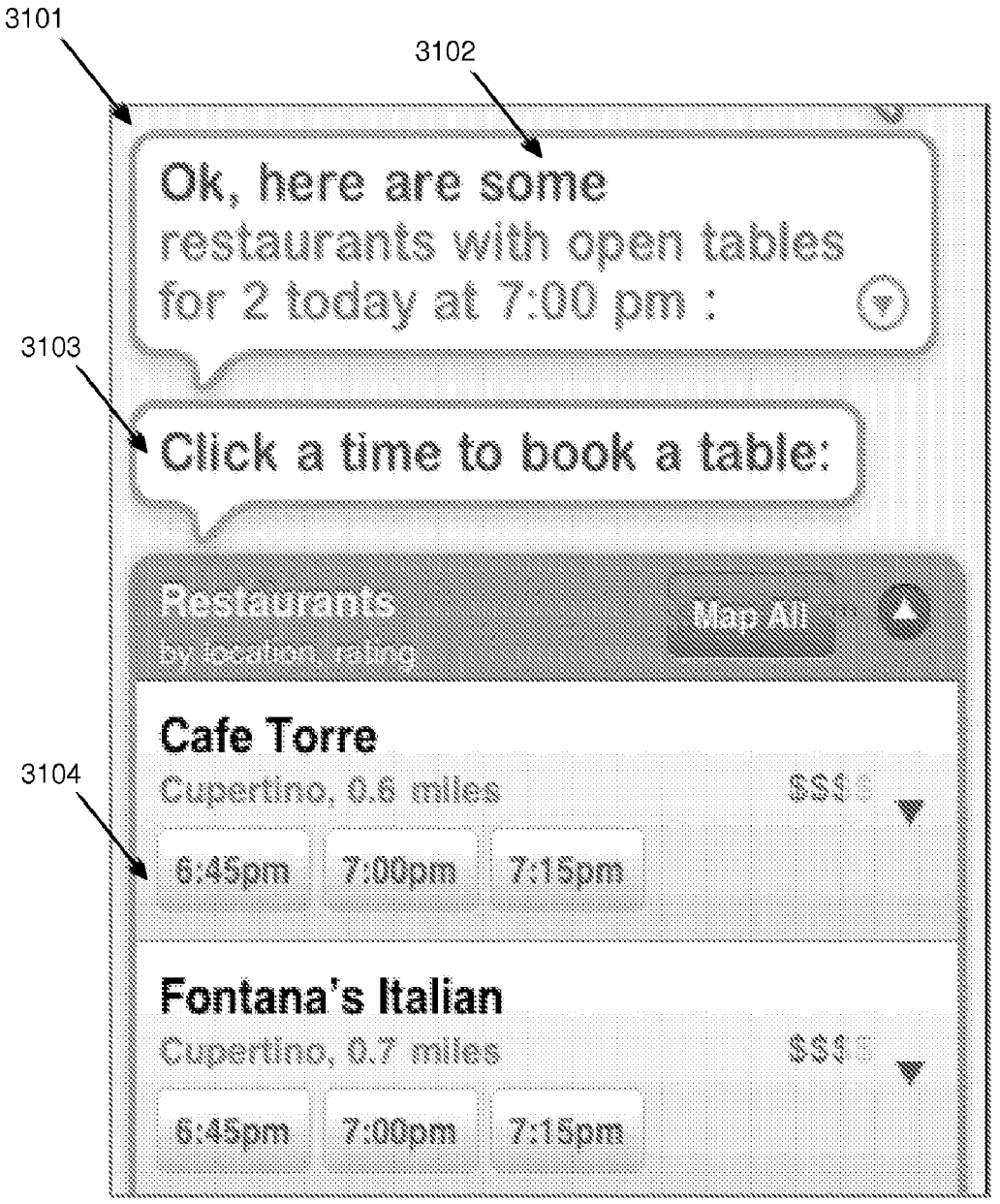

3101
3102
Ok, here are some
restaurants with open tables
for 2 today at 7:00 pm :
3103
Click a time to book a table:
Restaurants
Map All
Cafe Torre
Cupertino, 0.6 miles
$$$
3104
6:45pm
7:00pm
7:15pm
Fontana's Italian
Cupertino, 0.7 miles
$$$
6:45pm
7:00pm
7:15pm

FIG. 31

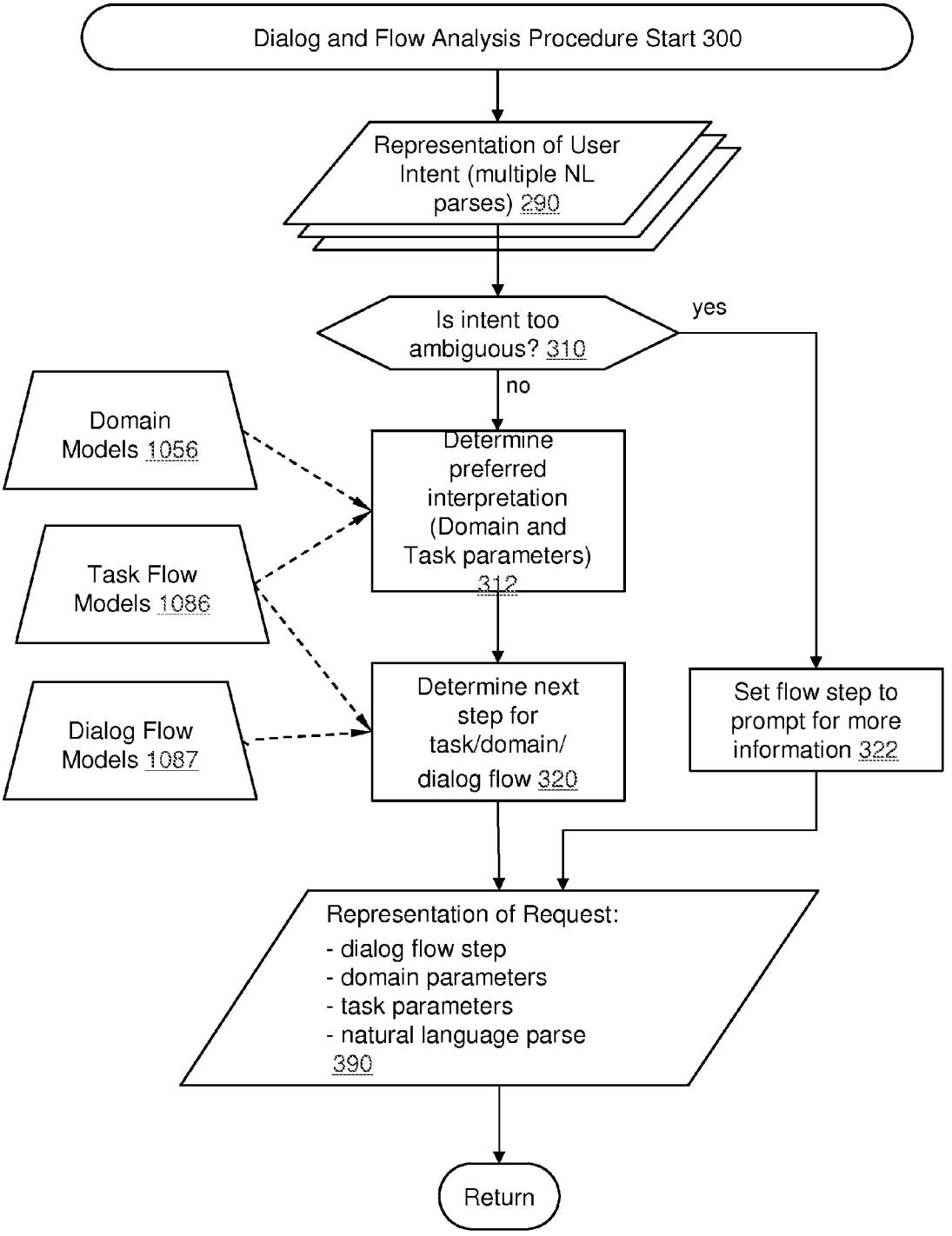

Dialog and Flow Analysis Procedure Start 300
Representation of User Intent (multiple NL parses) 290
Is intent too ambiguous? 310
yes
no
Domain Models 1056
Task Flow Models 1086
Dialog Flow Models 1087
Determine preferred interpretation (Domain and Task parameters) 312
Determine next step for task/domain/ dialog flow 320
Set flow step to prompt for more information 322
Representation of Request:
- dialog flow step
- domain parameters
- task parameters
- natural language parse
390
Return

FIG. 32

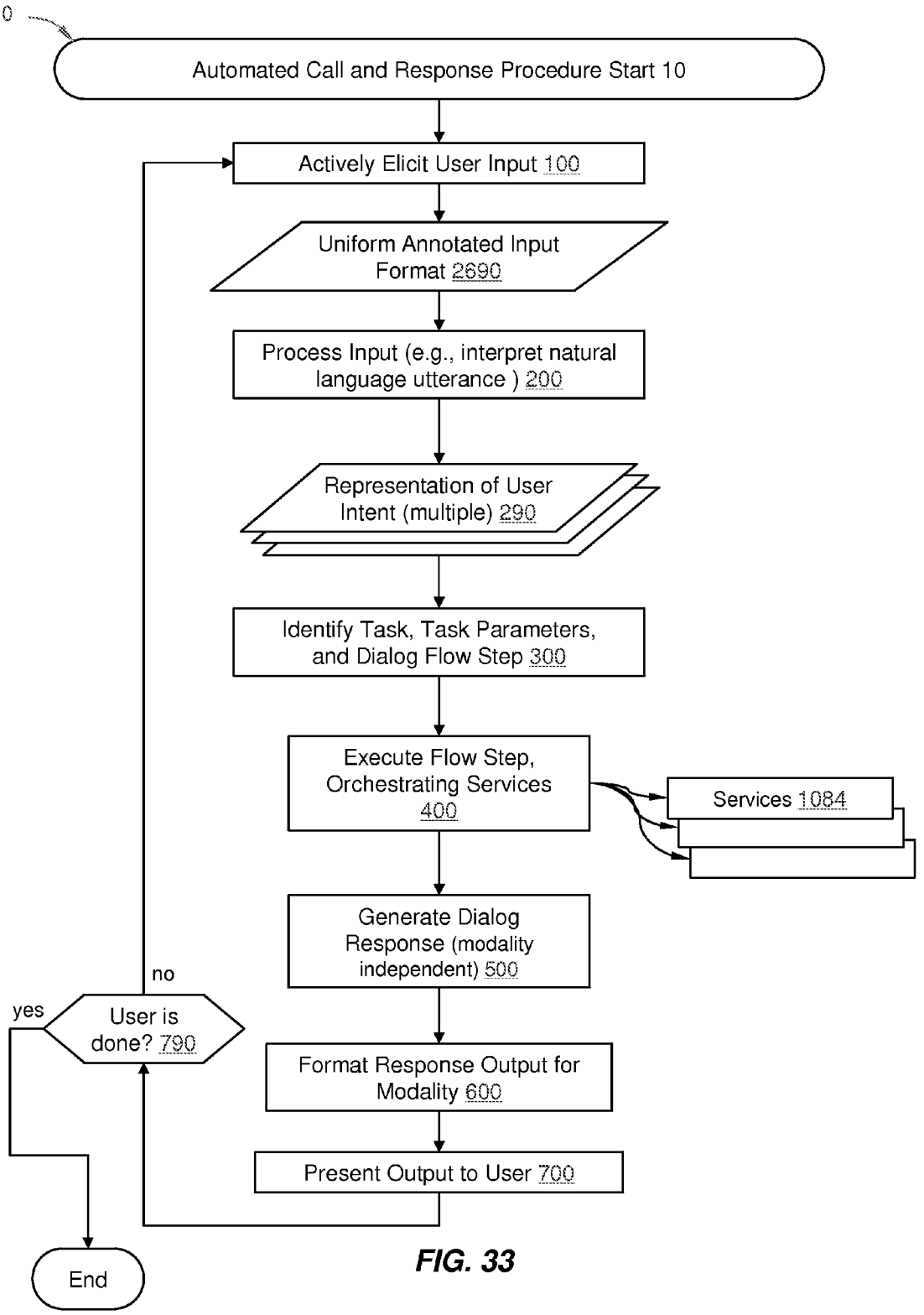

10
Automated Call and Response Procedure Start 10
Actively Elicit User Input 100
Uniform Annotated Input Format 2690
Process Input (e.g., interpret natural language utterance) 200
Representation of User Intent (multiple) 290
Identify Task, Task Parameters, and Dialog Flow Step 300
Execute Flow Step, Orchestrating Services 400
Services 1084
Generate Dialog Response (modality independent) 500
no
yes
User is done? 790
Format Response Output for Modality 600
Present Output to User 700
End

FIG. 33

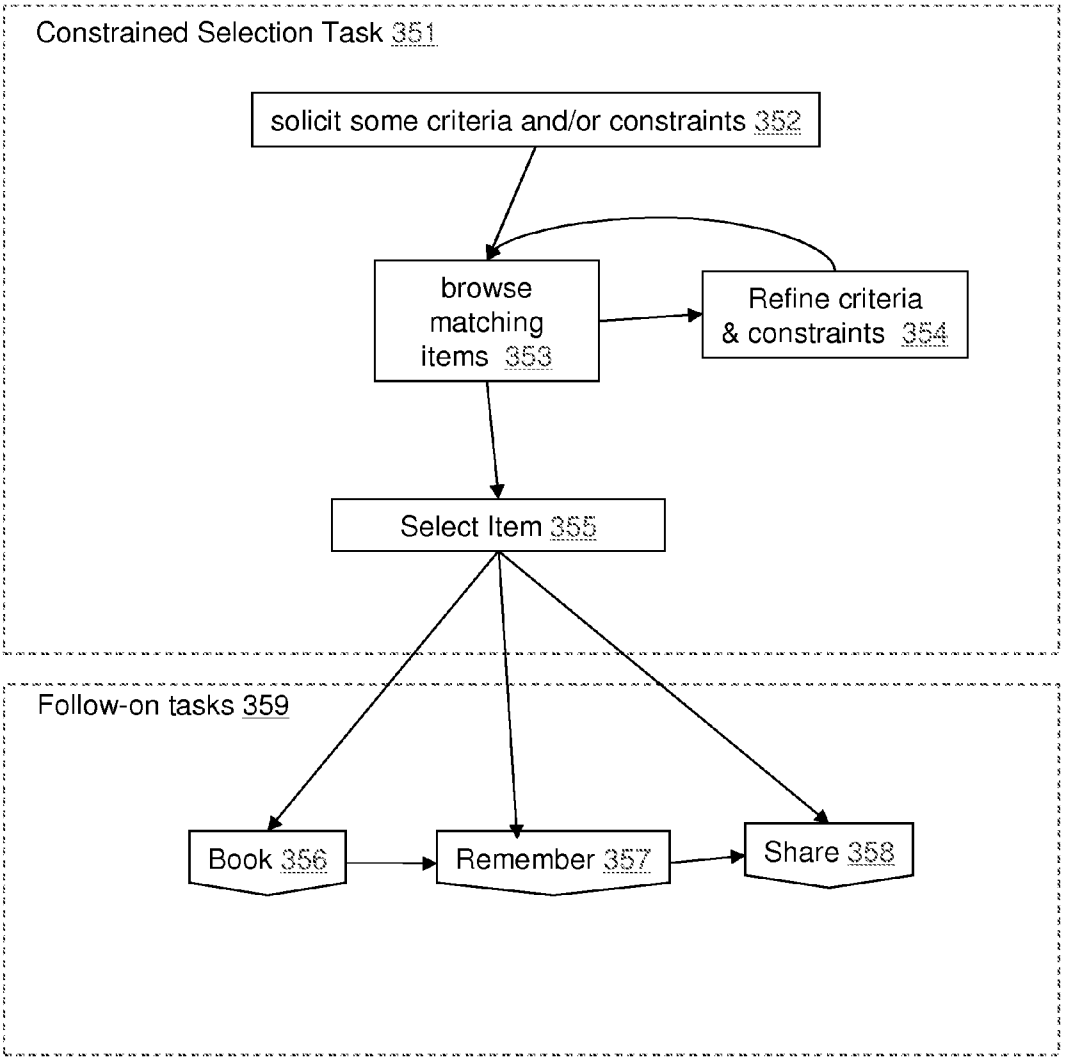

Constrained Selection Task 351
solicit some criteria and/or constraints 352
browse matching items 353
Refine criteria & constraints 354
Select Item 355
Follow-on tasks 359
Book 356
Remember 357
Share 358

FIG. 34

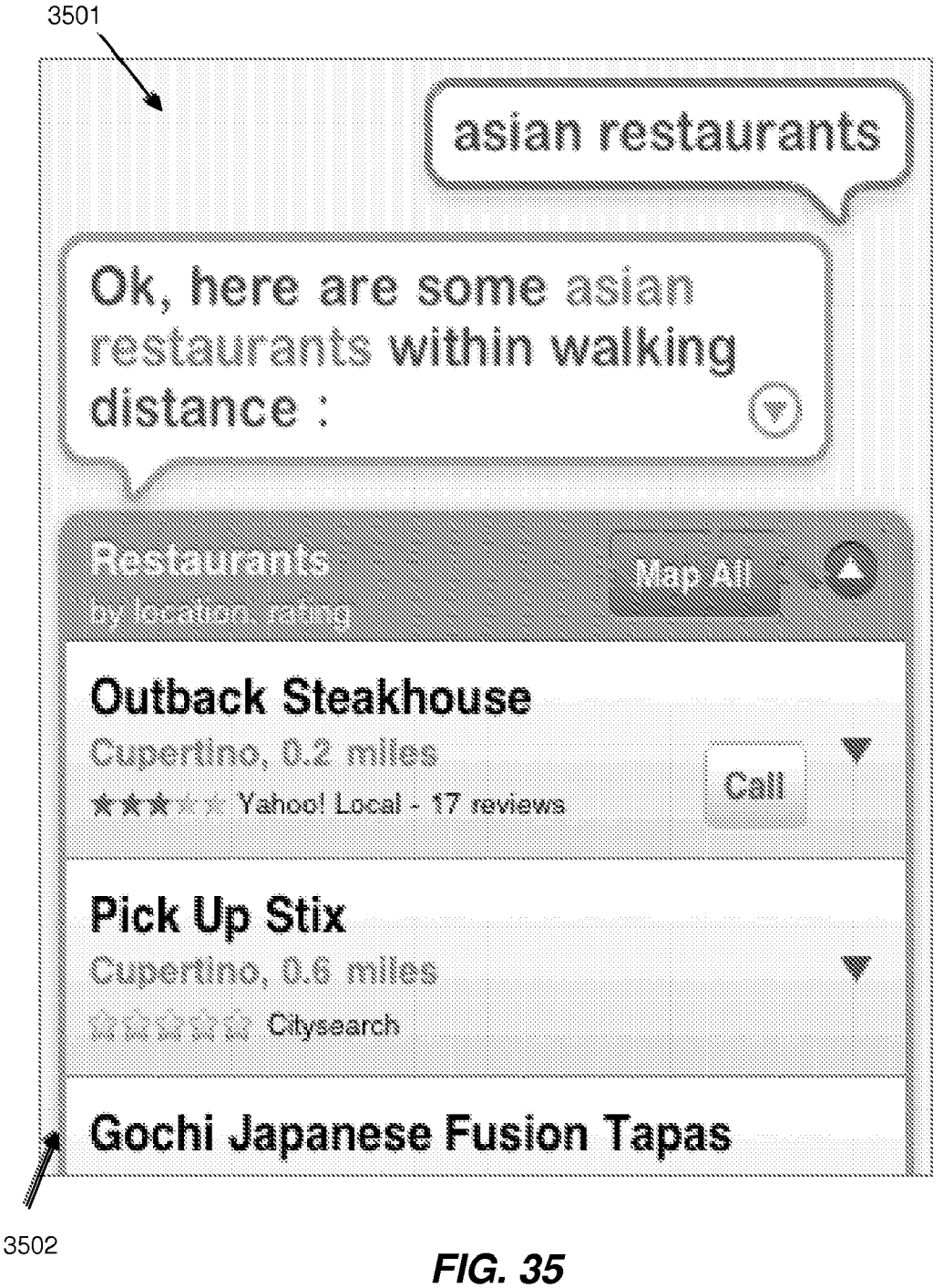

3501
asian restaurants
Ok, here are some asian restaurants within walking distance :
Restaurants
Map All
Outback Steakhouse
Cupertino, 0.2 miles
Yahoo! Local - 17 reviews
Call
Pick Up Stix
Cupertino, 0.6 miles
Citysearch
Gochi Japanese Fusion Tapas
3502

FIG. 35

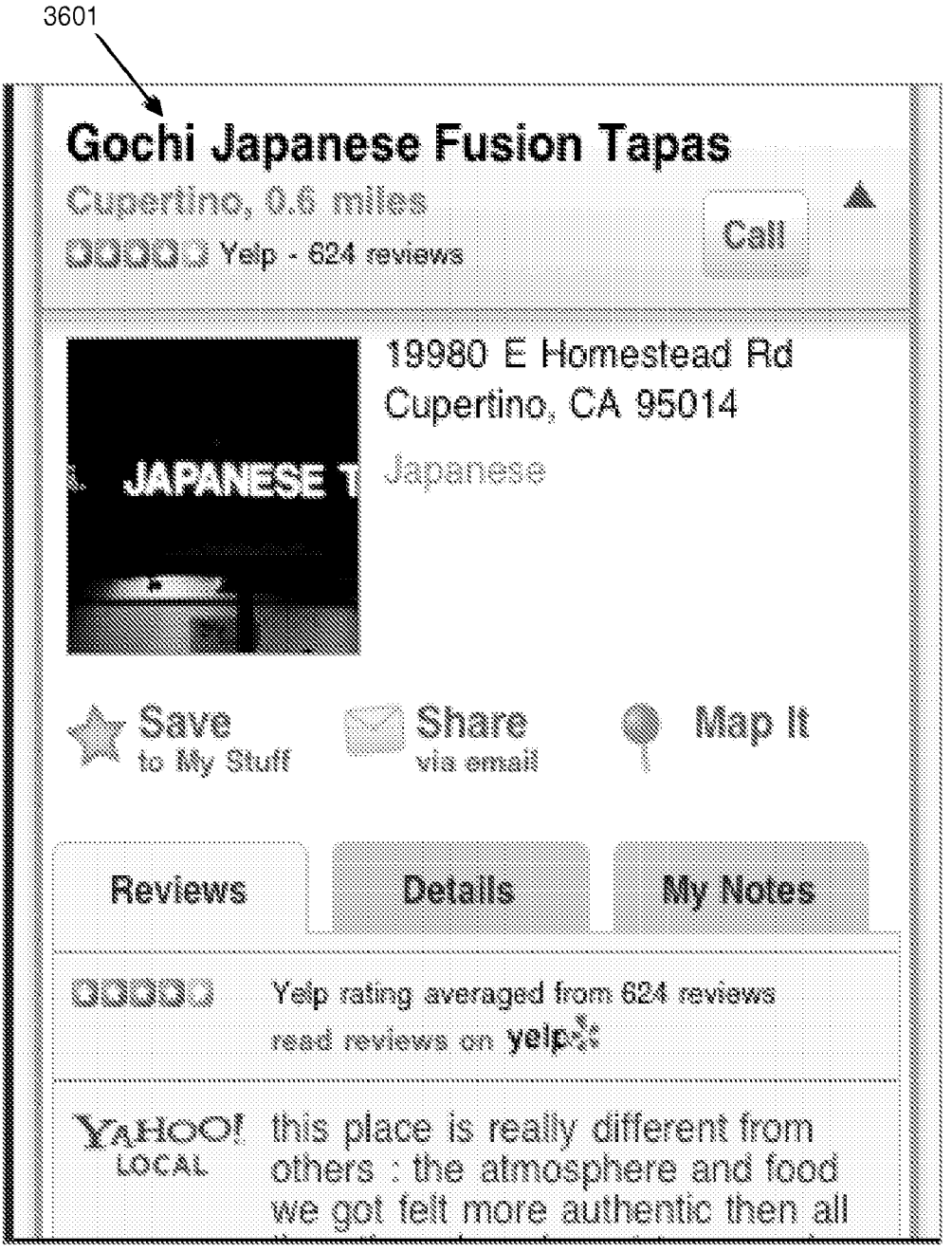

3601
Gochi Japanese Fusion Tapas
Cupertino, 0.6 miles
Yelp - 624 reviews
Call
19980 E Homestead Rd
Cupertino, CA 95014
Japanese
Save
to My Stuff
Share
via email
Map It
Reviews
Details
My Notes
Yelp rating averaged from 624 reviews
read reviews on yelp
YAHOO! LOCAL
this place is really different from others : the atmosphere and food we got felt more authentic then all

FIG. 36

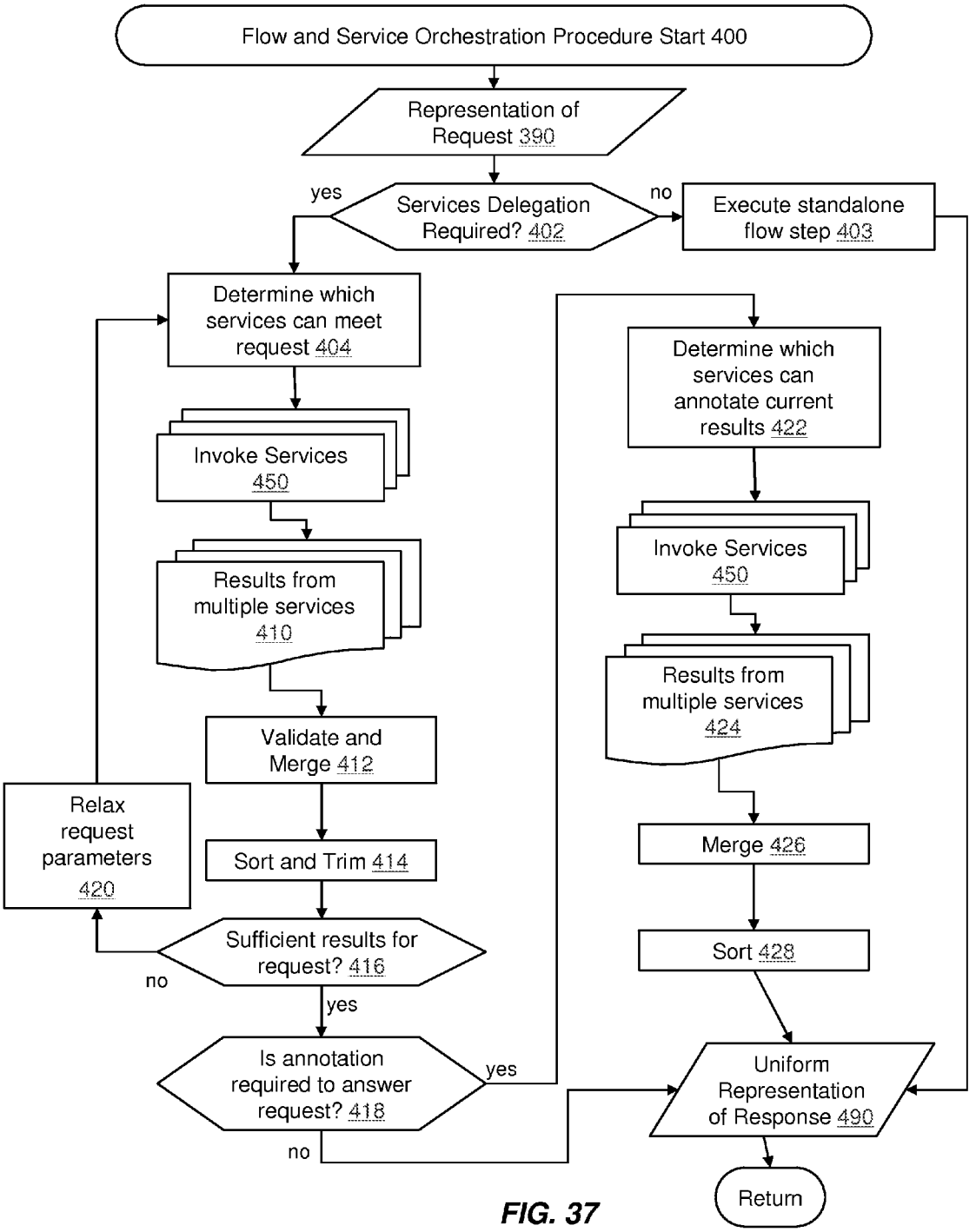

Flow and Service Orchestration Procedure Start 400
Representation of Request 390
Services Delegation Required? 402
yes
no
Execute standalone flow step 403
Determine which services can meet request 404
Invoke Services 450
Results from multiple services 410
Validate and Merge 412
Sort and Trim 414
Sufficient results for request? 416
no
yes
Relax request parameters 420
Is annotation required to answer request? 418
yes
no
Determine which services can annotate current results 422
Invoke Services 450
Results from multiple services 424
Merge 426
Sort 428
Uniform Representation of Response 490
Return

FIG. 37

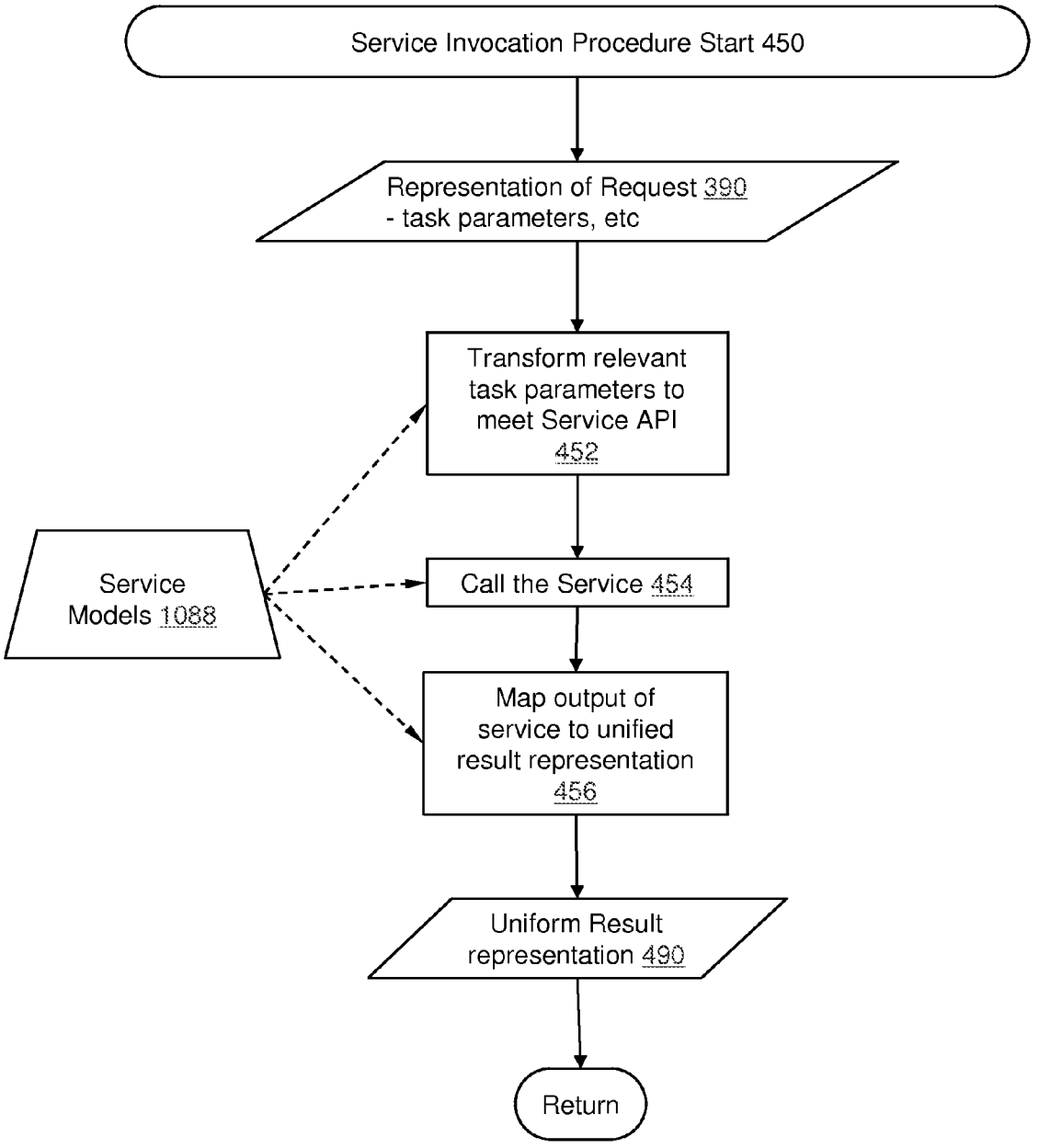

Service Invocation Procedure Start 450
Representation of Request 390
- task parameters, etc
Transform relevant task parameters to meet Service API 452
Service Models 1088
Call the Service 454
Map output of service to unified result representation 456
Uniform Result representation 490
Return

FIG. 38

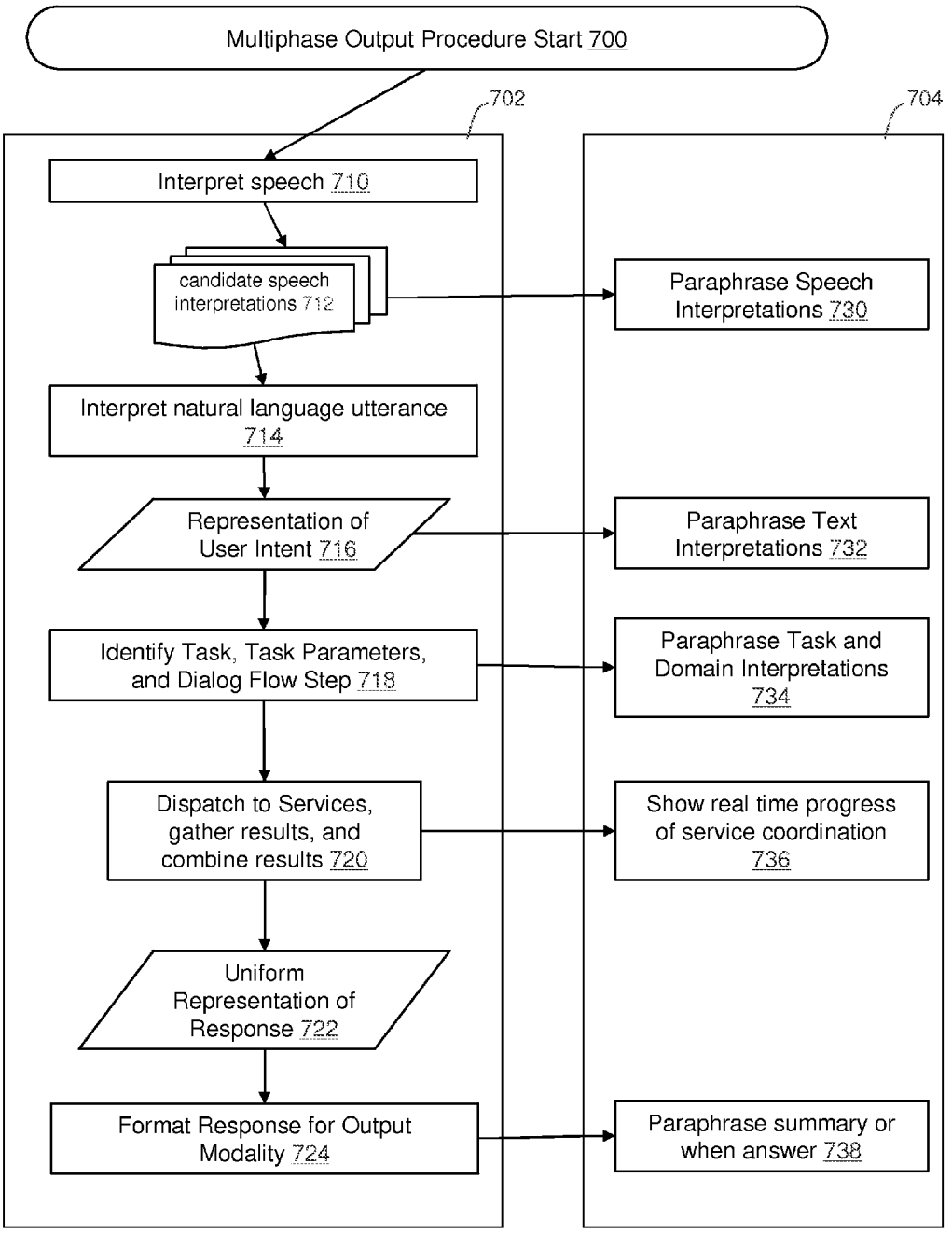

Multiphase Output Procedure Start 700
702
704
Interpret speech 710
candidate speech interpretations 712
Paraphrase Speech Interpretations 730
Interpret natural language utterance 714
Representation of User Intent 716
Paraphrase Text Interpretations 732
Identify Task, Task Parameters, and Dialog Flow Step 718
Paraphrase Task and Domain Interpretations 734
Dispatch to Services, gather results, and combine results 720
Show real time progress of service coordination 736
Uniform Representation of Response 722
Format Response for Output Modality 724
Paraphrase summary or when answer 738

FIG. 39

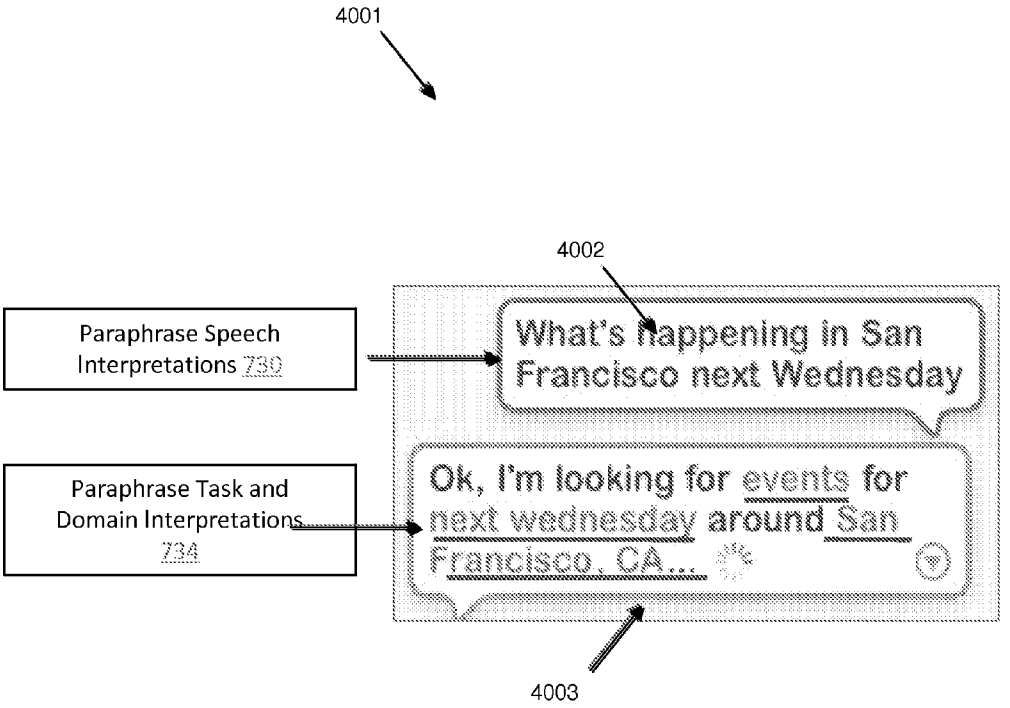

4001
4002
Paraphrase Speech Interpretations 730
What's happening in San Francisco next Wednesday
Paraphrase Task and Domain Interpretations 734
Ok, I'm looking for events for next wednesday around San Francisco, CA...
4003

FIG. 40

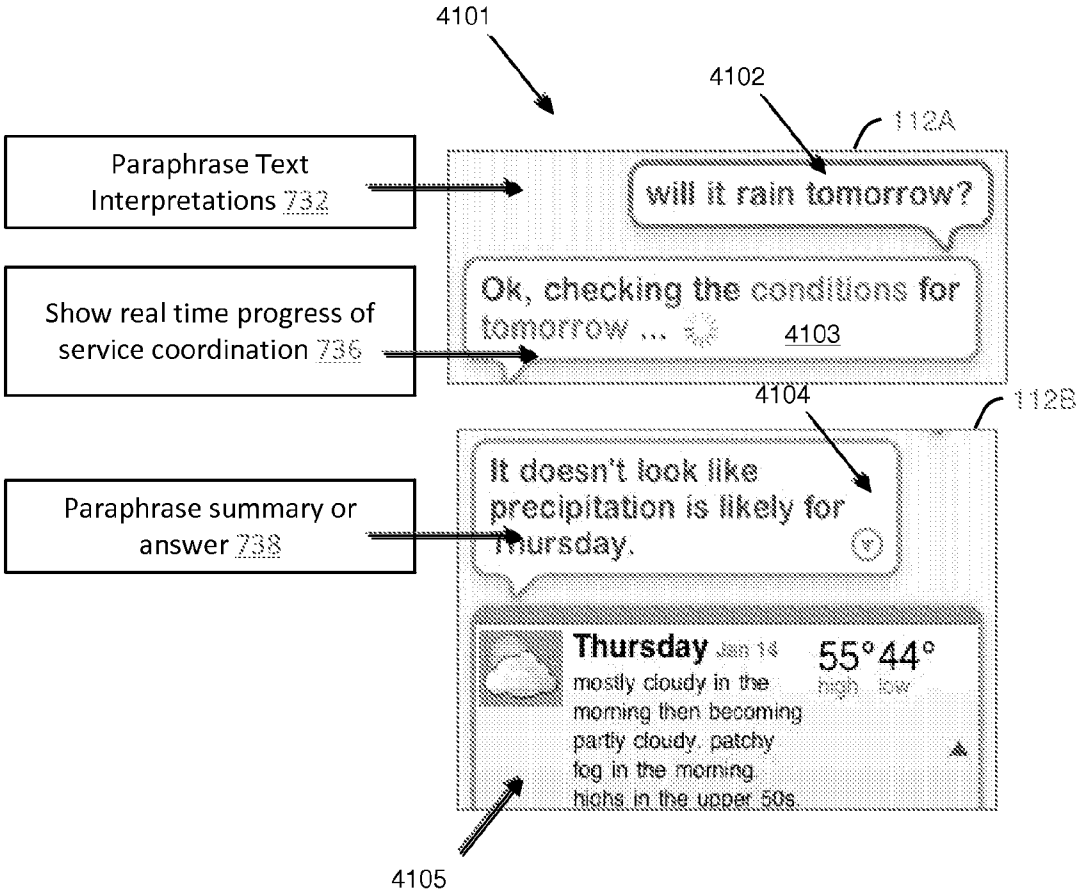

4101
4102
112A
Paraphrase Text Interpretations 732
will it rain tomorrow?
Ok, checking the conditions for tomorrow ...
4103
Show real time progress of service coordination 736
4104
112B
It doesn't look like precipitation is likely for Thursday.
Paraphrase summary or answer 738
Thursday
55°44°
mostly cloudy in the morning then becoming partly cloudy. patchy fog in the morning. highs in the upper 50s.
4105

FIG. 41

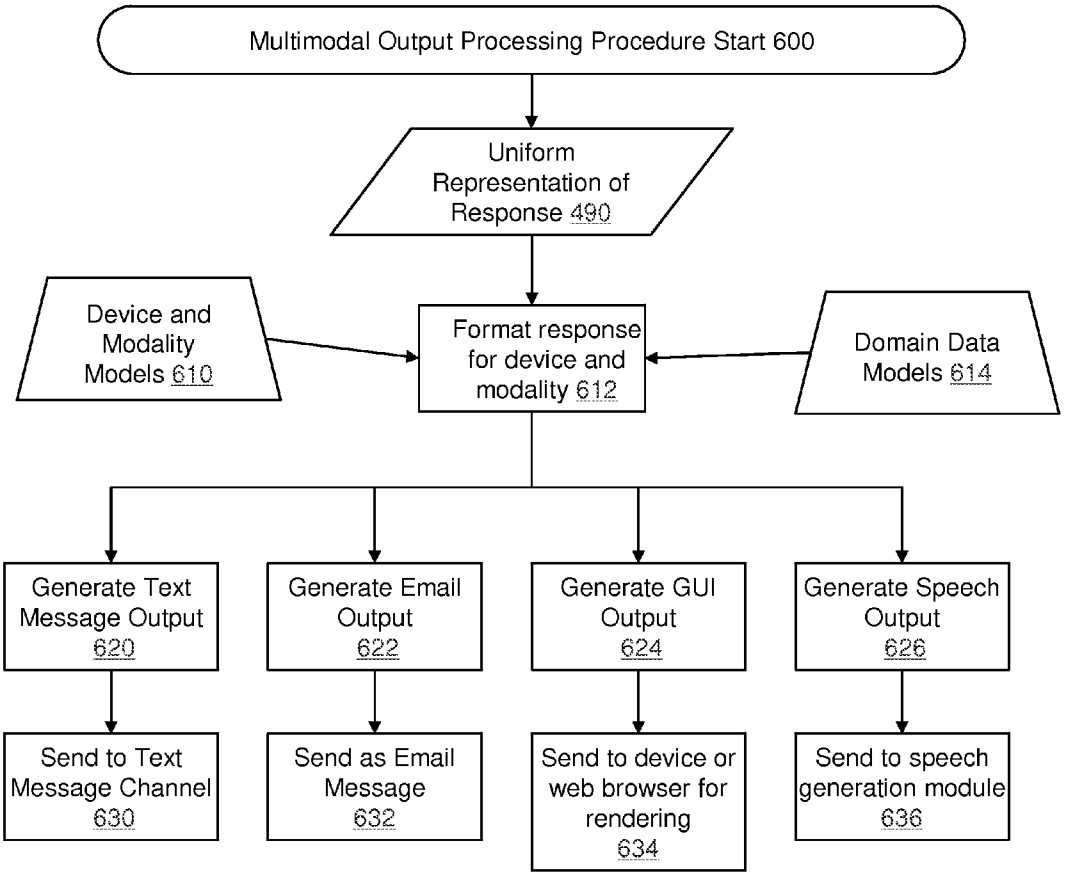

Multimodal Output Processing Procedure Start 600
Uniform Representation of Response 490
Device and Modality Models 610
Format response for device and modality 612
Domain Data Models 614
Generate Text Message Output 620
Generate Email Output 622
Generate GUI Output 624
Generate Speech Output 626
Send to Text Message Channel 630
Send as Email Message 632
Send to device or web browser for rendering 634
Send to speech generation module 636

FIG. 42

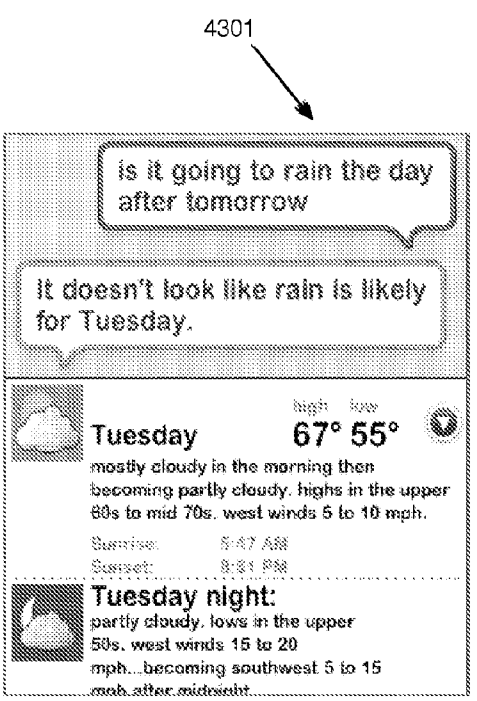


FIG. 43A

FIG. 43B

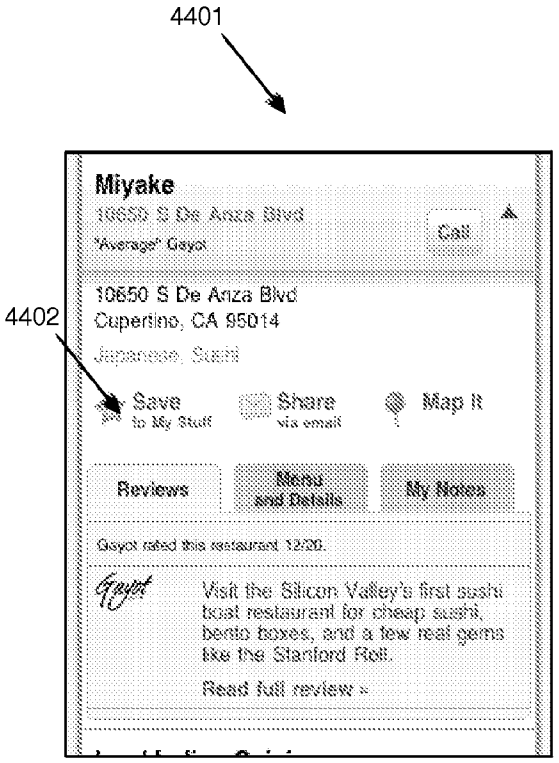


FIG. 44A

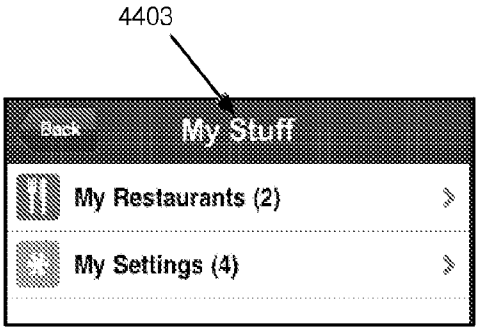


FIG. 44B

FIG. 44C

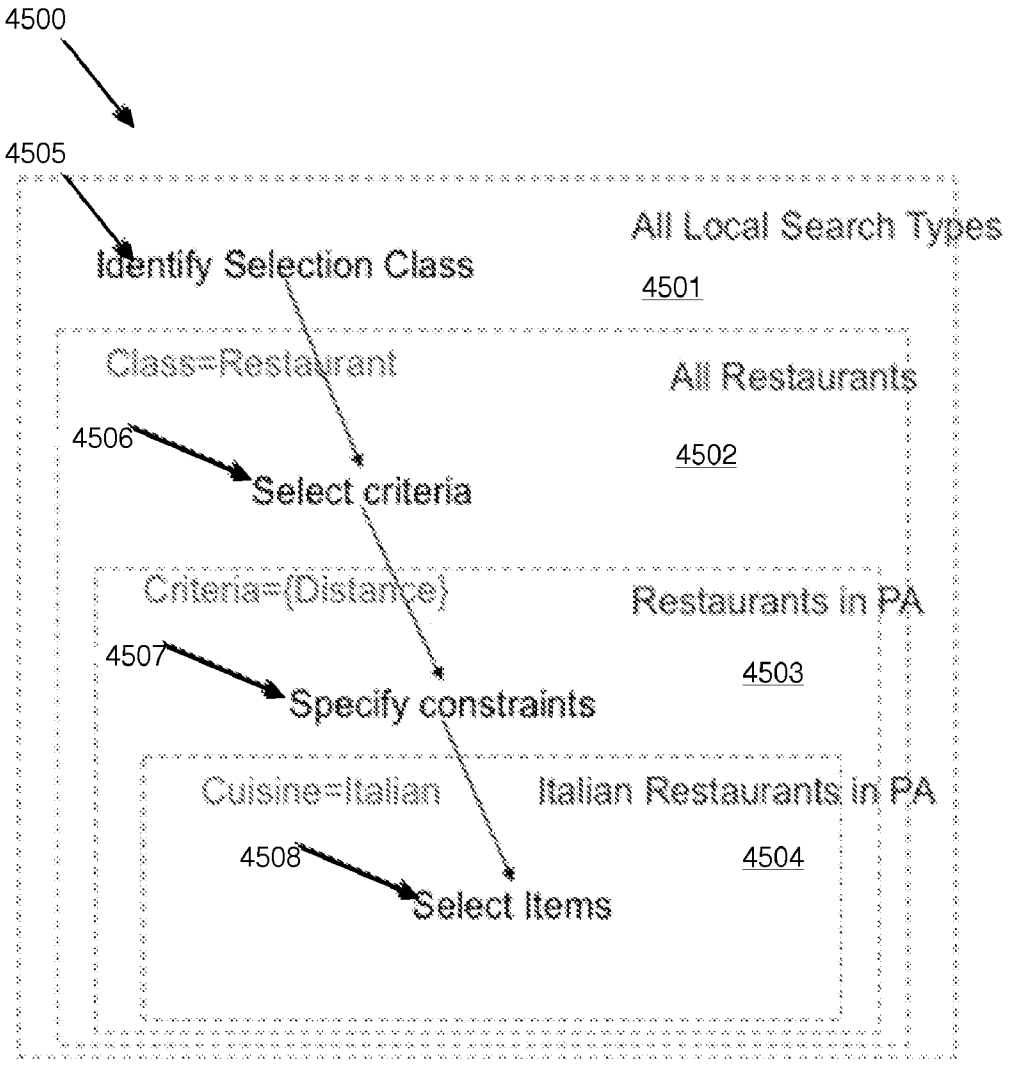

4500
4505
All Local Search Types
Identify Selection Class
4501
Class=Restaurant
All Restaurants
4506
4502
Select criteria
Criteria={Distance}
Restaurants in PA
4507
4503
Specify constraints
Cuisine=Italian
Italian Restaurants in PA
4508
4504
Select Items

FIG. 45

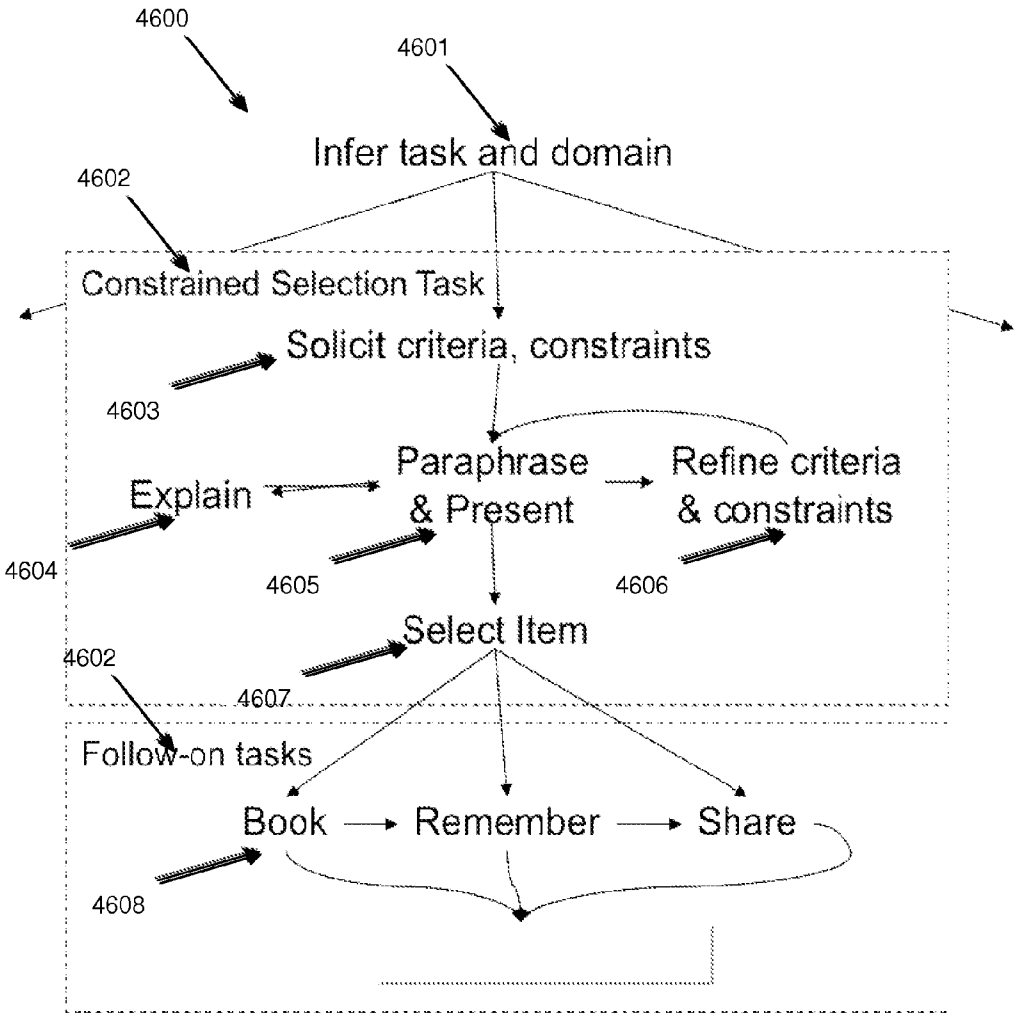

4600
4601
Infer task and domain
4602
Constrained Selection Task
Solicit criteria, constraints
4603
Explain
Paraphrase & Present
Refine criteria & constraints
4604
4605
4606
Select Item
4602
4607
Follow-on tasks
Book
Remember
Share
4608

FIG. 46

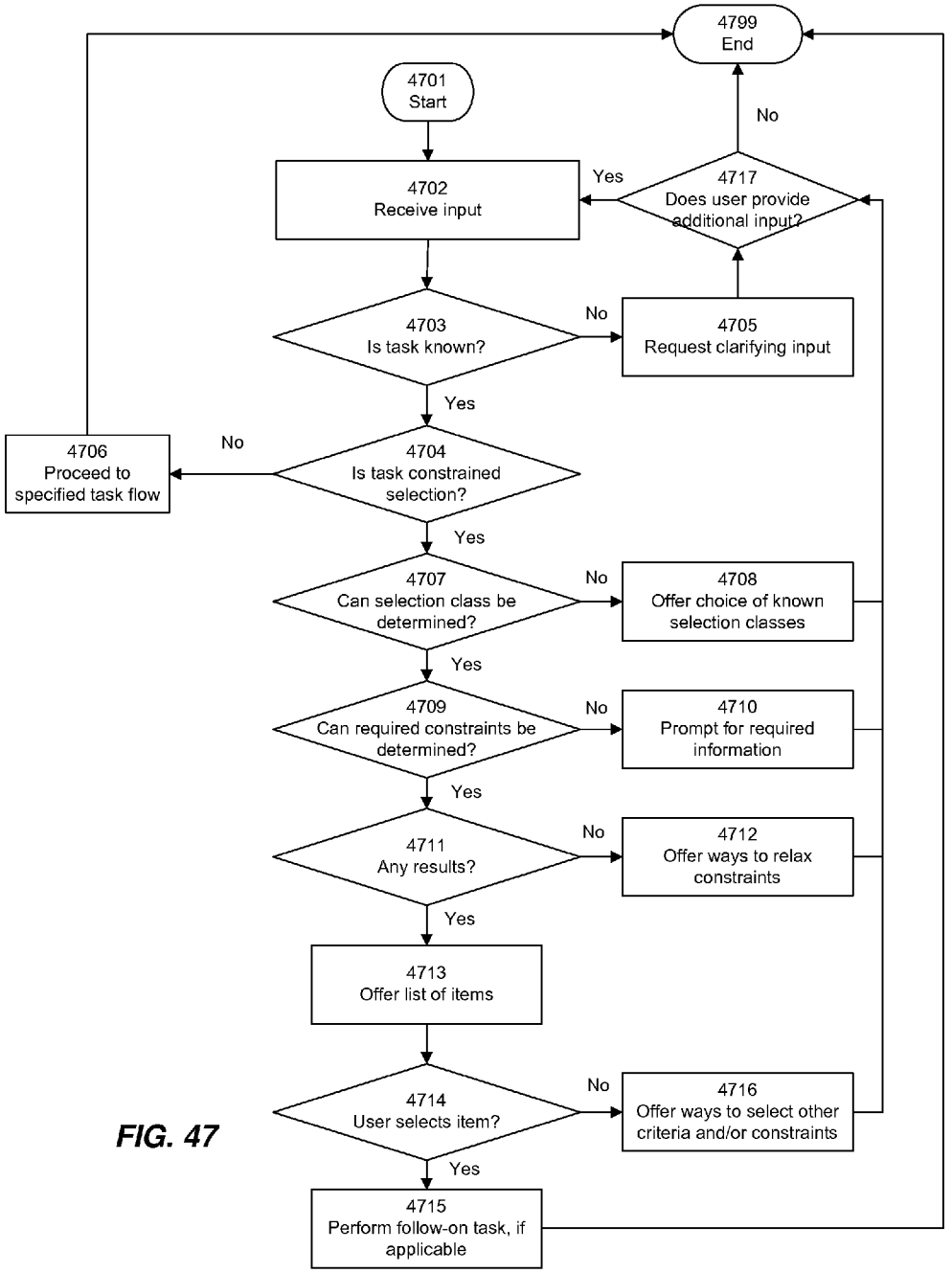

4799
End
4701
Start
No
4702
Receive input
Yes
4717
Does user provide additional input?
4703
Is task known?
No
4705
Request clarifying input
Yes
4706
Proceed to specified task flow
No
4704
Is task constrained selection?
Yes
4707
Can selection class be determined?
No
4708
Offer choice of known selection classes
Yes
4709
Can required constraints be determined?
No
4710
Prompt for required information
Yes
4711
Any results?
No
4712
Offer ways to relax constraints
Yes
4713
Offer list of items
4714
User selects item?
No
4716
Offer ways to select other criteria and/or constraints
Yes
4715
Perform follow-on task, if applicable

FIG. 47

INTELLIGENT AUTOMATED ASSISTANT

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. Provisional Patent Application Ser. No. 61/295,774 for "Intelligent Automated Assistant", attorney docket number SIRIP003P, filed Jan. 18, 2010, which is incorporated herein by reference.

[0002] This application is further related to U.S. patent application Ser. No. 11/518,292 for "Method and Apparatus for Building an Intelligent Automated Assistant", filed Sep. 8, 2006, which is incorporated herein by reference.

[0003] This application is further related to U.S. Provisional Patent Application Ser. No. 61/186,414 for "System and Method for Semantic Auto-Completion", filed Jun. 12, 2009, which is incorporated herein by reference.

FIELD OF THE INVENTION

[0004] The present invention relates to intelligent systems, and more specifically for classes of applications for intelligent automated assistants.

BACKGROUND OF THE INVENTION

[0005] Today's electronic devices are able to access a large, growing, and diverse quantity of functions, services, and information, both via the Internet and from other sources. Functionality for such devices is increasing rapidly, as many consumer devices, smartphones, tablet computers, and the like, are able to run software applications to perform various tasks and provide different types of information. Often, each application, function, website, or feature has its own user interface and its own operational paradigms, many of which can be burdensome to learn or overwhelming for users. In addition, many users may have difficulty even discovering what functionality and/or information is available on their electronic devices or on various websites; thus, such users may become frustrated or overwhelmed, or may simply be unable to use the resources available to them in an effective manner.

[0006] In particular, novice users, or individuals who are impaired or disabled in some manner, and/or are elderly, busy, distracted, and/or operating a vehicle may have difficulty interfacing with their electronic devices effectively, and/or engaging online services effectively. Such users are particularly likely to have difficulty with the large number of diverse and inconsistent functions, applications, and websites that may be available for their use.

[0007] Accordingly, existing systems are often difficult to use and to navigate, and often present users with inconsistent and overwhelming interfaces that often prevent the users from making effective use of the technology.

SUMMARY

[0008] According to various embodiments of the present invention, an intelligent automated assistant is implemented on an electronic device, to facilitate user interaction with a device, and to help the user more effectively engage with local and/or remote services. In various embodiments, the intelligent automated assistant engages with the user in an integrated, conversational manner using natural language dialog, and invokes external services when appropriate to obtain information or perform various actions.

[0009] According to various embodiments of the present invention, the intelligent automated assistant integrates a variety of capabilities provided by different software components (e.g., for supporting natural language recognition and dialog, multimodal input, personal information management, task flow management, orchestrating distributed services, and the like). Furthermore, to offer intelligent interfaces and useful functionality to users, the intelligent automated assistant of the present invention may, in at least some embodiments, coordinate these components and services. The conversation interface, and the ability to obtain information and perform follow-on task, are implemented, in at least some embodiments, by coordinating various components such as language components, dialog components, task management components, information management components and/or a plurality of external services.

[0010] According to various embodiments of the present invention, intelligent automated assistant systems may be configured, designed, and/or operable to provide various different types of operations, functionalities, and/or features, and/or to combine a plurality of features, operations, and applications of an electronic device on which it is installed. In some embodiments, the intelligent automated assistant systems of the present invention can perform any or all of: actively eliciting input from a user, interpreting user intent, disambiguating among competing interpretations, requesting and receiving clarifying information as needed, and performing (or initiating) actions based on the discerned intent. Actions can be performed, for example, by activating and/or interfacing with any applications or services that may be available on an electronic device, as well as services that are available over an electronic network such as the Internet. In various embodiments, such activation of external services can be performed via APIs or by any other suitable mechanism. In this manner, the intelligent automated assistant systems of various embodiments of the present invention can unify, simplify, and improve the user's experience with respect to many different applications and functions of an electronic device, and with respect to services that may be available over the Internet. The user can thereby be relieved of the burden of learning what functionality may be available on the device and on web-connected services, how to interface with such services to get what he or she wants, and how to interpret the output received from such services; rather, the assistant of the present invention can act as a go-between between the user and such diverse services.

[0011] In addition, in various embodiments, the assistant of the present invention provides a conversational interface that the user may find more intuitive and less burdensome than conventional graphical user interfaces. The user can engage in a form of conversational dialog with the assistant using any of a number of available input and output mechanisms, such as for example speech, graphical user interfaces (buttons and links), text entry, and the like. The system can be implemented using any of a number of different platforms, such as device APIs, the web, email, and the like, or any combination thereof. Requests for additional input can be presented to the user in the context of such a conversation. Short and long term memory can be engaged so that user input can be interpreted in proper context given previous events and communications within a given session, as well as historical and profile information about the user.

[0012] In addition, in various embodiments, context information derived from user interaction with a feature, opera-

tion, or application on a device can be used to streamline the operation of other features, operations, or applications on the device or on other devices. For example, the intelligent automated assistant can use the context of a phone call (such as the person called) to streamline the initiation of a text message (for example to determine that the text message should be sent to the same person, without the user having to explicitly specify the recipient of the text message). The intelligent automated assistant of the present invention can thereby interpret instructions such as "send him a text message", wherein the "him" is interpreted according to context information derived from a current phone call, and/or from any feature, operation, or application on the device. In various embodiments, the intelligent automated assistant takes into account various types of available context data to determine which address book contact to use, which contact data to use, which telephone number to use for the contact, and the like, so that the user need not re-specify such information manually.

[0013] In various embodiments, the assistant can also take into account external events and respond accordingly, for example, to initiate action, initiate communication with the user, provide alerts, and/or modify previously initiated action in view of the external events. If input is required from the user, a conversational interface can again be used.

[0014] In one embodiment, the system is based on sets of interrelated domains and tasks, and employs additional functionally powered by external services with which the system can interact. In various embodiments, these external services include web-enabled services, as well as functionality related to the hardware device itself. For example, in an embodiment where the intelligent automated assistant is implemented on a smartphone, personal digital assistant, tablet computer, or other device, the assistant can control many operations and functions of the device, such as to dial a telephone number, send a text message, set reminders, add events to a calendar, and the like.

[0015] In various embodiments, the system of the present invention can be implemented to provide assistance in any of a number of different domains. Examples include:

[0016] Local Services (including location- and time-specific services such as restaurants, movies, automated teller machines (ATMs), events, and places to meet);

[0017] Personal and Social Memory Services (including action items, notes, calendar events, shared links, and the like);

[0018] E-commerce (including online purchases of items such as books, DVDs, music, and the like);

[0019] Travel Services (including flights, hotels, attractions, and the like).

[0020] One skilled in the art will recognize that the above list of domains is merely exemplary. In addition, the system of the present invention can be implemented in any combination of domains.

[0021] In various embodiments, the intelligent automated assistant systems disclosed herein may be configured or designed to include functionality for automating the application of data and services available over the Internet to discover, find, choose among, purchase, reserve, or order products and services. In addition to automating the process of using these data and services, at least one intelligent automated assistant system embodiment disclosed herein may also enable the combined use of several sources of data and services at once. For example, it may combine information about products from several review sites, check prices and availability from multiple distributors, and check their locations and time constraints, and help a user find a personalized solution to their problem. Additionally, at least one intelligent automated assistant system embodiment disclosed herein may be configured or designed to include functionality for automating the use of data and services available over the Internet to discover, investigate, select among, reserve, and otherwise learn about things to do (including but not limited to movies, events, performances, exhibits, shows and attractions); places to go (including but not limited to travel destinations, hotels and other places to stay, landmarks and other sites of interest, etc.); places to eat or drink (such as restaurants and bars), times and places to meet others, and any other source of entertainment or social interaction which may be found on the Internet. Additionally, at least one intelligent automated assistant system embodiment disclosed herein may be configured or designed to include functionality for enabling the operation of applications and services via natural language dialog that may be otherwise provided by dedicated applications with graphical user interfaces including search (including location-based search); navigation (maps and directions); database lookup (such as finding businesses or people by name or other properties); getting weather conditions and forecasts, checking the price of market items or status of financial transactions; monitoring traffic or the status of flights; accessing and updating calendars and schedules; managing reminders, alerts, tasks and projects; communicating over email or other messaging platforms; and operating devices locally or remotely (e.g., dialing telephones, controlling light and temperature, controlling home security devices, playing music or video, etc.). Further, at least one intelligent automated assistant system embodiment disclosed herein may be configured or designed to include functionality for identifying, generating, and/or providing personalized recommendations for activities, products, services, source of entertainment, time management, or any other kind of recommendation service that benefits from an interactive dialog in natural language and automated access to data and services.

[0022] In various embodiments, the intelligent automated assistant of the present invention can control many features and operations of an electronic device. For example, the intelligent automated assistant can call services that interface with functionality and applications on a device via APIs or by other means, to perform functions and operations that might otherwise be initiated using a conventional user interface on the device. Such functions and operations may include, for example, setting an alarm, making a telephone call, sending a text message or email message, adding a calendar event, and the like. Such functions operations may be performed as add-on functions in the context of a conversational dialog between a user and the assistant. Such functions and operations can be specified by the user in the context of such a dialog, or they may be automatically performed based on the context of the dialog. One skilled in the art will recognize that the assistant can thereby be used as a control mechanism for initiating and controlling various operations on the electronic device, which may be used as an alternative to conventional mechanisms such as buttons or graphical user interfaces.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] The accompanying drawings illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention according

to the embodiments. One skilled in the art will recognize that the particular embodiments illustrated in the drawings are merely exemplary, and are not intended to limit the scope of the present invention.

[0024] FIG. **1** is a block diagram depicting an example of one embodiment of an intelligent automated assistant system.

[0025] FIG. **2** illustrates an example of an interaction between a user and an intelligent automated assistant according to at least one embodiment.

[0026] FIG. **3** is a block diagram depicting a computing device suitable for implementing at least a portion of an intelligent automated assistant according to at least one embodiment.

[0027] FIG. **4** is a block diagram depicting an architecture for implementing at least a portion of an intelligent automated assistant on a standalone computing system, according to at least one embodiment.

[0028] FIG. **5** is a block diagram depicting an architecture for implementing at least a portion of an intelligent automated assistant on a distributed computing network, according to at least one embodiment.

[0029] FIG. **6** is a block diagram depicting a system architecture illustrating several different types of clients and modes of operation.

[0030] FIG. **7** is a block diagram depicting a client and a server, which communicate with each other to implement the present invention according to one embodiment.

[0031] FIG. **8** is a block diagram depicting a fragment of an active ontology according to one embodiment.

[0032] FIG. **9** is a block diagram depicting an example of an alternative embodiment of an intelligent automated assistant system.

[0033] FIG. **10** is a flow diagram depicting a method of operation for active input elicitation component(s) according to one embodiment.

[0034] FIG. **11** is a flow diagram depicting a method for active typed-input elicitation according to one embodiment.

[0035] FIGS. **12** to **21** are screen shots illustrating some portions of some of the procedures for active typed-input elicitation according to one embodiment.

[0036] FIG. **22** is a flow diagram depicting a method for active input elicitation for voice or speech input according to one embodiment.

[0037] FIG. **23** is a flow diagram depicting a method for active input elicitation for GUI-based input according to one embodiment.

[0038] FIG. **24** is a flow diagram depicting a method for active input elicitation at the level of a dialog flow according to one embodiment.

[0039] FIG. **25** is a flow diagram depicting a method for active monitoring for relevant events according to one embodiment.

[0040] FIG. **26** is a flow diagram depicting a method for multimodal active input elicitation according to one embodiment.

[0041] FIG. **27** is a set of screen shots illustrating an example of various types of functions, operations, actions, and/or other features which may be provided by domain models component(s) and services orchestration according to one embodiment.

[0042] FIG. **28** is a flow diagram depicting an example of a method for natural language processing according to one embodiment.

[0043] FIG. **29** is a screen shot illustrating natural language processing according to one embodiment.

[0044] FIGS. **30** and **31** are screen shots illustrating an example of various types of functions, operations, actions, and/or other features which may be provided by dialog flow processor component(s) according to one embodiment.

[0045] FIG. **32** is a flow diagram depicting a method of operation for dialog flow processor component(s) according to one embodiment.

[0046] FIG. **33** is a flow diagram depicting an automatic call and response procedure, according to one embodiment.

[0047] FIG. **34** is a flow diagram depicting an example of task flow for a constrained selection task according to one embodiment.

[0048] FIGS. **35** and **36** are screen shots illustrating an example of the operation of constrained selection task according to one embodiment.

[0049] FIG. **37** is a flow diagram depicting an example of a procedure for executing a service orchestration procedure according to one embodiment.

[0050] FIG. **38** is a flow diagram depicting an example of a service invocation procedure according to one embodiment.

[0051] FIG. **39** is a flow diagram depicting an example of a multiphase output procedure according to one embodiment.

[0052] FIGS. **40** and **41** are screen shots depicting examples of output processing according to one embodiment.

[0053] FIG. **42** is a flow diagram depicting an example of multimodal output processing according to one embodiment.

[0054] FIGS. **43**A and **43**B are screen shots depicting an example of the use of short term personal memory component (s) to maintain dialog context while changing location, according to one embodiment.

[0055] FIGS. **44**A through **44**C are screen shots depicting an example of the use of long term personal memory component(s), according to one embodiment.

[0056] FIG. **45** depicts an example of an abstract model for a constrained selection task.

[0057] FIG. **46** depicts an example of a dialog flow model to help guide the user through a search process.

[0058] FIG. **47** is a flow diagram depicting a method of constrained selection according to one embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0059] Various techniques will now be described in detail with reference to a few example embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of one or more aspects and/or features described or reference herein. It will be apparent, however, to one skilled in the art, that one or more aspects and/or features described or reference herein may be practiced without some or all of these specific details. In other instances, well known process steps and/or structures have not been described in detail in order to not obscure some of the aspects and/or features described or reference herein.

[0060] One or more different inventions may be described in the present application. Further, for one or more of the invention(s) described herein, numerous embodiments may be described in this patent application, and are presented for illustrative purposes only. The described embodiments are not intended to be limiting in any sense. One or more of the invention(s) may be widely applicable to numerous embodiments, as is readily apparent from the disclosure. These

embodiments are described in sufficient detail to enable those skilled in the art to practice one or more of the invention(s), and it is to be understood that other embodiments may be utilized and that structural, logical, software, electrical and other changes may be made without departing from the scope of the one or more of the invention(s). Accordingly, those skilled in the art will recognize that the one or more of the invention(s) may be practiced with various modifications and alterations. Particular features of one or more of the invention (s) may be described with reference to one or more particular embodiments or figures that form a part of the present disclosure, and in which are shown, by way of illustration, specific embodiments of one or more of the invention(s). It should be understood, however, that such features are not limited to usage in the one or more particular embodiments or figures with reference to which they are described. The present disclosure is neither a literal description of all embodiments of one or more of the invention(s) nor a listing of features of one or more of the invention(s) that must be present in all embodiments.

[0061] Headings of sections provided in this patent application and the title of this patent application are for convenience only, and are not to be taken as limiting the disclosure in any way.

[0062] Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more intermediaries.

[0063] A description of an embodiment with several components in communication with each other does not imply that all such components are required. To the contrary, a variety of optional components are described to illustrate the wide variety of possible embodiments of one or more of the invention(s).

[0064] Further, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described in this patent application does not, in and of itself, indicate a requirement that the steps be performed in that order. The steps of described processes may be performed in any order practical. Further, some steps may be performed simultaneously despite being described or implied as occurring non-simultaneously (e.g., because one step is described after the other step). Moreover, the illustration of a process by its depiction in a drawing does not imply that the illustrated process is exclusive of other variations and modifications thereto, does not imply that the illustrated process or any of its steps are necessary to one or more of the invention(s), and does not imply that the illustrated process is preferred.

[0065] When a single device or article is described, it will be readily apparent that more than one device/article (whether or not they cooperate) may be used in place of a single device/article. Similarly, where more than one device or article is described (whether or not they cooperate), it will be readily apparent that a single device/article may be used in place of the more than one device or article.

[0066] The functionality and/or the features of a device may be alternatively embodied by one or more other devices that are not explicitly described as having such functionality/features. Thus, other embodiments of one or more of the invention(s) need not include the device itself.

[0067] Techniques and mechanisms described or reference herein will sometimes be described in singular form for clarity. However, it should be noted that particular embodiments include multiple iterations of a technique or multiple instantiations of a mechanism unless noted otherwise.

[0068] Although described within the context of intelligent automated assistant technology, it may be understood that the various aspects and techniques described herein (such as those associated with active ontologies, for example) may also be deployed and/or applied in other fields of technology involving human and/or computerized interaction with software.

[0069] Other aspects relating to intelligent automated assistant technology (e.g., which may be utilized by, provided by, and/or implemented at one or more intelligent automated assistant system embodiments described herein) are disclosed in one or more of the following references:

[0070] U.S. Provisional Patent Application Ser. No. 61/295,774 for "Intelligent Automated Assistant", attorney docket number SIRIP003P, filed Jan. 18, 2010, the disclosure of which is incorporated herein by reference;

[0071] U.S. patent application Ser. No. 11/518,292 for "Method And Apparatus for Building an Intelligent Automated Assistant", filed Sep. 8, 2006, the disclosure of which is incorporated herein by reference; and

[0072] U.S. Provisional Patent Application Ser. No. 61/186,414 for "System and Method for Semantic Auto-Completion", filed Jun. 12, 2009, the disclosure of which is incorporated herein by reference.

Hardware Architecture

[0073] Generally, the intelligent automated assistant techniques disclosed herein may be implemented on hardware or a combination of software and hardware. For example, they may be implemented in an operating system kernel, in a separate user process, in a library package bound into network applications, on a specially constructed machine, or on a network interface card. In a specific embodiment, the techniques disclosed herein may be implemented in software such as an operating system or in an application running on an operating system.

[0074] Software/hardware hybrid implementation(s) of at least some of the intelligent automated assistant embodiment (s) disclosed herein may be implemented on a programmable machine selectively activated or reconfigured by a computer program stored in memory. Such network devices may have multiple network interfaces which may be configured or designed to utilize different types of network communication protocols. A general architecture for some of these machines may appear from the descriptions disclosed herein. According to specific embodiments, at least some of the features and/or functionalities of the various intelligent automated assistant embodiments disclosed herein may be implemented on one or more general-purpose network host machines such as an end-user computer system, computer, network server or server system, mobile computing device (e.g., personal digital assistant, mobile phone, smartphone, laptop, tablet computer, or the like), consumer electronic device, music player, or any other suitable electronic device, router, switch, or the like, or any combination thereof. In at least some embodiments, at least some of the features and/or functionalities of the various intelligent automated assistant embodiments dis-

closed herein may be implemented in one or more virtualized computing environments (e.g., network computing clouds, or the like).

[0075] Referring now to FIG. **3**, there is shown a block diagram depicting a computing device **60** suitable for implementing at least a portion of the intelligent automated assistant features and/or functionalities disclosed herein. Computing device **60** may be, for example, an end-user computer system, network server or server system, mobile computing device (e.g., personal digital assistant, mobile phone, smartphone, laptop, tablet computer, or the like), consumer electronic device, music player, or any other suitable electronic device, or any combination or portion thereof. Computing device **60** may be adapted to communicate with other computing devices, such as clients and/or servers, over a communications network such as the Internet, using known protocols for such communication, whether wireless or wired.

[0076] In one embodiment, computing device **60** includes central processing unit (CPU) **62**, interfaces **68**, and a bus **67** (such as a peripheral component interconnect (PCI) bus). When acting under the control of appropriate software or firmware, CPU **62** may be responsible for implementing specific functions associated with the functions of a specifically configured computing device or machine. For example, in at least one embodiment, a user's personal digital assistant (PDA) may be configured or designed to function as an intelligent automated assistant system utilizing CPU **62**, memory **61**, **65**, and interface(s) **68**. In at least one embodiment, the CPU **62** may be caused to perform one or more of the different types of intelligent automated assistant functions and/or operations under the control of software modules/components, which for example, may include an operating system and any appropriate applications software, drivers, and the like.

[0077] CPU **62** may include one or more processor(s) **63** such as, for example, a processor from the Motorola or Intel family of microprocessors or the MIPS family of microprocessors. In some embodiments, processor(s) **63** may include specially designed hardware (e.g., application-specific integrated circuits (AS-ICs), electrically erasable programmable read-only memories (EEPROMs), field-programmable gate arrays (FPGAs), and the like) for controlling the operations of computing device **60**. In a specific embodiment, a memory **61** (such as non-volatile random access memory (RAM) and/or read-only memory (ROM)) also forms part of CPU **62**. However, there are many different ways in which memory may be coupled to the system. Memory block **61** may be used for a variety of purposes such as, for example, caching and/or storing data, programming instructions, and the like.

[0078] As used herein, the term "processor" is not limited merely to those integrated circuits referred to in the art as a processor, but broadly refers to a microcontroller, a microcomputer, a programmable logic controller, an application-specific integrated circuit, and any other programmable circuit.

[0079] In one embodiment, interfaces **68** are provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receiving of data packets over a computing network and sometimes support other peripherals used with computing device **60**. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various types of interfaces may be provided such as, for example, universal serial bus (USB), Serial, Ethernet, Firewire, PCI, parallel, radio frequency (RF), Bluetooth™, near-field communications (e.g., using near-field magnetics), 802.11 (WiFi), frame relay, TCP/IP, ISDN, fast Ethernet interfaces, Gigabit Ethernet interfaces, asynchronous transfer mode (ATM) interfaces, high-speed serial interface (HSSI) interfaces, Point of Sale (POS) interfaces, fiber data distributed interfaces (FDDIs), and the like. Generally, such interfaces **68** may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile and/or non-volatile memory (e.g., RAM).

[0080] Although the system shown in FIG. **3** illustrates one specific architecture for a computing device **60** for implementing the techniques of the invention described herein, it is by no means the only device architecture on which at least a portion of the features and techniques described herein may be implemented. For example, architectures having one or any number of processors **63** can be used, and such processors **63** can be present in a single device or distributed among any number of devices. In one embodiment, a single processor **63** handles communications as well as routing computations. In various embodiments, different types of intelligent automated assistant features and/or functionalities may be implemented in an intelligent automated assistant system which includes a client device (such as a personal digital assistant or smartphone running client software) and server system(s) (such as a server system described in more detail below).

[0081] Regardless of network device configuration, the system of the present invention may employ one or more memories or memory modules (such as, for example, memory block **65**) configured to store data, program instructions for the general-purpose network operations and/or other information relating to the functionality of the intelligent automated assistant techniques described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example. The memory or memories may also be configured to store data structures, keyword taxonomy information, advertisement information, user click and impression information, and/or other specific non-program information described herein.

[0082] Because such information and program instructions may be employed to implement the systems/methods described herein, at least some network device embodiments may include nontransitory machine-readable storage media, which, for example, may be configured or designed to store program instructions, state information, and the like for performing various operations described herein. Examples of such nontransitory machine-readable storage media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as floptical disks, and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM), flash memory, memristor memory, random access memory (RAM), and the like. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

[0083] In one embodiment, the system of the present invention is implemented on a standalone computing system. Referring now to FIG. **4**, there is shown a block diagram depicting an architecture for implementing at least a portion of an intelligent automated assistant on a standalone comput-

ing system, according to at least one embodiment. Computing device **60** includes processor(s) **63** which run software for implementing intelligent automated assistant **1002**. Input device **1206** can be of any type suitable for receiving user input, including for example a keyboard, touchscreen, microphone (for example, for voice input), mouse, touchpad, trackball, five-way switch, joystick, and/or any combination thereof. Output device **1207** can be a screen, speaker, printer, and/or any combination thereof. Memory **1210** can be random-access memory having a structure and architecture as are known in the art, for use by processor(s) **63** in the course of running software. Storage device **1208** can be any magnetic, optical, and/or electrical storage device for storage of data in digital form; examples include flash memory, magnetic hard drive, CD-ROM, and/or the like.

[0084] In another embodiment, the system of the present invention is implemented on a distributed computing network, such as one having any number of clients and/or servers. Referring now to FIG. **5**, there is shown a block diagram depicting an architecture for implementing at least a portion of an intelligent automated assistant on a distributed computing network, according to at least one embodiment.

[0085] In the arrangement shown in FIG. **5**, any number of clients **1304** are provided; each client **1304** may run software for implementing client-side portions of the present invention. In addition, any number of servers **1340** can be provided for handling requests received from clients **1304**. Clients **1304** and servers **1340** can communicate with one another via electronic network **1361**, such as the Internet. Network **1361** may be implemented using any known network protocols, including for example wired and/or wireless protocols.

[0086] In addition, in one embodiment, servers **1340** can call external services **1360** when needed to obtain additional information or refer to store data concerning previous interactions with particular users. Communications with external services **1360** can take place, for example, via network **1361**. In various embodiments, external services **1360** include web-enabled services and/or functionality related to or installed on the hardware device itself. For example, in an embodiment where assistant **1002** is implemented on a smartphone or other electronic device, assistant **1002** can obtain information stored in a calendar application ("app"), contacts, and/or other sources.

[0087] In various embodiments, assistant **1002** can control many features and operations of an electronic device on which it is installed. For example, assistant **1002** can call external services **1360** that interface with functionality and applications on a device via APIs or by other means, to perform functions and operations that might otherwise be initiated using a conventional user interface on the device. Such functions and operations may include, for example, setting an alarm, making a telephone call, sending a text message or email message, adding a calendar event, and the like. Such functions and operations may be performed as add-on functions in the context of a conversational dialog between a user and assistant **1002**. Such functions and operations can be specified by the user in the context of such a dialog, or they may be automatically performed based on the context of the dialog. One skilled in the art will recognize that assistant **1002** can thereby be used as a control mechanism for initiating and controlling various operations on the electronic device, which may be used as an alternative to conventional mechanisms such as buttons or graphical user interfaces.

[0088] For example, the user may provide input to assistant **1002** such as "I need to wake tomorrow at 8 am". Once assistant **1002** has determined the user's intent, using the techniques described herein, assistant **1002** can call external services **1340** to interface with an alarm clock function or application on the device. Assistant **1002** sets the alarm on behalf of the user. In this manner, the user can use assistant **1002** as a replacement for conventional mechanisms for setting the alarm or performing other functions on the device. If the user's requests are ambiguous or need further clarification, assistant **1002** can use the various techniques described herein, including active elicitation, paraphrasing, suggestions, and the like, to obtain the needed information so that the correct services **1340** are called and the intended action taken. In one embodiment, assistant **1002** may prompt the user for confirmation before calling a service **1340** to perform a function. In one embodiment, a user can selectively disable assistant's **1002** ability to call particular services **1340**, or can disable all such service-calling if desired.

[0089] The system of the present invention can be implemented with many different types of clients **1304** and modes of operation. Referring now to FIG. **6**, there is shown a block diagram depicting a system architecture illustrating several different types of clients **1304** and modes of operation. One skilled in the art will recognize that the various types of clients **1304** and modes of operation shown in FIG. **6** are merely exemplary, and that the system of the present invention can be implemented using clients **1304** and/or modes of operation other than those depicted. Additionally, the system can include any or all of such clients **1304** and/or modes of operation, alone or in any combination. Depicted examples include:

[0090] Computer devices with input/output devices and/or sensors **1402**. A client component may be deployed on any such computer device **1402**. At least one embodiment may be implemented using a web browser **1304**A or other software application for enabling communication with servers **1340** via network **1361**. Input and output channels may of any type, including for example visual and/or auditory channels. For example, in one embodiment, the system of the invention can be implemented using voice-based communication methods, allowing for an embodiment of the assistant for the blind whose equivalent of a web browser is driven by speech and uses speech for output.

[0091] Mobile Devices with I/O and sensors **1406**, for which the client may be implemented as an application on the mobile device **1304**B. This includes, but is not limited to, mobile phones, smartphones, personal digital assistants, tablet devices, networked game consoles, and the like.

[0092] Consumer Appliances with I/O and sensors **1410**, for which the client may be implemented as an embedded application on the appliance **1304**C.

[0093] Automobiles and other vehicles with dashboard interfaces and sensors **1414**, for which the client may be implemented as an embedded system application **1304**D. This includes, but is not limited to, car navigation systems, voice control systems, in-car entertainment systems, and the like.

[0094] Networked computing devices such as routers **1418** or any other device that resides on or interfaces with a network, for which the client may be implemented as a device-resident application **1304**E.

[0095] Email clients **1424**, for which an embodiment of the assistant is connected via an Email Modality Server **1426**. Email Modality server **1426** acts as a communication bridge, for example taking input from the user as email messages sent to the assistant and sending output from the assistant to the user as replies.

[0096] Instant messaging clients **1428**, for which an embodiment of the assistant is connected via a Messaging Modality Server **1430**. Messaging Modality server **1430** acts as a communication bridge, taking input from the user as messages sent to the assistant and sending output from the assistant to the user as messages in reply.

[0097] Voice telephones **1432**, for which an embodiment of the assistant is connected via a Voice over Internet Protocol (VoIP) Modality Server **1430**. VoIP Modality server **1430** acts as a communication bridge, taking input from the user as voice spoken to the assistant and sending output from the assistant to the user, for example as synthesized speech, in reply.

[0098] For messaging platforms including but not limited to email, instant messaging, discussion forums, group chat sessions, live help or customer support sessions and the like, assistant **1002** may act as a participant in the conversations. Assistant **1002** may monitor the conversation and reply to individuals or the group using one or more the techniques and methods described herein for one-to-one interactions.

[0099] In various embodiments, functionality for implementing the techniques of the present invention can be distributed among any number of client and/or server components. For example, various software modules can be implemented for performing various functions in connection with the present invention, and such modules can be variously implemented to run on server and/or client components. Referring now to FIG. **7**, there is shown an example of a client **1304** and a server **1340**, which communicate with each other to implement the present invention according to one embodiment. FIG. **7** depicts one possible arrangement by which software modules can be distributed among client **1304** and server **1340**. One skilled in the art will recognize that the depicted arrangement is merely exemplary, and that such modules can be distributed in many different ways. In addition, any number of clients **1304** and/or servers **1340** can be provided, and the modules can be distributed among these clients **1304** and/or servers **1340** in any of a number of different ways.

[0100] In the example of FIG. **7**, input elicitation functionality and output processing functionality are distributed among client **1304** and server **1340**, with client part of input elicitation **1094***a* and client part of output processing **1092***a* located at client **1304**, and server part of input elicitation **1094***b* and server part of output processing **1092***b* located at server **1340**. The following components are located at server **1340**:

[0101] complete vocabulary **1058***b;*

[0102] complete library of language pattern recognizers **1060***b;*

[0103] master version of short term personal memory **1052***b;*

[0104] master version of long term personal memory **1054***b.*

[0105] In one embodiment, client **1304** maintains subsets and/or portions of these components locally, to improve responsiveness and reduce dependence on network communications. Such subsets and/or portions can be maintained and updated according to well known cache management techniques. Such subsets and/or portions include, for example:

[0106] subset of vocabulary **1058***a;*

[0107] subset of library of language pattern recognizers **1060***a;*

[0108] cache of short term personal memory **1052***a;*

[0109] cache of long term personal memory **1054***a.*

[0110] Additional components may be implemented as part of server **1340**, including for example:

[0111] language interpreter **1070**;

[0112] dialog flow processor **1080**;

[0113] output processor **1090**;

[0114] domain entity databases **1072**;

[0115] task flow models **1086**;

[0116] services orchestration **1082**;

[0117] service capability models **1088**.

[0118] Each of these components will be described in more detail below. Server **1340** obtains additional information by interfacing with external services **1360** when needed.

Conceptual Architecture

[0119] Referring now to FIG. **1**, there is shown a simplified block diagram of a specific example embodiment of an intelligent automated assistant **1002**. As described in greater detail herein, different embodiments of intelligent automated assistant systems may be configured, designed, and/or operable to provide various different types of operations, functionalities, and/or features generally relating to intelligent automated assistant technology. Further, as described in greater detail herein, many of the various operations, functionalities, and/or features of the intelligent automated assistant system(s) disclosed herein may provide may enable or provide different types of advantages and/or benefits to different entities interacting with the intelligent automated assistant system(s). The embodiment shown in FIG. **1** may be implemented using any of the hardware architectures described above, or using a different type of hardware architecture.

[0120] For example, according to different embodiments, at least some intelligent automated assistant system(s) may be configured, designed, and/or operable to provide various different types of operations, functionalities, and/or features, such as, for example, one or more of the following (or combinations thereof):

[0121] automate the application of data and services available over the Internet to discover, find, choose among, purchase, reserve, or order products and services. In addition to automating the process of using these data and services, intelligent automated assistant **1002** may also enable the combined use of several sources of data and services at once. For example, it may combine information about products from several review sites, check prices and availability from multiple distributors, and check their locations and time constraints, and help a user find a personalized solution to their problem.

[0122] automate the use of data and services available over the Internet to discover, investigate, select among, reserve, and otherwise learn about things to do (including but not limited to movies, events, performances, exhibits, shows and attractions); places to go (including but not limited to travel destinations, hotels and other places to stay, landmarks and other sites of interest, and the like); places to eat or drink (such as restaurants and

bars), times and places to meet others, and any other source of entertainment or social interaction which may be found on the Internet.

[0123] enable the operation of applications and services via natural language dialog that are otherwise provided by dedicated applications with graphical user interfaces including search (including location-based search); navigation (maps and directions); database lookup (such as finding businesses or people by name or other properties); getting weather conditions and forecasts, checking the price of market items or status of financial transactions; monitoring traffic or the status of flights; accessing and updating calendars and schedules; managing reminders, alerts, tasks and projects; communicating over email or other messaging platforms; and operating devices locally or remotely (e.g., dialing telephones, controlling light and temperature, controlling home security devices, playing music or video, and the like). In one embodiment, assistant **1002** can be used to initiate, operate, and control many functions and apps available on the device.

[0124] offer personal recommendations for activities, products, services, source of entertainment, time management, or any other kind of recommendation service that benefits from an interactive dialog in natural language and automated access to data and services.

[0125] According to different embodiments, at least a portion of the various types of functions, operations, actions, and/or other features provided by intelligent automated assistant **1002** may be implemented at one or more client systems (s), at one or more server systems (s), and/or combinations thereof.

[0126] According to different embodiments, at least a portion of the various types of functions, operations, actions, and/or other features provided by assistant **1002** may implement by at least one embodiment of an automated call and response procedure, such as that illustrated and described, for example, with respect to FIG. **33**.

[0127] Additionally, various embodiments of assistant **1002** described herein may include or provide a number of different advantages and/or benefits over currently existing intelligent automated assistant technology such as, for example, one or more of the following (or combinations thereof):

[0128] The integration of speech-to-text and natural language understanding technology that is constrained by a set of explicit models of domains, tasks, services, and dialogs. Unlike assistant technology that attempts to implement a general-purpose artificial intelligence system, the embodiments described herein may apply the multiple sources of constraints to reduce the number of solutions to a more tractable size. This results in fewer ambiguous interpretations of language, fewer relevant domains or tasks, and fewer ways to operationalize the intent in services. The focus on specific domains, tasks, and dialogs also makes it feasible to achieve coverage over domains and tasks with human-managed vocabulary and mappings from intent to services parameters.

[0129] The ability to solve user problems by invoking services on their behalf over the Internet, using APIs. Unlike search engines which only return links and content, some embodiments of automated assistants **1002** described herein may automate research and problem-solving activities. The ability to invoke multiple services for a given request also provides broader functionality to the user than is achieved by visiting a single site, for instance to produce a product or service or find something to do.

[0130] The application of personal information and personal interaction history in the interpretation and execution of user requests. Unlike conventional search engines or question answering services, the embodiments described herein use information from personal interaction history (e.g., dialog history, previous selections from results, and the like), personal physical context (e.g., user's location and time), and personal information gathered in the context of interaction (e.g., name, email addresses, physical addresses, phone numbers, account numbers, preferences, and the like). Using these sources of information enables, for example,

[0131] better interpretation of user input (e.g., using personal history and physical context when interpreting language);

[0132] more personalized results (e.g., that bias toward preferences or recent selections);

[0133] improved efficiency for the user (e.g., by automating steps involving the signing up to services or filling out forms).

[0134] The use of dialog history in interpreting the natural language of user inputs. Because the embodiments may keep personal history and apply natural language understanding on user inputs, they may also use dialog context such as current location, time, domain, task step, and task parameters to interpret the new inputs. Conventional search engines and command processors interpret at least one query independent of a dialog history. The ability to use dialog history may make a more natural interaction possible, one which resembles normal human conversation.

[0135] Active input elicitation, in which assistant **1002** actively guides and constrains the input from the user, based on the same models and information used to interpret their input. For example, assistant **1002** may apply dialog models to suggest next steps in a dialog with the user in which they are refining a request; offer completions to partially typed input based on domain and context specific possibilities; or use semantic interpretation to select from among ambiguous interpretations of speech as text or text as intent.

[0136] The explicit modeling and dynamic management of services, with dynamic and robust services orchestration. The architecture of embodiments described enables assistant **1002** to interface with many external services, dynamically determine which services may provide information for a specific user request, map parameters of the user request to different service APIs, call multiple services at once, integrate results from multiple services, fail over gracefully on failed services, and/or efficiently maintain the implementation of services as their APIs and capabilities evolve.

[0137] The use of active ontologies as a method and apparatus for building assistants **1002**, which simplifies the software engineering and data maintenance of automated assistant systems. Active ontologies are an integration of data modeling and execution environments for assistants. They provide a framework to tie together the various sources of models and data (domain concepts, task flows, vocabulary, language pattern recogniz-

ers, dialog context, user personal information, and mappings from domain and task requests to external services. Active ontologies and the other architectural innovations described herein make it practical to build deep functionality within domains, unifying multiple sources of information and services, and to do this across a set of domains.

[0138] In at least one embodiment, intelligent automated assistant **1002** may be operable to utilize and/or generate various different types of data and/or other types of information when performing specific tasks and/or operations. This may include, for example, input data/information and/or output data/information. For example, in at least one embodiment, intelligent automated assistant **1002** may be operable to access, process, and/or otherwise utilize information from one or more different types of sources, such as, for example, one or more local and/or remote memories, devices and/or systems. Additionally, in at least one embodiment, intelligent automated assistant **1002** may be operable to generate one or more different types of output data/information, which, for example, may be stored in memory of one or more local and/or remote devices and/or systems.

[0139] Examples of different types of input data/information which may be accessed and/or utilized by intelligent automated assistant **1002** may include, but are not limited to, one or more of the following (or combinations thereof):

[0140] Voice input: from mobile devices such as mobile telephones and tablets, computers with microphones, Bluetooth headsets, automobile voice control systems, over the telephone system, recordings on answering services, audio voicemail on integrated messaging services, consumer applications with voice input such as clock radios, telephone station, home entertainment control systems, and game consoles.

[0141] Text input from keyboards on computers or mobile devices, keypads on remote controls or other consumer electronics devices, email messages sent to the assistant, instant messages or similar short messages sent to the assistant, text received from players in multiuser game environments, and text streamed in message feeds.

[0142] Location information coming from sensors or location-based systems. Examples include Global Positioning System (GPS) and Assisted GPS (A-GPS) on mobile phones. In one embodiment, location information is combined with explicit user input. In one embodiment, the system of the present invention is able to detect when a user is at home, based on known address information and current location determination. In this manner, certain inferences may be made about the type of information the user might be interested in when at home as opposed to outside the home, as well as the type of services and actions that should be invoked on behalf of the user depending on whether or not he or she is at home.

[0143] Time information from clocks on client devices. This may include, for example, time from telephones or other client devices indicating the local time and time zone. In addition, time may be used in the context of user requests, such as for instance, to interpret phrases such as "in an hour" and "tonight".

[0144] Compass, accelerometer, gyroscope, and/or travel velocity data, as well as other sensor data from mobile or handheld devices or embedded systems such as automobile control systems. This may also include device positioning data from remote controls to appliances and game consoles.

[0145] Clicking and menu selection and other events from a graphical user interface (GUI) on any device having a GUI. Further examples include touches to a touch screen.

[0146] Events from sensors and other data-driven triggers, such as alarm clocks, calendar alerts, price change triggers, location triggers, push notification onto a device from servers, and the like.

[0147] The input to the embodiments described herein also includes the context of the user interaction history, including dialog and request history.

[0148] Examples of different types of output data/information which may be generated by intelligent automated assistant **1002** may include, but are not limited to, one or more of the following (or combinations thereof):

[0149] Text output sent directly to an output device and/or to the user interface of a device

[0150] Text and graphics sent to a user over email

[0151] Text and graphics send to a user over a messaging service

[0152] Speech output, may include one or more of the following (or combinations thereof):

[0153] Synthesized speech

[0154] Sampled speech

[0155] Recorded messages

[0156] Graphical layout of information with photos, rich text, videos, sounds, and hyperlinks. For instance, the content rendered in a web browser.

[0157] Actuator output to control physical actions on a device, such as causing it to turn on or off, make a sound, change color, vibrate, control a light, or the like.

[0158] Invoking other applications on a device, such as calling a mapping application, voice dialing a telephone, sending an email or instant message, playing media, making entries in calendars, task managers, and note applications, and other applications.

[0159] Actuator output to control physical actions to devices attached or controlled by a device, such as operating a remote camera, controlling a wheelchair, playing music on remote speakers, playing videos on remote displays, and the like.

[0160] It may be appreciated that the intelligent automated assistant **1002** of FIG. **1** is but one example from a wide range of intelligent automated assistant system embodiments which may be implemented. Other embodiments of the intelligent automated assistant system (not shown) may include additional, fewer and/or different components/features than those illustrated, for example, in the example intelligent automated assistant system embodiment of FIG. **1**.

User Interaction

[0161] Referring now to FIG. **2**, there is shown an example of an interaction between a user and at least one embodiment of an intelligent automated assistant **1002**. The example of FIG. **2** assumes that a user is speaking to intelligent automated assistant **1002** using input device **1206**, which may be a speech input mechanism, and the output is graphical layout to output device **1207**, which may be a scrollable screen. Conversation screen **101**A features a conversational user interface showing what the user said **101**B ("I'd like a romantic place for Italian food near my office") and assistant's **1002**

response, which is a summary of its findings **101**C ("OK, I found these Italian restaurants which reviews say are romantic close to your work:") and a set of results **101**D (the first three of a list of restaurants are shown). In this example, the user clicks on the first result in the list, and the result automatically opens up to reveal more information about the restaurant, shown in information screen **101**E. Information screen **101**E and conversation screen **101**A may appear on the same output device, such as a touchscreen or other display device; the examples depicted in FIG. **2** are two different output states for the same output device.

[0162] In one embodiment, information screen **101**E shows information gathered and combined from a variety of services, including for example, any or all of the following:

[0163] Addresses and geolocations of businesses;

[0164] Distance from user's current location;

[0165] Reviews from a plurality of sources;

[0166] In one embodiment, information screen **101**E also includes some examples of services that assistant **1002** might offer on behalf of the user, including:

[0167] Dial a telephone to call the business ("call");

[0168] Remember this restaurant for future reference ("save");

[0169] Send an email to someone with the directions and information about this restaurant ("share");

[0170] Show the location of and directions to this restaurant on a map ("map it");

[0171] Save personal notes about this restaurant ("my notes").

[0172] As shown in the example of FIG. **2**, in one embodiment, assistant **1002** includes intelligence beyond simple database applications, such as, for example,

[0173] Processing a statement of intent in a natural language **101**B, not just keywords;

[0174] Inferring semantic intent from that language input, such as interpreting "place for Italian food" as "Italian restaurants";

[0175] Operationalizing semantic intent into a strategy for using online services and executing that strategy on behalf of the user (e.g., operationalizing the desire for a romantic place into the strategy of checking online review sites for reviews that describe a place as "romantic").

Intelligent Automated Assistant Components

[0176] According to various embodiments, intelligent automated assistant **1002** may include a plurality of different types of components, devices, modules, processes, systems, and the like, which, for example, may be implemented and/or instantiated via the use of hardware and/or combinations of hardware and software. For example, as illustrated in the example embodiment of FIG. **1**, assistant **1002** may include one or more of the following types of systems, components, devices, processes, and the like (or combinations thereof):

[0177] One or more active ontologies **1050**;

[0178] Active input elicitation component(s) **1094** (may include client part **1094***a* and server part **1094***b*);

[0179] Short term personal memory component(s) **1052** (may include master version **1052***b* and cache **1052***a*);

[0180] Long-term personal memory component(s) **1054** (may include master version **1052***b* and cache **1052***a*);

[0181] Domain models component(s) **1056**;

[0182] Vocabulary component(s) **1058** (may include complete vocabulary **1058***b* and subset **1058***a*);

[0183] Language pattern recognizer(s) component(s) **1060** (may include full library **1060***b* and subset **1560***a*);

[0184] Language interpreter component(s) **1070**;

[0185] Domain entity database(s) **1072**;

[0186] Dialog flow processor component(s) **1080**;

[0187] Services orchestration component(s) **1082**;

[0188] Services component(s) **1084**;

[0189] Task flow models component(s) **1086**;

[0190] Dialog flow models component(s) **1087**;

[0191] Service models component(s) **1088**;

[0192] Output processor component(s) **1090**.

[0193] As described in connection with FIG. **7**, in certain client/server-based embodiments, some or all of these components may be distributed between client **1304** and server **1340**.

[0194] For purposes of illustration, at least a portion of the different types of components of a specific example embodiment of intelligent automated assistant **1002** will now be described in greater detail with reference to the example intelligent automated assistant **1002** embodiment of FIG. **1**.

Active Ontologies **1050**

[0195] Active ontologies **1050** serve as a unifying infrastructure that integrates models, components, and/or data from other parts of embodiments of intelligent automated assistants **1002**. In the field of computer and information science, ontologies provide structures for data and knowledge representation such as classes/types, relations, attributes/properties and their instantiation in instances. Ontologies are used, for example, to build models of data and knowledge. In some embodiments of the intelligent automated system **1002**, ontologies are part of the modeling framework in which to build models such as domain models.

[0196] Within the context of the present invention, an "active ontology" **1050** may also serve as an execution environment, in which distinct processing elements are arranged in an ontology-like manner (e.g., having distinct attributes and relations with other processing elements). These processing elements carry out at least some of the tasks of intelligent automated assistant **1002**. Any number of active ontologies **1050** can be provided.

[0197] In at least one embodiment, active ontologies **1050** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0198] Act as a modeling and development environment, integrating models and data from various model and data components, including but not limited to

[0199] Domain models **1056**

[0200] Vocabulary **1058**

[0201] Domain entity databases **1072**

[0202] Task flow models **1086**

[0203] Dialog flow models **1087**

[0204] Service capability models **1088**

[0205] Act as a data-modeling environment on which ontology-based editing tools may operate to develop new models, data structures, database schemata, and representations.

[0206] Act as a live execution environment, instantiating values for elements of domain **1056**, task **1086**, and/or dialog models **1087**, language pattern recognizers, and/or vocabulary **1058**, and user-specific information such as that found in short term personal memory **1052**, long

term personal memory **1054**, and/or the results of service orchestration **1182**. For example, some nodes of an active ontology may correspond to domain concepts such as restaurant and its property restaurant name. During live execution, these active ontology nodes may be instantiated with the identity of a particular restaurant entity and its name, and how its name corresponds to words in a natural language input utterance. Thus, in this embodiment, the active ontology is serving as both a modeling environment specifying the concept that restaurants are entities with identities that have names, and for storing dynamic bindings of those modeling nodes with data from entity databases and parses of natural language.

[0207] Enable the communication and coordination among components and processing elements of an intelligent automated assistant, such as, for example, one or more of the following (or combinations thereof):

[0208] Active input elicitation component(s) **1094**

[0209] Language interpreter component(s) **1070**

[0210] Dialog flow processor component(s) **1080**

[0211] Services orchestration component(s) **1082**

[0212] Services component(s) **1084**

[0213] In one embodiment, at least a portion of the functions, operations, actions, and/or other features of active ontologies **1050** described herein may be implemented, at least in part, using various methods and apparatuses described in U.S. patent application Ser. No. 11/518,292 for "Method and Apparatus for Building an Intelligent Automated Assistant", filed Sep. 8, 2006.

[0214] In at least one embodiment, a given instance of active ontology **1050** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by active ontologies **1050** may include, but are not limited to, one or more of the following (or combinations thereof):

[0215] Static data that is available from one or more components of intelligent automated assistant **1002**;

[0216] Data that is dynamically instantiated per user session, for example, but not limited to, maintaining the state of the user-specific inputs and outputs exchanged among components of intelligent automated assistant **1002**, the contents of short term personal memory, the inferences made from previous states of the user session, and the like.

[0217] In this manner, active ontologies **1050** are used to unify elements of various components in intelligent automated assistant **1002**. An active ontology **1050** allows an author, designer, or system builder to integrate components so that the elements of one component are identified with elements of other components. The author, designer, or system builder can thus combine and integrate the components more easily.

[0218] Referring now to FIG. **8**, there is shown an example of a fragment of an active ontology **1050** according to one embodiment. This example is intended to help illustrate some of the various types of functions, operations, actions, and/or other features that may be provided by active ontologies **1050**.

[0219] Active ontology **1050** in FIG. **8** includes representations of a restaurant and meal event. In this example, a restaurant is a concept **1610** with properties such as its name **1612**, cuisines served **1615**, and its location **1613**, which in turn might be modeled as a structured node with properties for street address **1614**. The concept of a meal event might be modeled as a node **1616** including a dining party **1617** (which has a size **1619**) and time period **1618**.

[0220] Active ontologies may include and/or make reference to domain models **1056**. For example, FIG. **8** depicts a dining out domain model **1622** linked to restaurant concept **1610** and meal event concept **1616**. In this instance, active ontology **1050** includes dining out domain model **1622**; specifically, at least two nodes of active ontology **1050**, namely restaurant **1610** and meal event **1616**, are also included in and/or referenced by dining out domain model **1622**. This domain model represents, among other things, the idea that dining out involves meal event that occur at restaurants. The active ontology nodes restaurant **1610** and meal event **1616** are also included and/or referenced by other components of the intelligent automated assistant, a shown by dotted lines in FIG. **8**.

[0221] Active ontologies may include and/or make reference to task flow models **1086**. For example, FIG. **8** depicts an event planning task flow model **1630**, which models the planning of events independent of domains, applied to a domain-specific kind of event: meal event **1616**. Here, active ontology **1050** includes general event planning task flow model **1630**, which comprises nodes representing events and other concepts involved in planning them. Active ontology **1050** also includes the node meal event **1616**, which is a particular kind of event. In this example, meal event **1616** is included or made reference to by both domain model **1622** and task flow model **1630**, and both of these models are included in and/or referenced by active ontology **1050**. Again, meal event **1616** is an example of how active ontologies can unify elements of various components included and/or referenced by other components of the intelligent automated assistant, a shown by dotted lines in FIG. **8**.

[0222] Active ontologies may include and/or make reference to dialog flow models **1087**. For example, FIG. **8** depicts a dialog flow model **1642** for getting the values of constraints required for a transaction instantiated on the constraint party size as represented in concept **1619**. Again, active ontology **1050** provides a framework for relating and unifying various components such as dialog flow models **1087**. In this case, dialog flow model **1642** has a general concept of a constraint that is instantiated in this particular example to the active ontology node party size **1619**. This particular dialog flow model **1642** operates at the abstraction of constraints, independent of domain. Active ontology **1050** represents party size property **1619** of party node **1617**, which is related to meal event node **1616**. In such an embodiment, intelligent automated assistant **1002** uses active ontology **1050** to unify the concept of constraint in dialog flow model **1642** with the property of party size **1619** as part of a cluster of nodes representing meal event concept **1616**, which is part of the domain model **1622** for dining out.

[0223] Active ontologies may include and/or make reference to service models **1088**. For example, FIG. **8** depicts a model of a restaurant reservation service **1672** associated with the dialog flow step for getting values required for that service to perform a transaction. In this

instance, service model **1672** for a restaurant reservation service specifies that a reservation requires a value for party size **1619** (the number of people sitting at a table to reserve). The concept party size **1619**, which is part of active ontology **1050**, also is linked or related to a general dialog flow model **1642** for asking the user about the constraints for a transaction; in this instance, the party size is a required constraint for dialog flow model **1642**.

[0224] Active ontologies may include and/or make reference to domain entity databases **1072**. For example, FIG. **8** depicts a domain entity database of restaurants **1652** associated with restaurant node **1610** in active ontology **1050**. Active ontology **1050** represents the general concept of restaurant **1610**, as may be used by the various components of intelligent automated assistant **1002**, and it is instantiated by data about specific restaurants in restaurant database **1652**.

[0225] Active ontologies may include and/or make reference to vocabulary databases **1058**. For example, FIG. **8** depicts a vocabulary database of cuisines **1662**, such as Italian, French, and the like, and the words associated with each cuisine such as "French", "continental", "provincial", and the like. Active ontology **1050** includes restaurant node **1610**, which is related to cuisines served node **1615**, which is associated with the representation of cuisines in cuisines database **1662**. A specific entry in database **1662** for a cuisine, such as "French", is thus related through active ontology **1050** as an instance of the concept of cuisines served **1615**.

[0226] Active ontologies may include and/or make reference to any database that can be mapped to concepts or other representations in ontology **1050**. Domain entity databases **1072** and vocabulary databases **1058** are merely two examples of how active ontology **1050** may integrate databases with each other and with other components of automated assistant **1002**. Active ontologies allow the author, designer, or system builder to specify a nontrivial mapping between representations in the database and representations in ontology **1050**. For example, the database schema for restaurants database **1652** may represent a restaurant as a table of strings and numbers, or as a projection from a larger database of business, or any other representation suitable for database **1652**. In this example active ontology **1050**, restaurant **1610** is a concept node with properties and relations, organized differently from the database tables. In this example, nodes of ontology **1050** are associated with elements of database schemata. The integration of database and ontology **1050** provides a unified representation for interpreting and acting on specific data entries in databases in terms of the larger sets of models and data in active ontology **1050**. For instance, the word "French" may be an entry in cuisines database **1662**. Because, in this example, database **1662** is integrated in active ontology **1050**, that same word "French" also has an interpretation as a possible cuisine served at a restaurant, which is involved in planning meal events, and this cuisine serves as a constraint to use when using restaurants reservation services, and so forth. Active ontologies can thus integrate databases into the modeling and execution environment to inter-operate with other components of automated assistant **1002**.

[0227] As described above, active ontology **1050** allows the author, designer, or system builder to integrate components; thus, in the example of FIG. **8**, the elements of a component such as constraint in dialog flow model **1642** can be identified with elements of other components such as required parameter of restaurant reservation service **1672**.

[0228] Active ontologies **1050** may be embodied as, for example, configurations of models, databases, and components in which the relationships among models, databases, and components are any of:

[0229] containership and/or inclusion;

[0230] relationship with links and/or pointers;

[0231] interface over APIs, both internal to a program and between programs.

[0232] For example, referring now to FIG. **9**, there is shown an example of an alternative embodiment of intelligent automated assistant system **1002**, wherein domain models **1056**, vocabulary **1058**, language pattern recognizers **1060**, short term personal memory **1052**, and long term personal memory **1054** components are organized under a common container associated with active ontology **1050**, and other components such as active input elicitation component(s) **1094**, language interpreter **1070** and dialog flow processor **1080** are associated with active ontology **1050** via API relationships.

Active Input Elicitation Component(s) **1094**

[0233] In at least one embodiment, active input elicitation component(s) **1094** (which, as described above, may be implemented in a stand-alone configuration or in a configuration including both server and client components) may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0234] Elicit, facilitate and/or process input from the user or the user's environment, and/or information about their need(s) or request(s). For example, if the user is looking to find a restaurant, the input elicitation module may get information about the user's constraints or preferences for location, time, cuisine, price, and so forth.

[0235] Facilitate different kinds of input from various sources, such as for example, one or more of the following (or combinations thereof):

[0236] input from keyboards or any other input device that generates text

[0237] input from keyboards in user interfaces that offer dynamic suggested completions of partial input

[0238] input from voice or speech input systems

[0239] input from Graphical User Interfaces (GUIs) in which users click, select, or otherwise directly manipulate graphical objects to indicate choices

[0240] input from other applications that generate text and send it to the automated assistant, including email, text messaging, or other text communication platforms

[0241] By performing active input elicitation, assistant **1002** is able to disambiguate intent at an early phase of input processing. For example, in an embodiment where input is provided by speech, the waveform might be sent to a server **1340** where words are extracted, and semantic interpretation performed. The results of such semantic interpretation can then be used to drive active input elicitation, which may offer the user alternative candidate words to choose among based on their degree of semantic fit as well as phonetic match.

[0242] In at least one embodiment, active input elicitation component(s) **1094** actively, automatically, and dynamically

guide the user toward inputs that may be acted upon by one or more of the services offered by embodiments of assistant **1002**. Referring now to FIG. **10**, there is shown a flow diagram depicting a method of operation for active input elicitation component(s) **1094** according to one embodiment.

[0243] The procedure begins **20**. In step **21**, assistant **1002** may offer interfaces on one or more input channels. For example, a user interface may offer the user options to speak or type or tap at any stage of a conversational interaction. In step **22**, the user selects an input channel by initiating input on one modality, such as pressing a button to start recording speech or to bring up an interface for typing.

[0244] In at least one embodiment, assistant **1002** offers default suggestions for the selected modality **23**. That is, it offers options **24** that are relevant in the current context prior to the user entering any input on that modality. For example, in a text input modality, assistant **1002** might offer a list of common words that would begin textual requests or commands such as, for example, one or more of the following (or combinations thereof): imperative verbs (e.g., find, buy, reserve, get, call, check, schedule, and the like), nouns (e.g., restaurants, movies, events, businesses, and the like), or menu-like options naming domains of discourse (e.g., weather, sports, news, and the like)

[0245] If the user selects one of the default options in **25**, and a preference to autosubmit **30** is set, the procedure may return immediately. This is similar to the operation of a conventional menu selection.

[0246] However, the initial option may be taken as a partial input, or the user may have started to enter a partial input **26**. At any point of input, in at least one embodiment, the user may choose to indicate that the partial input is complete **22**, which causes the procedure to return.

[0247] In **28**, the latest input, whether selected or entered, is added to the cumulative input.

[0248] In **29**, the system suggestions next possible inputs that are relevant given the current input and other sources of constraints on what constitutes relevant and/or meaningful input.

[0249] In at least one embodiment, the sources of constraints on user input (for example, which are used in steps **23** and **29**) are one or more of the various models and data sources that may be included in assistant **1002**, which may include, but are not limited to, one or more of the following (or combinations thereof):

[0250] Vocabulary **1058**. For example, words or phrases that match the current input may be suggested. In at least one embodiment, vocabulary may be associated with any or one or more nodes of active ontologies, domain models, task models, dialog models, and/or service models.

[0251] Domain models **1056**, which may constrain the inputs that may instantiate or otherwise be consistent with the domain model. For example, in at least one embodiment, domain models **1056** may be used to suggest concepts, relations, properties, and/or instances that would be consistent with the current input.

[0252] Language pattern recognizers **1060**, which may be used to recognize idioms, phrases, grammatical constructs, or other patterns in the current input and be used to suggest completions that fill out the pattern.

[0253] Domain entity databases **1072**, which may be used to suggest possible entities in the domain that match the input (e.g., business names, movie names, event names, and the like).

[0254] Short term memory **1052**, which may be used to match any prior input or portion of prior input, and/or any other property or fact about the history of interaction with a user. For example, partial input may be matched against cities that the user has encountered in a session, whether hypothetically (e.g., mentioned in queries) and/or physically (e.g., as determined from location sensors).

[0255] In at least one embodiment, semantic paraphrases of recent inputs, request, or results may be matched against the current input. For example, if the user had previously request "live music" and obtained concert listing, and then typed "music" in an active input elicitation environment, suggestions may include "live music" and/or "concerts".

[0256] Long term personal memory **1054**, which may be used to suggest matching items from long term memory. Such matching items may include, for example, one or more or any combination of: domain entities that are saved (e.g., "favorite" restaurants, movies, theaters, venues, and the like), to-do items, list items, calendar entries, people names in contacts/address books, street or city names mentioned in contact/address books, and the like.

[0257] Task flow models **1086**, which may be used to suggest inputs based on the next possible steps of in a task flow.

[0258] Dialog flow models **1087**, which may be used to suggest inputs based on the next possible steps of in a dialog flow.

[0259] Service capability models **1088**, which may be used to suggest possible services to employ, by name, category, capability, or any other property in the model. For example, a user may type part of the name of a preferred review site, and assistant **1002** may suggest a complete command for querying that review site for review.

[0260] In at least one embodiment, active input elicitation component(s) **1094** present to the user a conversational interface, for example, an interface in which the user and assistant communicate by making utterances back and forth in a conversational manner. Active input elicitation component(s) **1094** may be operable to perform and/or implement various types of conversational interfaces.

[0261] In at least one embodiment, active input elicitation component(s) **1094** may be operable to perform and/or implement various types of conversational interfaces in which assistant **1002** uses plies of the conversation to prompt for information from the user according to dialog models. Dialog models may represent a procedure for executing a dialog, such as, for example, a series of steps required to elicit the information needed to perform a service.

[0262] In at least one embodiment, active input elicitation component(s) **1094** offer constraints and guidance to the user in real time, while the user is in the midst of typing, speaking, or otherwise creating input. For example, active elicitation may guide the user to type text inputs that are recognizable by an embodiment of assistant **1002** and/or that may be serviced by one or more services offered by embodiments of assistant **1002**. This is an advantage over passively waiting for uncon-

strained input from a user because it enables the user's efforts to be focused on inputs that may or might be useful, and/or it enables embodiments of assistant **1002** to apply its interpretations of the input in real time as the user is inputting it.

[0263] At least a portion of the functions, operations, actions, and/or other features of active input elicitation described herein may be implemented, at least in part, using various methods and apparatuses described in U.S. patent application Ser. No. 11/518,292 for "Method and Apparatus for Building an Intelligent Automated Assistant", filed Sep. 8, 2006.

[0264] According to specific embodiments, multiple instances or threads of active input elicitation component(s) **1094** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software.

[0265] According to different embodiments, one or more different threads or instances of active input elicitation component(s) **1094** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of active input elicitation component(s) **1094**. Various examples of conditions or events which may trigger initiation and/or implementation of one or more different threads or instances of active input elicitation component(s) **1094** may include, but are not limited to, one or more of the following (or combinations thereof):

[0266] Start of user session. For example, when the user session starts up an application that is an embodiment of assistant **1002**, the interface may offer the opportunity for the user to initiate input, for example, by pressing a button to initiate a speech input system or clicking on a text field to initiate a text input session.

[0267] User input detected.

[0268] When assistant **1002** explicitly prompts the user for input, as when it requests a response to a question or offers a menu of next steps from which to choose.

[0269] When assistant **1002** is helping the user perform a transaction and is gathering data for that transaction, e.g., filling in a form.

[0270] In at least one embodiment, a given instance of active input elicitation component(s) **1094** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by active input elicitation component(s) **1094** may include, but are not limited to, one or more of the following (or combinations thereof):

[0271] database of possible words to use in a textual input;

[0272] grammar of possible phrases to use in a textual input utterance;

[0273] database of possible interpretations of speech input;

[0274] database of previous inputs from a user or from other users;

[0275] data from any of the various models and data sources that may be part of embodiments of assistant **1002**, which may include, but are not limited to, one or more of the following (or combinations thereof):

[0276] Domain models **1056**;

[0277] Vocabulary **1058**;

[0278] Language pattern recognizers **1060**;

[0279] Domain entity databases **1072**;

[0280] Short term memory **1052**;

[0281] Long term personal memory **1054**;

[0282] Task flow models **1086**;

[0283] Dialog flow models **1087**;

[0284] Service capability models **1088**.

[0285] According to different embodiments, active input elicitation component(s) **1094** may apply active elicitation procedures to, for example, one or more of the following (or combinations thereof):

[0286] typed input;

[0287] speech input;

[0288] input from graphical user interfaces (GUIs), including gestures;

[0289] input from suggestions offered in a dialog; and

[0290] events from the computational and/or sensed environments.

Active Typed Input Elicitation

[0291] Referring now to FIG. **11**, there is shown a flow diagram depicting a method for active typed input elicitation according to one embodiment.

[0292] The method begins **110**. Assistant **1002** receives **111** partial text input, for example via input device **1206**. Partial text input may include, for example, the characters that have been typed so far in a text input field. At any time, a user may indicate that the typed input is complete **112**, as, for example, by pressing an Enter key. If not complete, a suggestion generator generates **114** candidate suggestions **116**. These suggestions may be syntactic, semantic, and/or other kinds of suggestion based any of the sources of information or constraints described herein. If the suggestion is selected **118**, the input is transformed **117** to include the selected suggestion.

[0293] In at least one embodiment, the suggestions may include extensions to the current input. For example, a suggestion for "rest" may be "restaurants".

[0294] In at least one embodiment, the suggestions may include replacements of parts of the current input. For example, a suggestion for "rest" may be "places to eat".

[0295] In at least one embodiment, the suggestions may include replacing and rephrasing of parts of the current input. For example, if the current input is "find restaurants of style" a suggestion may be "italian" and when the suggestion is chosen, the entire input may be rewritten as "find Italian restaurants".

[0296] In at least one embodiment, the resulting input that is returned is annotated **119**, so that information about which choices were made in **118** is preserved along with the textual input. This enables, for example, the semantic concepts or entities underlying a string to be associated with the string when it is returned, which improves accuracy of subsequent language interpretation.

[0297] Referring now to FIGS. **12** to **21**, there are shown screen shots illustrating some portions of some of the procedures for active typed-input elicitation according to one embodiment. The screen shots depict an example of an embodiment of assistant **1002** as implemented on a smartphone such as the iPhone available from Apple Inc. of Cupertino, Calif. Input is provided to such device via a touchscreen, including on-screen keyboard functionality. One skilled in the art will recognize that the screen shots depict an embodiment that is merely exemplary, and that the techniques of the

present invention can be implemented on other devices and using other layouts and arrangements.

[0298] In FIG. **12**, screen **1201** includes a top-level set of suggestions **1202** shown when no input has been provided in field **1203**. This corresponds to no-input step **23** of FIG. **10** applied to step **114** of FIG. **11** where there is no input.

[0299] In FIG. **13**, screen **1301** depicts an example of the use of vocabulary to offer suggested completions **1303** of partial user input **1305** entered in field **1203** using on-screen keyboard **1304**. These suggested completions **1303** may be part of the function of active input elicitation **1094**. The user has entered partial user input **1305** including the string "comm". Vocabulary component **1058** has provided a mapping of this string into three different kinds of instances, which are listed as suggested completions **1303**: the phrase "community & local events" is a category of the events domain; "chambers of commerce" is a category of the local business search domain, and "Jewish Community Center" is the name of an instance of local businesses. Vocabulary component **1058** may provide the data lookup and management of name spaces like these. The user can tap Go button **1306** to indicate that he or she has finished entering input; this causes assistant **1002** to proceed with the completed text string as a unit of user input.

[0300] In FIG. **14**, screen **1401** depicts an example in which suggested semantic completions **1303** for a partial string "wh" **1305** include entire phrases with typed parameters. These kinds of suggestions may be enabled by the use of one or more of the various models and sources of input constraints described herein. For example, in one embodiment shown in FIG. **14**, "what is happening in city" is an active elicitation of the location parameter of the Local Events domain; "where is business name" is an active elicitation of the Business Name constraint of the Local Business Search domain; "what is showing at the venue name" is an active elicitation of the Venue Name constraint of the Local Events domain; and "what is playing at the movie theater" is an active elicitation of the Movie Theater Name constraint of the Local Events domain. These examples illustrate that the suggested completions are generated by models rather than simply drawn from a database of previously entered queries.

[0301] In FIG. **15**, screen **1501** depicts a continuation of the same example, after the user has entered additional text **1305** in field **1203**. Suggested completions **1303** are updated to match the additional text **1305**. In this example, data from a domain entity database **1072** were used: venues whose name starts with "f". Note that this is a significantly smaller and more semantically relevant set of suggestions than all words that begin with "f". Again, the suggestions are generated by applying a model, in this case the domain model that represents Local Events as happening at Venues, which are Businesses with Names. The suggestions actively elicit inputs that would make potentially meaningful entries when using a Local Events service.

[0302] In FIG. **16**, screen **1601** depicts a continuation of the same example, after the user has selected one of suggested completions **1303**. Active elicitation continues by prompting the user to further specify the type of information desired, here by presenting a number of specifiers **1602** from which the user can select. In this example, these specifiers are generated by the domain, task flow, and dialog flow models. The Domain is Local Events, which includes Categories of events that happen on Dates in Locations and have Event Names and Feature Performers. In this embodiment, the fact that these five options are offered to the user is generated from the Dialog Flow model that indicates that users should be asked for Constraints that they have not yet entered and from the Service Model that indicates that these five Constraints are parameters to Local Event services available to the assistant. Even the choice of preferred phrases to use as specifiers, such as "by category" and "featured", are generated from the Domain Vocabulary databases.

[0303] In FIG. **17**, screen **1701** depicts a continuation of the same example, after the user has selected one of specifiers **1602**.

[0304] In FIG. **18**, screen **1801** depicts a continuation of the same example, wherein the selected specifier **1602** has been added to field **1203**, and additional specifiers **1602** are presented. The user can select one of specifiers **1602** and/or provide additional text input via keyboard **1304**.

[0305] In FIG. **19**, screen **1901** depicts a continuation of the same example, wherein the selected specifier **1602** has been added to field **1203**, and yet more specifiers **1602** are presented. In this example, previously entered constraints are not actively elicited redundantly.

[0306] In FIG. **20**, screen **2001** depicts a continuation of the same example, wherein the user has tapped the Go button **1306**. The user's input is shown in box **2002**, and a message is shown in box **2003**, providing feedback to the user as to the query being performed in response to the user's input.

[0307] In FIG. **21**, screen **2101** depicts a continuation of the same example, wherein results have been found. Message is shown in box **2102**. Results **2103**, including input elements allowing the user to view further details, save the identified event, buy tickets, add notes, or the like.

[0308] In one screen **2101**, and other displayed screens, are scrollable, allowing the user to scroll upwards to see screen **2001** or other previously presented screens, and to make changes to the query if desired.

Active Speech Input Elicitation

[0309] Referring now to FIG. **22**, there is shown a flow diagram depicting a method for active input elicitation for voice or speech input according to one embodiment.

[0310] The method begins **221**. Assistant **1002** receives **121** voice or speech input in the form of an auditory signal. A speech-to-text service **122** or processor generates a set of candidate text interpretations **124** of the auditory signal. In one embodiment, speech-to-text service **122** is implemented using, for example, Nuance Recognizer, available from Nuance Communications, Inc. of Burlington, Mass.

[0311] In one embodiment, assistant **1002** employs statistical language models to generate candidate text interpretations **124** of speech input **121**.

[0312] In addition, in one embodiment, the statistical language models are tuned to look for words, names, and phrases that occur in the various models of assistant **1002** shown in FIG. **8**. For example, in at least one embodiment the statistical language models are given words, names, and phrases from some or all of: domain models **1056** (e.g., words and phrases relating to restaurant and meal events), task flow models **1086** (e.g., words and phrases relating to planning an event), dialog flow models **1087** (e.g., words and phrases related to the constraints that are needed to gather the inputs for a restaurant reservation), domain entity databases **1072** (e.g., names of restaurants), vocabulary databases **1058** (e.g., names of cuisines), service models **1088** (e.g., names of service provides

such as OpenTable), and/or any words, names, or phrases associated with any node of active ontology **1050**.

[0313] In one embodiment, the statistical language models are also tuned to look for words, names, and phrases from long-term personal memory **1054**. For example, statistical language models can be given text from to-do items, list items, personal notes, calendar entries, people names in contacts/address books, email addresses, street or city names mentioned in contact/address books, and the like.

[0314] A ranking component analyzes the candidate interpretations **124** and ranks **126** them according to how well they fit syntactic and/or semantic models of intelligent automated assistant **1002**. Any sources of constraints on user input may be used. For example, in one embodiment, assistant **1002** may rank the output of the speech-to-text interpreter according to how well the interpretations parse in a syntactic and/or semantic sense, a domain model, task flow model, and/or dialog model, and/or the like: it evaluates how well various combinations of words in the text interpretations **124** would fit the concepts, relations, entities, and properties of active ontology **1050** and its associated models. For example, if speech-to-text service **122** generates the two candidate interpretations "italian food for lunch" and "italian shoes for lunch", the ranking by semantic relevance **126** might rank "italian food for lunch" higher if it better matches the nodes assistant's **1002** active ontology **1050** (e.g., the words "italian", "food" and "lunch" all match nodes in ontology **1050** and they are all connected by relationships in ontology **1050**, whereas the word "shoes" does not match ontology **1050** or matches a node that is not part of the dining out domain network).

[0315] In various embodiments, algorithms or procedures used by assistant **1002** for interpretation of text inputs, including any embodiment of the natural language processing procedure shown in FIG. **28**, can be used to rank and score candidate text interpretations **124** generated by speech-to-text service **122**.

[0316] In one embodiment, if ranking component **126** determines **128** that the highest-ranking speech interpretation from interpretations **124** ranks above a specified threshold, the highest-ranking interpretation may be automatically selected **130**. If no interpretation ranks above a specified threshold, possible candidate interpretations of speech **134** are presented **132** to the user. The user can then select **136** among the displayed choices.

[0317] In various embodiments, user selection **136** among the displayed choices can be achieved by any mode of input, including for example any of the modes of multimodal input described in connection with FIG. **16**. Such input modes include, without limitation, actively elicited typed input **2610**, actively elicited speech input **2620**, actively presented GUI for input **2640**, and/or the like. In one embodiment, the user can select among candidate interpretations **134**, for example by tapping or speaking. In the case of speaking, the possible interpretation of the new speech input is highly constrained by the small set of choices offered **134**. For example, if offered "Did you mean italian food or italian shoes?" the user can just say "food" and the assistant can match this to the phrase "italian food" and not get it confused with other global interpretations of the input.

[0318] Whether input is automatically selected **130** or selected **136** by the user, the resulting input **138** is returned. In at least one embodiment, the returned input is annotated **138**, so that information about which choices were made in step **136** is preserved along with the textual input. This enables, for example, the semantic concepts or entities underlying a string to be associated with the string when it is returned, which improves accuracy of subsequent language interpretation. For example, if "Italian food" was offered as one of the candidate interpretations **134** based on a semantic interpretation of Cuisine=ItalianFood, then the machine-readable semantic interpretation can be sent along with the user's selection of the string "Italian food" as annotated text input **138**.

[0319] In at least one embodiment, candidate text interpretations **124** are generated based on speech interpretations received as output of speech-to-text service **122**.

[0320] In at least one embodiment, candidate text interpretations **124** are generated by paraphrasing speech interpretations in terms of their semantic meaning. In some embodiments, there can be multiple paraphrases of the same speech interpretation, offering different word sense or homonym alternatives. For example, if speech-to-text service **122** indicates "place for meet", the candidate interpretations presented to the user could be paraphrased as "place to meet (local businesses)" and "place for meat (restaurants)".

[0321] In at least one embodiment, candidate text interpretations **124** include offers to correct substrings.

[0322] In at least one embodiment, candidate text interpretations **124** include offers to correct substrings of candidate interpretations using syntactic and semantic analysis as described herein.

[0323] In at least one embodiment, when the user selects a candidate interpretation, it is returned.

[0324] In at least one embodiment, the user is offered an interface to edit the interpretation before it is returned.

[0325] In at least one embodiment, the user is offered an interface to continue with more voice input before input is returned. This enables one to incrementally build up an input utterance, getting syntactic and semantic corrections, suggestions, and guidance at one iteration.

[0326] In at least one embodiment, the user is offered an interface to proceed directly from **136** to step **111** of a method of active typed input elicitation (described above in connection with FIG. **11**). This enables one to interleave typed and spoken input, getting syntactic and semantic corrections, suggestions, and guidance at one step.

[0327] In at least one embodiment, the user is offered an interface to proceed directly from step **111** of an embodiment of active typed input elicitation to an embodiment of active speech input elicitation. This enables one to interleave typed and spoken input, getting syntactic and semantic corrections, suggestions, and guidance at one step.

Active GUI-Based Input Elicitation

[0328] Referring now to FIG. **23**, there is shown a flow diagram depicting a method for active input elicitation for GUI-based input according to one embodiment.

[0329] The method begins **140**. Assistant **1002** presents **141** graphical user interface (GUI) on output device **1207**, which may include, for example, links and buttons. The user interacts **142** with at least one GUI element. Data **144** is received, and converted **146** to a uniform format. The converted data is then returned.

[0330] In at least one embodiment, some of the elements of the GUI are generated dynamically from the models of the active ontology, rather than written into a computer program. For example, assistant **1002** can offer a set of constraints to

guide a restaurant reservation service as regions for tapping on a screen, with each region representing the name of the constraint and/or a value. For instance, the screen could have rows of a dynamically generated GUI layout with regions for the constraints Cuisine, Location, and Price Range. If the models of the active ontology change, the GUI screen would automatically change without reprogramming.

Active Dialog Suggestion Input Elicitation

[0331] FIG. **24** is a flow diagram depicting a method for active input elicitation at the level of a dialog flow according to one embodiment. Assistant **1002** suggests **151** possible responses **152**. The user selects **154** a suggested response. The received input is converted **154** to a uniform format. The converted data is then returned.

[0332] In at least one embodiment, the suggestions offered in step **151** are offered as follow-up steps in a dialog and/or task flow.

[0333] In at least one embodiment, the suggestions offer options to refine a query, for example using parameters from a domain and/or task model. For example, one may be offered to change the assumed location or time of a request.

[0334] In at least one embodiment, the suggestions offer options to choose among ambiguous alternative interpretations given by a language interpretation procedure or component.

[0335] In at least one embodiment, the suggestions offer options to choose among ambiguous alternative interpretations given by a language interpretation procedure or component.

[0336] In at least one embodiment, the suggestions offer options to choose among next steps in a workflow associated dialog flow model **1087**. For example, dialog flow model **1087** may suggest that after gathering the constrained for one domain (e.g., restaurant dining), assistant **1002** should suggest other related domains (e.g., a movie nearby).

Active Monitoring for Relevant Events

[0337] In at least one embodiment, asynchronous events may be treated as inputs in an analogous manner to the other modalities of active elicited input. Thus, such events may be provided as inputs to assistant **1002**. Once interpreted, such events can be treated in a manner similar to any other input.

[0338] For example, a flight status change may initiate an alert notification to be sent to a user. If a flight is indicated as being late, assistant **1002** may continue the dialog by presenting alternative flights, making other suggestions, and the like, based on the detected event.

[0339] Such events can be of any type. For example, assistant **1002** might detect that the user just got home, or is lost (off a specified route), or that a stock price hit a threshold value, or that a television show the user is interested in is starting, or that a musician of interest is touring in the area. In any of these situations, assistant **1002** can proceed with a dialog in substantially the same manner as if the user had him- or herself initiated the inquiry. In one embodiment, events can even be based on data provided from other devices, for example to tell the user when a coworker has returned from lunch (the coworker's device can signal such an event to the user's device, at which time assistant **1002** installed on the user's device responds accordingly).

[0340] In one embodiment, the events can be notifications or alerts from a calendar, clock, reminder, or to-do application. For example, an alert from a calendar application about a dinner date can initiate a dialog with assistant **1002** about the dining event. The dialog can proceed as if the user had just spoken or typed the information about the upcoming dinner event, such as "dinner for 2 in San Francisco".

[0341] In one embodiment, the context of possible event trigger **162** can include information about people, places, times, and other data. These data can be used as part of the input to assistant **1002** to use in various steps of processing.

[0342] In one embodiment, these data from the context of event trigger **162** can be used to disambiguate speech or text inputs from the user. For example, if a calendar event alert includes the name of a person invited to the event, that information can help disambiguate input which might match several people with the same or similar name.

[0343] Referring now to FIG. **25**, there is shown a flow diagram depicting a method for active monitoring for relevant events according to one embodiment. In this example, event trigger events are sets of input **162**. Assistant **1002** monitors **161** for such events. Detected events may be filtered and sorted **164** for semantic relevance using models, data and information available from other components in intelligent automated assistant **1002**. For example, an event that reports a change in flight status may be given higher relevance if the short-term or long-term memory records for a user indicate that the user is on that flight and/or have made inquiries about it to assistant **1002**. This sorting and filtering may then present only the top events for review by the user, who may then choose to pick one or more and act on them.

[0344] Event data is converted **166** to a uniform input format, and returned.

[0345] In at least one embodiment, assistant **1002** may proactively offer services associated with events that were suggested for user attention. For example, if a flight status alert indicates a flight may be missed, assistant **1002** may suggest to the user a task flow for re-planning the itinerary or booking a hotel.

Input Elicitation Component(s) Example:

[0346] The following example is intended to help illustrate some of the various types of functions, operations, actions, and/or other features which may be provided by active input elicitation component(s) **1094**.

Example: Command Completion (What can the User Say to Assistant **1002**?)

[0347] The user is facing a text input box with the general instruction to enter "what do you want to do?" Depending on the context and the user input, any of several system responses are provided. Examples are shown below

Case: Null Input

[0348]

User Input	System Response	Comment
With an empty field and no dialog context, type a question or space	Find restaurants, hotels, movies, . . . Book a table, a room, a seat, . . . Order food, flowers, . . . Buy a ticket, a book,	Black text would be inserted into the input field. Italicized text would be the follow-up suggestions

-continued

User Input	System Response	Comment
character, or enter nothing for some period of time. |	a DVD, . . . Schedule a meeting, an event, . . . Remember to a thing to do . . . Explain results	once the text was inserted.

Case: First Word Input

[0349]

User Input	System Response	Comment
With empty dialog context If the user types the first letter of any command, it is suggested. For example F|	Find restaurants, hotels, movies, . . .	Black text would be inserted into the input field. Italicized text would be the follow-up suggestions once the text was inserted.

Case: Key Word Input

[0350]

User Input	System Response	Comment
With empty dialog context If the user types the first two letters of any keyword, it is suggested as well as command. For example Re|	Remember to a thing to do . . . Find restaurants by location, cuisine,	Black text would be inserted into the input field. Italicized text would be the follow-up suggestions once the text was inserted.

Case: Prompting for Arguments

[0351]

User Input	System Response	Comment
The user has entered or selected a "command" verb and nothing else. For example, Find__|	restaurants by location, cuisine, . . . hotels by location, availability, . . . movies by location, genre, . . . name of place	Offer not only the selection class but advertise the constraint options. Note that name of place is a just a prompt, and would not insert text.

Case: Suggesting Criteria

[0352]

User Input	System Response	Comment
The user has already entered enough to establish a task and domain, and now is prompted for constraints in order. Find restaurants__|	in city, zip code near home, office, place name named restaurant name	Black text would be inserted into the input field. Italicized text would be the follow-up suggestions once the text was inserted.

-continued

User Input	System Response	Comment
a task and domain, and now is prompted for constraints in order. Find restaurants__|	place name named restaurant name	field. Italicized text would be the follow-up suggestions once the text was inserted.

Case: Adding Criteria

[0353]

User Input	System Response	Comment
The user has said "in" and now is prompted for locations Find restaurants in__|	city, zip code Palo Alto, CA Menlo Park, CA	Until they enter the next word, the suggestions are a prompt for what kind of thing to type. The Places are cities from personal memory.
The user has said "near" and now is prompted for locations Find restaurants near__|	my home my work Palo Alto, CA Menlo Park, CA	The Places are one or more places from personal memory.

Case: Adding Location or Other Constraints

[0354]

User Input	System Response	Comment
Find restaurants in Palo Alto__|	serving food type or cuisine with availability tonight, tomorrow, . . . described as romantic, good service	Prompt for other constraints
Find restaurants in Palo Alto with availability__|	tonight tomorrow at 7 pm at 9 pm other time or date	The 7:00 and 9:00 are our suggestions based on current time

Case: Starting from Constraint, Unknown Task or Domain

User Input	System Response	Comment
romantic__|	restaurants by location, cuisine, . . . hotels by location, availability, . . . movies by location, genre, . . .	Constraint value from ontology tells you which selection classes to use.
comedy__|	movies by location, genre, . . . events by location, . . . clubs by location	"comedy" is a constraint value for genre in movies, a genre in events, and "comedy clubs" is a local directory business category

Example: Name Completion

[0355] Here, the user has typed some text without accepting any of the commands, or he or she is just extending a command with an entity name. The system may attempt to com-

plete the names, depending on context. It also disambiguates the domain.

Case: Words without Context

User Input	System Response	Comment
il for	Il Fornaio (restaurant) Ill Forgotten Gains (movie)	May require entity name lookup. Notice multi-word completion. Show domain as a suggestion.
tom cruise	movies	May require entity name lookup.
romantic	movies restaurants	Based on ontology only

Case: Name with Context

User Input	System Response	Comment
Find restaurants il for	Il Fornaio (restaurant)	May require entity name lookup. Uses selection class and location context from current and past queries to constrain completion. Show domain as a suggestion.
tom cruise	movies	May require entity name lookup.
Find movies rom	romantic Roman Holiday (movie)	Based on ontology only
g-rated	movies	

Example: Choosing Values from a Set

[0356] Here, the user is responding to a system request to enter a value for a specific parameter, such as location, time, cuisine, or genre. The user may either select from a list or enter a value. As she types, matching items from the list are shown as options. Examples are shown below.

Case: Valueclass Selection

[0357]

User Input	System Response	Comment
Restaurants serving	burgers Ice cream Hot dogs	User may type anything else and doesn't have to pick from the menu.
Restaurants serving h	Hamburgers Hot dogs Hot sauce	
Movies playing	today tonight Friday night	

Example: Reusing Previous Commands

[0358] Previous queries are also options to complete on in an autocomplete interface. They may be just matched as strings (when the input field is empty there are no known constraints) or they may be suggested as relevant when in certain situations.

[0359] Case: completion on previous queries

User Input	System Response	Comment
Ital	Italian restaurants (normal completion) Films starring Italian actors (recent query)	Using string matching to retrieve previous queries
Lunch	lunch places in marin (recent query) buy the book naked lunch	

Example: Retrieving Personal Memory Items

[0360] Assistant **1002** may remember certain events and/or entities in personal memory associated with the user. Autocomplete can be performed based on such remembered items. Examples appear below.

Case: Completion on Events and Entities in Personal Memory

[0361]

User Input	System Response	Comment
Mary	lunch with Mary on Saturday (from My Events) films called "Something about Mary"	
Lunch	lunch with Mary on Saturday (from My Events) buy the book naked lunch (from My Todos)	
Hob	Hobee's Restaurant in Palo Alto (from My Restaurants)	

Multimodal Active Input Elicitation

[0362] In at least one embodiment, active input elicitation component(s) **1094** may process input from a plurality of input modalities. At least one modality might be implemented with an active input elicitation procedure that takes advantages of the particular kinds of inputs and methods for selecting from suggested options. A described herein, they may be embodiments of procedures for active input elicitation for text input, speech input, GUI-based input, input in the context of a dialog, and/or input resulting from event triggers.

[0363] In at least one embodiment, for a single instance of intelligent automated assistant **1002**, there may be support for one or more (or any combination of) typed input, speech input, GUI input, dialog input, and/or event input.

[0364] Referring now to FIG. **26**, there is shown a flow diagram depicting a method for multimodal active input elicitation according to one embodiment. The method begins **100**. Inputs may be received concurrently from one or more or any combination of the input modalities, in any sequence. Thus, the method includes actively eliciting typed input **2610**, speech input **2620**, GUI-based input **2640**, input in the context of a dialog **2650**, and/or input resulting from event triggers **2660**. Any or all of these input sources are unified into unified input format **2690** and returned. Unified input format **2690** enables the other components of intelligent automated assistant **1002** to be designed and to operate independently of the particular modality of the input.

[0365] Offering active guidance for multiple modalities and levels enables constraint and guidance on the input beyond those available to isolated modalities. For example, the kinds of suggestions offered to choose among speech, text, and dialog steps are independent, so their combination is

a significant improvement over adding active elicitation techniques to individual modalities or levels.

[0366] Combining multiple sources of constraints as described herein (syntactic/linguistic, vocabulary, entity databases, domain models, task models, service models, and the like) and multiple places where these constraints may be actively applied (speech, text, GUI, dialog, and asynchronous events) provides a new level of functionality for human-machine interaction.

Domain Models Component(s) **1056**

[0367] Domain models **1056** component(s) include representations of the concepts, entities, relations, properties, and instances of a domain. For example, dining out domain model **1622** might include the concept of a restaurant as a business with a name and an address and phone number, the concept of a meal event with a party size and date and time associated with the restaurant.

[0368] In at least one embodiment, domain models component(s) **1056** of assistant **1002** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0369] Domain model component(s) **1056** may be used by automated assistant **1002** for several processes, including: eliciting input **100**, interpreting natural language **200**, dispatching to services **400**, and generating output **600**.

[0370] Domain model component(s) **1056** may provide lists of words that might match a domain concept or entity, such as names of restaurants, which may be used for active elicitation of input **100** and natural language processing **200**.

[0371] Domain model component(s) **1056** may classify candidate words in processes, for instance, to determine that a word is the name of a restaurant.

[0372] Domain model component(s) **1056** may show the relationship between partial information for interpreting natural language, for example that cuisine may be associated with business entities (e.g., "local Mexican food" may be interpreted as "find restaurants with style=Mexican", this inference is possible because of the information in domain model **1056**).

[0373] Domain model component(s) **1056** may organize information about services used in service orchestration **1082**, for example, that a particular web service may provide reviews of restaurants.

[0374] Domain model component(s) **1056** may provide the information for generating natural language paraphrases and other output formatting, for example, by providing canonical ways of describing concepts, relations, properties and instances.

[0375] According to specific embodiments, multiple instances or threads of the domain models component(s) **1056** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of domain models component(s) **1056** may be performed, implemented and/or initiated by one or more of the following types of systems, components, systems, devices, procedures, processes, and the like (or combinations thereof):

[0376] Domain models component(s) **1056** may be implemented as data structures that represent concepts, relations, properties, and instances. These data structures may be stored in memory, files, or databases.

[0377] Access to domain model component(s) **1056** may be implemented through direct APIs, network APIs, database query interfaces, and/or the like.

[0378] Creation and maintenance of domain models component(s) **1056** may be achieved, for example, via direct editing of files, database transactions, and/or through the use of domain model editing tools.

[0379] Domain models component(s) **1056** may be implemented as part of or in association with active ontologies **1050**, which combine models with instantiations of the models for servers and users.

[0380] According to various embodiments, one or more different threads or instances of domain models component (s) **1056** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of domain models component(s) **1056**. For example, trigger initiation and/or implementation of one or more different threads or instances of domain models component(s) **1056** may be triggered when domain model information is required, including during input elicitation, input interpretation, task and domain identification, natural language processing, service orchestration, and/or formatting output for users.

[0381] In at least one embodiment, a given instance of domain models component(s) **1056** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. For example, data from domain model component(s) **1056** may be associated with other model modeling components including vocabulary **1058**, language pattern recognizers **1060**, dialog flow models **1087**, task flow models **1086**, service capability models **1088**, domain entity databases **1072**, and the like. For example, businesses in domain entity databases **1072** that are classified as restaurants might be known by type identifiers which are maintained in the dining out domain model components.

Domain Models Component(s) Example:

[0382] Referring now to FIG. **27**, there is shown a set of screen shots illustrating an example of various types of functions, operations, actions, and/or other features which may be provided by domain models component(s) **1056** according to one embodiment.

[0383] In at least one embodiment, domain models component(s) **1056** are the unifying data representation that enables the presentation of information shown in screens **103**A and **103**B about a restaurant, which combines data from several distinct data sources and services and which includes, for example: name, address, business categories, phone number, identifier for saving to long term personal memory, identifier for sharing over email, reviews from multiple sources, map coordinates, personal notes, and the like.

Language Interpreter Component(s) **1070**

[0384] In at least one embodiment, language interpreter component(s) **1070** of assistant **1002** may be operable to perform and/or implement various types of functions, opera-

tions, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0385] Analyze user input and identify a set of parse results.

[0386] User input can include any information from the user and his/her device context that can contribute to understanding the user's intent, which can include, for example one or more of the following (or combinations thereof): sequences of words, the identity of gestures or GUI elements involved in eliciting the input, current context of the dialog, current device application and its current data objects, and/or any other personal dynamic data obtained about the user such as location, time, and the like. For example, in one embodiment, user input is in the form of the uniform annotated input format **2690** resulting from active input elicitation **1094**.

[0387] Parse results are associations of data in the user input with concepts, relationships, properties, instances, and/or other nodes and/or data structures in models, databases, and/or other representations of user intent and/context. Parse result associations can be complex mappings from sets and sequences of words, signals, and other elements of user input to one or more associated concepts, relations, properties, instances, other nodes, and/or data structures described herein.

[0388] Analyze user input and identify a set of syntactic parse results, which are parse results that associate data in the user input with structures that represent syntactic parts of speech, clauses and phrases including multi-word names, sentence structure, and/or other grammatical graph structures. Syntactic parse results are described in element **212** of natural language processing procedure described in connection with FIG. **28**.

[0389] Analyze user input and identify a set of semantic parse results, which are parse results that associate data in the user input with structures that represent concepts, relationships, properties, entities, quantities, propositions, and/or other representations of meaning and user intent. In one embodiment, these representations of meaning and intent are represented by sets of and/or elements of and/or instances of models or databases and/or nodes in ontologies, as described in element **220** of natural language processing procedure described in connection with FIG. **28**.

[0390] Disambiguate among alternative syntactic or semantic parse results as described in element **230** of natural language processing procedure described in connection with FIG. **28**.

[0391] Determine whether a partially typed input is syntactically and/or semantically meaningful in an autocomplete procedure such as one described in connection with FIG. **11**.

[0392] Help generate suggested completions **114** in an autocomplete procedure such as one described in connection with FIG. **11**.

[0393] Determine whether interpretations of spoken input are syntactically and/or semantically meaningful in a speech input procedure such as one described in connection with FIG. **22**.

[0394] According to specific embodiments, multiple instances or threads of language interpreter component(s) **1070** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software.

[0395] According to different embodiments, one or more different threads or instances of language interpreter component(s) **1070** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of language interpreter component(s) **1070**. Various examples of conditions or events which may trigger initiation and/or implementation of one or more different threads or instances of language interpreter component (s) **1070** may include, but are not limited to, one or more of the following (or combinations thereof):

[0396] while eliciting input, including but not limited to

[0397] Suggesting possible completions of typed input **114** (FIG. **11**);

[0398] Ranking interpretations of speech **126** (FIG. **22**);

[0399] When offering ambiguities as suggested responses in dialog **152** (FIG. **24**);

[0400] when the result of eliciting input is available, including when input is elicited by any mode of active multimodal input elicitation **100**.

[0401] In at least one embodiment, a given instance of language interpreter component(s) **1070** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of such database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by the Language Interpreter component(s) may include, but are not limited to, one or more of the following (or combinations thereof):

[0402] Domain models **1056**;

[0403] Vocabulary **1058**;

[0404] Domain entity databases **1072**;

[0405] Short term memory **1052**;

[0406] Long term personal memory **1054**;

[0407] Task flow models **1086**;

[0408] Dialog flow models **1087**;

[0409] Service capability models **1088**.

[0410] Referring now also to FIG. **29**, there is shown a screen shot illustrating natural language processing according to one embodiment. The user has entered (via voice or text) language input **2902** consisting of the phrase "who is playing this weekend at the filmore". This phrase is echoed back to the user on screen **2901**. Language interpreter component(s) **1070** component process input **2902** and generates a parse result. The parse result associates that input with a request to show the local events that are scheduled for any of the upcoming weekend days at any event venue whose name matches "filmore". A paraphrase of the parse results is shown as **2903** on screen **2901**.

[0411] Referring now also to FIG. **28**, there is shown a flow diagram depicting an example of a method for natural language processing according to one embodiment.

[0412] The method begins **200**. Language input **202** is received, such as the string "who is playing this weekend at the filmore" in the example of FIG. **29**. In one embodiment, the input is augmented by current context information, such as the current user location and local time. In word/phrase matching **210**, language interpreter component(s) **1070** find associations between user input and concepts. In this example, associations are found between the string "playing" and the concept of listings at event venues; the string "this

weekend" (along with the current local time of the user) and an instantiation of an approximate time period that represents the upcoming weekend; and the string "filmore" with the name of a venue. Word/phrase matching **210** may use data from, for example, language pattern recognizers **1060**, vocabulary database **1058**, active ontology **1050**, short term personal memory **1052**, and long term personal memory **1054**.

[0413] Language interpreter component(s) **1070** generate candidate syntactic parses **212** which include the chosen parse result but may also include other parse results. For example, other parse results may include those wherein "playing" is associated with other domains such as games or with a category of event such as sporting events.

[0414] Short- and/or long-term memory **1052**, **1054** can also be used by language interpreter component(s) **1070** in generating candidate syntactic parses **212**. Thus, input that was provided previously in the same session, and/or known information about the user, can be used, to improve performance, reduce ambiguity, and reinforce the conversational nature of the interaction. Data from active ontology **1050**, domain models **1056**, and task flow models **1086** can also be used, to implement evidential reasoning in determining valid candidate syntactic parses **212**.

[0415] In semantic matching **220**, language interpreter component(s) **1070** consider combinations of possible parse results according to how well they fit semantic models such as domain models and databases. In this case, the parse includes the associations (1) "playing" (a word in the user input) as "Local Event At Venue" (part of a domain model **1056** represented by a cluster of nodes in active ontology **1050**) and (2) "filmore" (another word in the input) as a match to an entity name in a domain entity database **1072** for Local Event Venues, which is represented by a domain model element and active ontology node (Venue Name).

[0416] Semantic matching **220** may use data from, for example, active ontology **1050**, short term personal memory **1052**, and long term personal memory **1054**. For example, semantic matching **220** may use data from previous references to venues or local events in the dialog (from short term personal memory **1052**) or personal favorite venues (from long term personal memory **1054**).

[0417] A set of candidate, or potential, semantic parse results is generated **222**.

[0418] In disambiguation step **230**, language interpreter component(s) **1070** weigh the evidential strength of candidate semantic parse results **222**. In this example, the combination of the parse of "playing" as "Local Event At Venue" and the match of "filmore" as a Venue Name is a stronger match to a domain model than alternative combinations where, for instance, "playing" is associated with a domain model for sports but there is no association in the sports domain for "filmore".

[0419] Disambiguation **230** may use data from, for example, the structure of active ontology **1050**. In at least one embodiment, the connections between nodes in an active ontology provide evidential support for disambiguating among candidate semantic parse results **222**. For example, in one embodiment, if three active ontology nodes are semantically matched and are all connected in active ontology **1050**, this indicates higher evidential strength of the semantic parse than if these matching nodes were not connected or connected by longer paths of connections in active ontology **1050**. For example, in one embodiment of semantic matching **220**, the parse that matches both Local Event At Venue and Venue Name is given increased evidential support because the combined representations of these aspects of the user intent are connected by links and/or relations in active ontology **1050**: in this instance, the Local Event node is connected to the Venue node which is connected to the Venue Name node which is connected to the entity name in the database of venue names.

[0420] In at least one embodiment, the connections between nodes in an active ontology that provide evidential support for disambiguating among candidate semantic parse results **222** are directed arcs, forming an inference lattice, in which matching nodes provide evidence for nodes to which they are connected by directed arcs.

[0421] In **232**, language interpreter component(s) **1070** sort and select **232** the top semantic parses as the representation of user intent **290**.

Domain Entity Database(s) **1072**

[0422] In at least one embodiment, domain entity database (s) **1072** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0423] Store data about domain entities. Domain entities are things in the world or computing environment that may be modeled in domain models. Examples may include, but are not limited to, one or more of the following (or combinations thereof):

[0424] Businesses of any kind;

[0425] Movies, videos, songs and/or other musical products, and/or any other named entertainment products;

[0426] Products of any kind;

[0427] Events;

[0428] Calendar entries;

[0429] Cities, states, countries, neighborhoods, and/or other geographic, geopolitical, and/or geospatial points or regions;

[0430] Named places such as landmarks, airports, and the like;

[0431] Provide database services on these databases, including but not limited to simple and complex queries, transactions, triggered events, and the like.

[0432] According to specific embodiments, multiple instances or threads of domain entity database(s) **1072** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of domain entity database(s) **1072** may be performed, implemented and/or initiated by database software and/or hardware residing on client(s) **1304** and/or on server(s) **1340**.

[0433] One example of a domain entity database **1072** that can be used in connection with the present invention according to one embodiment is a database of one or more businesses storing, for example, their names and locations. The database might be used, for example, to look up words contained in an input request for matching businesses and/or to look up the location of a business whose name is known. One

skilled in the art will recognize that many other arrangements and implementations are possible.

Vocabulary Component(s) **1058**

[0434] In at least one embodiment, vocabulary component (s) **1058** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0435] Provide databases associating words and strings with concepts, properties, relations, or instances of domain models or task models;

[0436] Vocabulary from vocabulary components may be used by automated assistant **1002** for several processes, including for example: eliciting input, interpreting natural language, and generating output.

[0437] According to specific embodiments, multiple instances or threads of vocabulary component(s) **1058** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of vocabulary component(s) **1058** may be implemented as data structures that associate strings with the names of concepts, relations, properties, and instances. These data structures may be stored in memory, files, or databases. Access to vocabulary component(s) **1058** may be implemented through direct APIs, network APIs, and/or database query interfaces. Creation and maintenance of vocabulary component(s) **1058** may be achieved via direct editing of files, database transactions, or through the use of domain model editing tools. Vocabulary component(s) **1058** may be implemented as part of or in association with active ontologies **1050**. One skilled in the art will recognize that many other arrangements and implementations are possible.

[0438] According to different embodiments, one or more different threads or instances of vocabulary component(s) **1058** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of vocabulary component(s) **1058**. In one embodiment, vocabulary component(s) **1058** are accessed whenever vocabulary information is required, including, for example, during input elicitation, input interpretation, and formatting output for users. One skilled in the art will recognize that other conditions or events may trigger initiation and/or implementation of one or more different threads or instances of vocabulary component(s) **1058**.

[0439] In at least one embodiment, a given instance of vocabulary component(s) **1058** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. In one embodiment, vocabulary component(s) **1058** may access data from external databases, for instance, from a data warehouse or dictionary.

Language Pattern Recognizer Component(s) **1060**

[0440] In at least one embodiment, language pattern recognizer component(s) **1060** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, looking for patterns in language or speech input that indicate grammatical, idiomatic, and/or other composites of input tokens. These patterns correspond to, for example, one or more of the following (or combinations thereof): words, names, phrases, data, parameters, commands, and/or signals of speech acts.

[0441] According to specific embodiments, multiple instances or threads of pattern recognizer component(s) **1060** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of language pattern recognizer component(s) **1060** may be performed, implemented and/or initiated by one or more files, databases, and/or programs containing expressions in a pattern matching language. In at least one embodiment, language pattern recognizer component(s) **1060** are represented declaratively, rather than as program code; this enables them to be created and maintained by editors and other tools other than programming tools. Examples of declarative representations may include, but are not limited to, one or more of the following (or combinations thereof): regular expressions, pattern matching rules, natural language grammars, parsers based on state machines and/or other parsing models.

[0442] One skilled in the art will recognize that other types of systems, components, systems, devices, procedures, processes, and the like (or combinations thereof) can be used for implementing language pattern recognizer component(s) **1060**.

[0443] According to different embodiments, one or more different threads or instances of language pattern recognizer component(s) **1060** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of language pattern recognizer component(s) **1060**. Various examples of conditions or events which may trigger initiation and/or implementation of one or more different threads or instances of language pattern recognizer component(s) **1060** may include, but are not limited to, one or more of the following (or combinations thereof):

[0444] during active elicitation of input, in which the structure of the language pattern recognizers may constrain and guide the input from the user;

[0445] during natural language processing, in which the language pattern recognizers help interpret input as language;

[0446] during the identification of tasks and dialogs, in which the language pattern recognizers may help identify tasks, dialogs, and/or steps therein.

[0447] In at least one embodiment, a given instance of language pattern recognizer component(s) **1060** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by language pattern recognizer component(s) **1060** may include, but are not limited to, data from any of the models various models and data sources that may be part of embodiments of assistant **1002**, which may include, but are not limited to, one or more of the following (or combinations thereof):

[0448] Domain models **1056**;

[0449] Vocabulary **1058**;

[0450] Domain entity databases **1072**;

[0451] Short term memory **1052**;

[0452] Long term personal memory **1054**;

[0453] Task flow models **1086**;

[0454] Dialog flow models **1087**;

[0455] Service capability models **1088**.

[0456] In one embodiment, access of data from other parts of embodiments of assistant **1002** may be coordinated by active ontologies **1050**.

[0457] Referring again to FIG. **14**, there is shown an example of some of the various types of functions, operations, actions, and/or other features which may be provided by language pattern recognizer component(s) **1060**. FIG. **14** illustrates language patterns that language pattern recognizer component(s) **1060** may recognize. For example, the idiom "what is happening" (in a city) may be associated with the task of event planning and the domain of local events.

Dialog Flow Processor Component(s) **1080**

[0458] In at least one embodiment, dialog flow processor component(s) **1080** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0459] Given a representation of the user intent **290** from language interpretation **200**, identify the task a user wants performed and/or a problem the user wants solved. For example, a task might be to find a restaurant.

[0460] For a given problem or task, given a representation of user intent **290**, identify parameters to the task or problem. For example, the user might be looking for a recommended restaurant that serves Italian food near the user's home. The constraints that a restaurant be recommended, serving Italian food, and near home are parameters to the task of finding a restaurant.

[0461] Given the task interpretation and current dialog with the user, such as that which may be represented in personal short term memory **1052**, select an appropriate dialog flow model and determine a step in the flow model corresponding to the current state.

[0462] According to specific embodiments, multiple instances or threads of dialog flow processor component(s) **1080** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software.

[0463] In at least one embodiment, a given instance of dialog flow processor component(s) **1080** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by dialog flow processor component(s) **1080** may include, but are not limited to, one or more of the following (or combinations thereof):

[0464] task flow models **1086**;

[0465] domain models **1056**;

[0466] dialog flow models **1087**.

[0467] Referring now to FIGS. **30** and **31**, there are shown screen shots illustrating an example of various types of functions, operations, actions, and/or other features which may be provided by dialog flow processor component(s) according to one embodiment.

[0468] As shown in screen **3001**, user requests a dinner reservation by providing speech or text input "book me a table for dinner". Assistant **1002** generates a prompt **3003** asking the user to specify time and party size.

[0469] Once these parameters have been provided, screen **3101** is shown. Assistant **1002** outputs a dialog box **3102** indicating that results are being presented, and a prompt **3103** asking the user to click a time. Listings **3104** are also displayed.

[0470] In one embodiment, such a dialog is implemented as follows. Dialog flow processor component(s) **1080** are given a representation of user intent from language interpreter component **1070** and determine that the appropriate response is to ask the user for information required to perform the next step in a task flow. In this case, the domain is restaurants, the task is getting a reservation, and the dialog step is to ask the user for information required to accomplish the next step in the task flow. This dialog step is exemplified by prompt **3003** of screen **3001**.

[0471] Referring now also to FIG. **32**, there is shown a flow diagram depicting a method of operation for dialog flow processor component(s) **1080** according to one embodiment. The flow diagram of FIG. **32** is described in connection with the example shown in FIGS. **30** and **31**.

[0472] The method begins **200**. Representation of user intent **290** is received. As described in connection with FIG. **28**, in one embodiment, representation of user intent **290** is a set of semantic parses. For the example shown in FIGS. **30** and **31**, the domain is restaurants, the verb is "book" associated with restaurant reservations, and the time parameter is the evening of the current day.

[0473] In **310**, dialog flow processor component(s) **1080** determine whether this interpretation of user intent is supported strongly enough to proceed, and/or if it is better supported than alternative ambiguous parses. In the current example, the interpretation is strongly supported, with no competing ambiguous parses. If, on the other hand, there are competing ambiguities or sufficient uncertainty, then step **322** is performed, to set the dialog flow step so that the execution phase causes the dialog to output a prompt for more information from the user.

[0474] In **312**, the dialog flow processor component(s) **1080** determine the preferred interpretation of the semantic parse with other information to determine the task to perform and its parameters. Information may be obtained, for example, from domain models **1056**, task flow models **1086**, and/or dialog flow models **1087**, or any combination thereof. In the current example, the task is identified as getting a reservation, which involves both finding a place that is reservable and available, and effecting a transaction to reserve a table. Task parameters are the time constraint along with others that are inferred in step **312**.

[0475] In **320**, the task flow model is consulted to determine an appropriate next step. Information may be obtained, for example, from domain models **1056**, task flow models **1086**, and/or dialog flow models **1087**, or any combination thereof. In the example, it is determined that in this task flow the next step is to elicit missing parameters to an availability search for restaurants, resulting in prompt **3003** illustrated in FIG. **30**, requesting party size and time for a reservation.

[0476] As described above, FIG. **31** depicts screen **3101** is shown including dialog element **3102** that is presented after the user answers the request for the party size and reservation time. In one embodiment, screen **3101** is presented as the

result of another iteration through an automated call and response procedure, as described in connection with FIG. **33**, which leads to another call to the dialog and flow procedure depicted in FIG. **32**. In this instantiation of the dialog and flow procedure, after receiving the user preferences, dialog flow processor component(s) **1080** determines a different task flow step in step **320**: to do an availability search. When request **390** is constructed, it includes the task parameters sufficient for dialog flow processor component(s) **1080** and services orchestration component(s) **1082** to dispatch to a restaurant booking service.

Dialog Flow Models Component(s) **1087**

[0477] In at least one embodiment, dialog flow models component(s) **1087** may be operable to provide dialog flow models, which represent the steps one takes in a particular kind of conversation between a user and intelligent automated assistant **1002**. For example, the dialog flow for the generic task of performing a transaction includes steps for getting the necessary data for the transaction and confirming the transaction parameters before committing it.

Task Flow Models Component(s) **1086**

[0478] In at least one embodiment, task flow models component(s) **1086** may be operable to provide task flow models, which represent the steps one takes to solve a problem or address a need. For example, the task flow for getting a dinner reservation involves finding a desirable restaurant, checking availability, and doing a transaction to get a reservation for a specific time with the restaurant.

[0479] According to specific embodiments, multiple instances or threads of task flow models component(s) **1086** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of task flow models component(s) **1086** may be may be implemented as programs, state machines, or other ways of identifying an appropriate step in a flow graph.

[0480] In at least one embodiment, task flow models component(s) **1086** may use a task modeling framework called generic tasks. Generic tasks are abstractions that model the steps in a task and their required inputs and generated outputs, without being specific to domains. For example, a generic task for transactions might include steps for gathering data required for the transaction, executing the transaction, and outputting results of the transaction—all without reference to any particular transaction domain or service for implementing it. It might be instantiated for a domain such as shopping, but it is independent of the shopping domain and might equally well apply to domains of reserving, scheduling, and the like.

[0481] At least a portion of the functions, operations, actions, and/or other features associated with task flow models component(s) **1086** and/or procedure(s) described herein may be implemented, at least in part, using concepts, features, components, processes, and/or other aspects disclosed herein in connection with generic task modeling framework.

[0482] Additionally, at least a portion of the functions, operations, actions, and/or other features associated with task flow models component(s) **1086** and/or procedure(s) described herein may be implemented, at least in part, using concepts, features, components, processes, and/or other aspects relating to constrained selection tasks, as described herein. For example, one embodiment of generic tasks may be implemented using a constrained selection task model.

[0483] In at least one embodiment, a given instance of task flow models component(s) **1086** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by task flow models component(s) **1086** may include, but are not limited to, one or more of the following (or combinations thereof):

[0484] Domain models **1056**;
[0485] Vocabulary **1058**;
[0486] Domain entity databases **1072**;
[0487] Short term memory **1052**;
[0488] Long term personal memory **1054**;
[0489] Dialog flow models **1087**;
[0490] Service capability models **1088**.

[0491] Referring now to FIG. **34**, there is shown a flow diagram depicting an example of task flow for a constrained selection task **351** according to one embodiment.

[0492] Constrained selection is a kind of generic task in which the goal is to select some item from a set of items in the world based on a set of constraints. For example, a constrained selection task **351** may be instantiated for the domain of restaurants. Constrained selection task **351** starts by soliciting criteria and constraints from the user **352**. For example, the user might be interested in Asian food and may want a place to eat near his or her office.

[0493] In step **353**, assistant **1002** presents items that meet the stated criteria and constraints for the user to browse. In this example, it may be a list of restaurants and their properties which may be used to select among them.

[0494] In step **354**, the user is given an opportunity to refine criteria and constraints. For example, the user might refine the request by saying "near my office". The system would then present a new set of results in step **353**.

[0495] Referring now also to FIG. **35**, there is shown an example of screen **3501** including list **3502** of items presented by constrained selection task **351** according to one embodiment.

[0496] In step **355**, the user can select among the matching items. Any of a number of follow-on tasks **359** may then be made available, such as for example book **356**, remember **357**, or share **358**. In various embodiments, follow-on tasks **359** can involve interaction with web-enabled services, and/or with functionality local to the device (such as setting a calendar appointment, making a telephone call, sending an email or text message, setting an alarm, and the like).

[0497] In the example of FIG. **35**, the user can select an item within list **3502** to see more details and to perform additional actions. Referring now also to FIG. **36**, there is shown an example of screen **3601** after the user has selected an item from list **3502**. Additional information and options corresponding to follow-on tasks **359** concerning the selected item are displayed.

[0498] In various embodiments, the flow steps may be offered to the user in any of several input modalities, including but not limited to any combination of explicit dialog prompts and GUI links.

Services Component(s) **1084**

[0499] Services component(s) **1084** represent the set of services that intelligent automated assistant **1002** might call

on behalf of the user. Any service that can be called may be offered in a services component **1084**.

[0500] In at least one embodiment, services component(s) **1084** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0501] Provide the functions over an API that would normally be provided by a web-based user interface to a service. For example, a review website might provide a service API that would return reviews of a given entity automatically when called by a program. The API offers to intelligent automated assistant **1002** the services that a human would otherwise obtain by operating the user interface of the website.

[0502] Provide the functions over an API that would normally be provided by a user interface to an application. For example, a calendar application might provide a service API that would return calendar entries automatically when called by a program. The API offers to intelligent automated assistant **1002** the services that a human would otherwise obtain by operating the user interface of the application. In one embodiment, assistant **1002** is able to initiate and control any of a number of different functions available on the device. For example, if assistant **1002** is installed on a smartphone, personal digital assistant, tablet computer, or other device, assistant **1002** can perform functions such as: initiate applications, make calls, send emails and/or text messages, add calendar events, set alarms, and the like. In one embodiment, such functions are activated using services component(s) **1084**.

[0503] Provide services that are not currently implemented in a user interface, but that are available through an API to assistant in larger tasks. For example, in one embodiment, an API to take a street address and return machine-readable geo-coordinates might be used by assistant **1002** as a service component **1084** even if it has no direct user interface on the web or a device.

[0504] According to specific embodiments, multiple instances or threads of services component(s) **1084** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of services component(s) **1084** may be performed, implemented and/or initiated by one or more of the following types of systems, components, systems, devices, procedures, processes, and the like (or combinations thereof):

[0505] implementation of an API exposed by a service, locally or remotely or any combination;

[0506] inclusion of a database within automated assistant **1002** or a database service available to assistant **1002**.

[0507] For example, a website that offers users an interface for browsing movies might be used by an embodiment of intelligent automated assistant **1002** as a copy of the database used by the website. Services component(s) **1084** would then offer an internal API to the data, as if it were provided over a network API, even though the data is kept locally.

[0508] As another example, services component(s) **1084** for an intelligent automated assistant **1002** that helps with restaurant selection and meal planning might include any or all of the following set of services which are available from third parties over the network:

[0509] a set of restaurant listing services which lists restaurants matching name, location, or other constraints;

[0510] a set of restaurant rating services which return rankings for named restaurants;

[0511] a set of restaurant reviews services which returns written reviews for named restaurants;

[0512] a geocoding service to locate restaurants on a map;

[0513] a reservation service that enables programmatic reservation of tables at restaurants.

Services Orchestration Component(s) **1082**

[0514] Services orchestration component(s) **1082** of intelligent automated assistant **1002** executes a service orchestration procedure.

[0515] In at least one embodiment, services orchestration component(s) **1082** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0516] Dynamically and automatically determine which services may meet the user's request and/or specified domain(s) and task(s);

[0517] Dynamically and automatically call multiple services, in any combination of concurrent and sequential ordering;

[0518] Dynamically and automatically transform task parameters and constraints to meet input requirements of service APIs;

[0519] Dynamically and automatically monitor for and gather results from multiple services;

[0520] Dynamically and automatically merge service results data from various services into to a unified result model;

[0521] Orchestrate a plurality of services to meet the constraints of a request;

[0522] Orchestrate a plurality of services to annotate an existing result set with auxiliary information;

[0523] Output the result of calling a plurality of services in a uniform, service independent representation that unifies the results from the various services (for example, as a result of calling several restaurant services that return lists of restaurants, merge the data on at least one restaurant from the several services, removing redundancy).

[0524] For example, in some situations, there may be several ways to accomplish a particular task. For example, user input such as "remind me to leave for my meeting across town at 2 pm" specifies an action that can be accomplished in at least three ways: set alarm clock; create a calendar event; or call a to-do manager. In one embodiment, services orchestration component(s) **1082** makes the determination as to which way to best satisfy the request.

[0525] Services orchestration component(s) **1082** can also make determinations as to which combination of several services would be best to invoke in order to perform a given overall task. For example, to find and reserve a table for dinner, services orchestration component(s) **1082** would make determinations as to which services to call in order to perform such functions as looking up reviews, getting availability, and making a reservation. Determination of which services to use may depend on any of a number of different

factors. For example, in at least one embodiment, information about reliability, ability of service to handle certain types of requests, user feedback, and the like, can be used as factors in determining which service(s) is/are appropriate to invoke.

[0526] According to specific embodiments, multiple instances or threads of services orchestration component(s) **1082** may be concurrently implemented and/or initiated via the use of one or more processors and/or other combinations of hardware and/or hardware and software.

[0527] In at least one embodiment, a given instance of services orchestration component(s) **1082** may use explicit service capability models **1088** to represent the capabilities and other properties of external services, and reason about these capabilities and properties while achieving the features of services orchestration component(s) **1082**. This affords advantages over manually programming a set of services that may include, for example, one or more of the following (or combinations thereof):

[0528] Ease of development;

[0529] Robustness and reliability in execution;

[0530] The ability to dynamically add and remove services without disrupting code;

[0531] The ability to implement general distributed query optimization algorithms that are driven by the properties and capabilities rather than hard coded to specific services or APIs.

[0532] In at least one embodiment, a given instance of services orchestration component(s) **1082** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Examples of different types of data which may be accessed by services orchestration component(s) **1082** may include, but are not limited to, one or more of the following (or combinations thereof):

[0533] Instantiations of domain models;

[0534] Syntactic and semantic parses of natural language input;

[0535] Instantiations of task models (with values for parameters);

[0536] Dialog and task flow models and/or selected steps within them;

[0537] Service capability models **1088**;

[0538] Any other information available in an active ontology **1050**.

[0539] Referring now to FIG. **37**, there is shown an example of a procedure for executing a service orchestration procedure according to one embodiment.

[0540] In this particular example, it is assumed a single user is interesting in finding a good place for dinner at a restaurant, and is engaging intelligent automated assistant **1002** in a conversation to help provide this service.

[0541] Consider the task of finding restaurants that are of high quality, are well reviewed, near a particular location, available for reservation at a particular time, and serve a particular kind of food. These domain and task parameters are given as input **390**.

[0542] The method begins **400**. At **402**, it is determined whether the given request may require any services. In some situations, services delegation may not be required, for example if assistant **1002** is able to perform the desired task itself. For example, in one embodiment, assistant **1002** may be able to answer a factual question without invoking services delegation. Accordingly, if the request does not require services, then standalone flow step is executed in **403** and its result **490** is returned. For example, if the task request was to ask for information about automated assistant **1002** itself, then the dialog response may be handled without invoking any external services.

[0543] If, in step **402**, it is determined that services delegation is required, services orchestration component(s) **1082** proceed to step **404**. In **404**, services orchestration component (s) **1082** may match up the task requirements with declarative descriptions of the capabilities and properties of services in service capability models **1088**. At least one service provider that might support the instantiated operation provides declarative, qualitative metadata detailing, for example, one or more of the following (or combinations thereof):

[0544] the data fields that are returned with results;

[0545] which classes of parameters the service provider is statically known to support;

[0546] policy functions for parameters the service provider might be able to support after dynamic inspection of the parameter values;

[0547] a performance rating defining how the service performs (e.g. relational DB, web service, triple store, full-text index, or some combination thereof);

[0548] property quality ratings statically defining the expected quality of property values returned with the result object;

[0549] an overall quality rating of the results the service may expect to return.

[0550] For example, reasoning about the classes of parameters that service may support, a service model may state that services **1**, **2**, **3**, and **4** may provide restaurants that are near a particular location (a parameter), services **2** and **3** may filter or rank restaurants by quality (another parameter), services **3**, **4**, and **5** may return reviews for restaurants (a data field returned), service **6** may list the food types served by restaurants (a data field returned), and service **7** may check availability of restaurants for particular time ranges (a parameter). Services **8** through **99** offer capabilities that are not required for this particular domain and task.

[0551] Using this declarative, qualitative metadata, the task, the task parameters, and other information available from the runtime environment of the assistant, services orchestration component(s) **1082** determines **404** an optimal set of service providers to invoke. The optimal set of service providers may support one or more task parameters (returning results that satisfy one or more parameters) and also considers the performance rating of at least one service provider and the overall quality rating of at least one service provider.

[0552] The result of step **404** is a dynamically generated list of services to call for this particular user and request.

[0553] In at least one embodiment, services orchestration component(s) **1082** considers the reliability of services as well as their ability to answer specific information requests.

[0554] In at least one embodiment, services orchestration component(s) **1082** hedges against unreliability by calling overlapping or redundant services.

[0555] In at least one embodiment, services orchestration component(s) **1082** considers personal information about the user (from the short term personal memory component) to select services. For example, the user may prefer some rating services over others.

[0556] In step **450**, services orchestration component(s) **1082** dynamically and automatically invokes multiple services on behalf of a user. In at least one embodiment, these are called dynamically while responding to a user's request. According to specific embodiments, multiple instances or threads of the services may be concurrently called. In at least one embodiment, these are called over a network using APIs, or over a network using web service APIs, or over the Internet using web service APIs, or any combination thereof.

[0557] In at least one embodiment, the rate at which services are called is programmatically limited and/or managed.

[0558] Referring now also to FIG. **38**, there is shown an example of a service invocation procedure **450** according to one embodiment. Service invocation is used, for example, to obtain additional information or to perform tasks by the use of external services. In one embodiment, request parameters are transformed as appropriate for the service's API. Once results are received from the service, the results are transformed to a results representation for presentation to the user within assistant **1002**.

[0559] In at least one embodiment, services invoked by service invocation procedure **450** can be a web service, application running on the device, operating system function, or the like.

[0560] Representation of request **390** is provided, including for example task parameters and the like. For at least one service available from service capability models **1088**, service invocation procedure **450** performs transformation **452**, calling **454**, and output-mapping **456** steps.

[0561] In transformation step **452**, the current task parameters from request representation **390** are transformed into a form that may be used by at least one service. Parameters to services, which may be offered as APIs or databases, may differ from the data representation used in task requests, and also from at least one other. Accordingly, the objective of step **452** is to map at least one task parameter in the one or more corresponding formats and values in at least one service being called.

[0562] For example, the names of businesses such as restaurants may vary across services that deal with such businesses. Accordingly, step **452** would involve transforming any names into forms that are best suited for at least one service.

[0563] As another example, locations are known at various levels of precision and using various units and conventions across services. Service 1 might may require ZIP codes, service 2 GPS coordinates, and service 3 postal street addresses.

[0564] The service is called **454** over an API and its data gathered. In at least one embodiment, the results are cached. In at least one embodiment, the services that do not return within a specified level performance (e.g., as specified in Service Level Agreement or SLA) are dropped.

[0565] In output mapping step **456**, the data returned by a service is mapped back onto unified result representation **490**. This step may include dealing with different formats, units, and so forth.

[0566] In step **412**, results from multiple services are validated and merged. In one embodiment, if validated results are collected, an equality policy function—defined on a per-domain basis—is then called pair-wise across one or more results to determine which results represent identical concepts in the real world. When a pair of equal results is discovered, a set of property policy functions—also defined on a per-domain basis—are used to merge property values into a merged result. The property policy function may use the property quality ratings from the service capability models, the task parameters, the domain context, and/or the long-term personal memory **1054** to decide the optimal merging strategy.

[0567] For example, lists of restaurants from different providers of restaurants might be merged and duplicates removed. In at least one embodiment, the criteria for identifying duplicates may include fuzzy name matching, fuzzy location matching, fuzzy matching against multiple properties of domain entities, such as name, location, phone number, and/or website address, and/or any combination thereof.

[0568] In step **414**, the results are sorted and trimmed to return a result list of the desired length.

[0569] In at least one embodiment, a request relaxation loop is also applied. If, in step **416**, services orchestration component(s) **1082** determines that the current result list is not sufficient (e.g., it has fewer than the desired number of matching items), then task parameters may be relaxed **420** to allow for more results. For example, if the number of restaurants of the desired sort found within N miles of the target location is too small, then relaxation would run the request again, looking in an area larger than N miles away, and/or relaxing some other parameter of the search.

[0570] In at least one embodiment, the service orchestration method is applied in a second pass to "annotate" results with auxiliary data that is useful to the task.

[0571] In step **418**, services orchestration component(s) **1082** determines whether annotation is required. It may be required if, for example, if the task may require a plot of the results on a map, but the primary services did not return geo-coordinates required for mapping.

[0572] In **422**, service capability models **1088** are consulted again to find services that may return the desired extra information. In one embodiment, the annotation process determines if additional or better data may be annotated to a merged result. It does this by delegating to a property policy function—defined on a per-domain basis—for at least one property of at least one merged result. The property policy function may use the merged property value and property quality rating, the property quality ratings of one or more other service providers, the domain context, and/or the user profile to decide if better data may be obtained. If it is determined that one or more service providers may annotate one or more properties for a merged result, a cost function is invoked to determine the optimal set of service providers to annotate.

[0573] At least one service provider in the optimal set of annotation service providers is then invoked **450** with the list of merged results, to obtain results **424**. The changes made to at least one merged result by at least one service provider are tracked during this process, and the changes are then merged using the same property policy function process as was used in step **412**. Their results are merged **426** into the existing result set.

[0574] The resulting data is sorted **428** and unified into a uniform representation **490**.

[0575] It may be appreciated that one advantage of the methods and systems described above with respect to services orchestration component(s) **1082** is that they may be advantageously applied and/or utilized in various fields of technology other than those specifically relating to intelligent automated assistants. Examples of such other areas of

technologies where aspects and/or features of service orchestration procedures include, for example, one or more of the following:

[0576] Dynamic "mash ups" on websites and web-based applications and services;

[0577] Distributed database query optimization;

[0578] Dynamic service oriented architecture configuration.

Service Capability Models Component(s) **1088**

[0579] In at least one embodiment, service capability models component(s) **1088** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0580] Provide machine readable information about the capabilities of services to perform certain classes of computation;

[0581] Provide machine readable information about the capabilities of services to answer certain classes of queries;

[0582] Provide machine readable information about which classes of transactions are provided by various services;

[0583] Provide machine readable information about the parameters to APIs exposed by various services;

[0584] Provide machine readable information about the parameters that may be used in database queries on databases provided by various services.

Output Processor Component(s) **1090**

[0585] In at least one embodiment, output processor component(s) **1090** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0586] Format output data that is represented in a uniform internal data structure into forms and layouts that render it appropriately on different modalities. Output data may include, for example, communication in natural language between the intelligent automated assistant and the user; data about domain entities, such as properties of restaurants, movies, products, and the like; domain specific data results from information services, such as weather reports, flight status checks, prices, and the like; and/or interactive links and buttons that enable the user to respond by directly interacting with the output presentation.

[0587] Render output data for modalities that may include, for example, any combination of: graphical user interfaces; text messages; email messages; sounds; animations; and/or speech output.

[0588] Dynamically render data for different graphical user interface display engines based on the request. For example, use different output processing layouts and formats depending on which web browser and/or device is being used.

[0589] Render output data in different speech voices dynamically.

[0590] Dynamically render to specified modalities based on user preferences.

[0591] Dynamically render output using user-specific "skins" that customize the look and feel.

[0592] Send a stream of output packages to a modality, showing intermediate status, feedback, or results throughout phases of interaction with assistant **1002**.

[0593] According to specific embodiments, multiple instances or threads of output processor component(s) **1090** may be concurrently implemented and/or initiated via the use of one or more processor(s) **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of output processor component(s) **1090** may be performed, implemented and/or initiated by one or more of the following types of systems, components, systems, devices, procedures, processes, and the like (or combinations thereof):

[0594] software modules within the client or server of an embodiment of an intelligent automated assistant;

[0595] remotely callable services;

[0596] using a mix of templates and procedural code.

[0597] Referring now to FIG. **39**, there is shown a flow diagram depicting an example of a multiphase output procedure according to one embodiment. The multiphase output procedure includes automated assistant **1002** processing steps **702** and multiphase output steps **704**

[0598] In step **710**, a speech input utterance is obtained and a speech-to-text component (such as component described in connection with FIG. **22**) interprets the speech to produce a set of candidate speech interpretations **712**. In one embodiment, speech-to-text component is implemented using, for example, Nuance Recognizer, available from Nuance Communications, Inc. of Burlington, Mass. Candidate speech interpretations **712** may be shown to the user in **730**, for example in paraphrased form. For example, the interface might show "did you say?" alternatives listing a few possible alternative textual interpretations of the same speech sound sample.

[0599] In at least one embodiment, a user interface is provided to enable the user to interrupt and choose among the candidate speech interpretations.

[0600] In step **714**, the candidate speech interpretations **712** are sent to a language interpreter **1070**, which may produce representations of user intent **716** for at least one candidate speech interpretation **712**. In step **732**, paraphrases of these representations of user intent **716** are generated and presented to the user. (See related step **132** of procedure **120** in FIG. **22**).

[0601] In at least one embodiment, the user interface enables the user to interrupt and choose among the paraphrases of natural language interpretations **732**.

[0602] In step **718**, task and dialog analysis is performed. In step **734**, task and domain interpretations are presented to the user using an intent paraphrasing algorithm.

[0603] Referring now also to FIG. **40**, there is shown a screen shot depicting an example of output processing according to one embodiment. Screen **4001** includes echo **4002** of the user's speech input, generated by step **730**. Screen **4001** further includes paraphrase **4003** of the user's intent, generated by step **734**. In one embodiment, as depicted in the example of FIG. **40**, special formatting/highlighting is used for key words such as "events", which may be used to facilitate training of the user for interaction with intelligent automated assistant **1002**. For example, by visually observing the formatting of the displayed text, the user may readily identify and interpret back the intelligent automated assistant recognizes keywords such as "events", "next Wednesday", "San Francisco", and the like.

[0604] Returning to FIG. **39**, as requests are dispatched **720** to services and results are dynamically gathered, intermediate results may be displayed in the form of real-time progress **736**. For example, a list of restaurants may be returned and then their reviews may be populated dynamically as the results from the reviews services arrive. Services can include web-enabled services and/or services that access information stored locally on the device and/or from any other source.

[0605] A uniform representation of response **722** is generated and formatted **724** for the appropriate output modality. After the final output format is completed, a different kind of paraphrase may be offered in **738**. In this phase, the entire result set may be analyzed and compared against the initial request. A summary of results or answer to a question may then be offered.

[0606] Referring also to FIG. **41**, there is shown another example of output processing according to one embodiment. Screen **4101** depicts paraphrase **4102** of the text interpretation, generated by step **732**, real-time progress **4103** generated by step **736**, and paraphrased summary **7104** generated by step **738**. Also included are detailed results **4105**.

[0607] In one embodiment, assistant **1002** is capable of generating output in multiple modes. Referring now to FIG. **42**, there is shown a flow diagram depicting an example of multimodal output processing according to one embodiment.

[0608] The method begins **600**. Output processor **1090** takes uniform representation of response **490** and formats **612** the response according to the device and modality that is appropriate and applicable. Step **612** may include information from device and modality models **610** and/or domain data models **614**.

[0609] Once response **490** has been formatted **612**, any of a number of different output mechanisms can be used, in any combination. Examples depicted in FIG. **42** include:

[0610] Generating **620** text message output, which is sent **630** to a text message channel;

[0611] Generating **622** email output, which is sent **632** as an email message;

[0612] Generating **624** GUI output, which is sent **634** to a device or web browser for rendering;

[0613] Generating **626** speech output, which is sent **636** to a speech generation module.

[0614] One skilled in the art will recognize that many other output mechanisms can be used.

[0615] In one embodiment, the content of output messages generated by multiphase output procedure **700** is tailored to the mode of multimodal output processing **600**. For example, if the output modality is speech **626**, the language of used to paraphrase user input **730**, text interpretations **732**, task and domain interpretations **734**, progress **736**, and/or result summaries **738** may be more or less verbose or use sentences that are easier to comprehend in audible form than in written form. In one embodiment, the language is tailored in the steps of the multiphase output procedure **700**; in other embodiments, the multiphase output procedure **700** produces an intermediate result that is further refined into specific language by multimodal output processing **600**.

Short Term Personal Memory Component(s) **1052**

[0616] In at least one embodiment, short term personal memory component(s) **1052** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0617] Keep a history of the recent dialog between the embodiment of the assistant and the user, including the history of user inputs and their interpretations;

[0618] Keep a history of recent selections by the user in the GUI, such as which items were opened or explored, which phone numbers were called, which items were mapped, which movie trailers where played, and the like;

[0619] Store the history of the dialog and user interactions in a database on the client, the server in a user-specific session, or in client session state such as web browser cookies or RAM used by the client;

[0620] Store the list of recent user requests;

[0621] Store the sequence of results of recent user requests;

[0622] Store the click-stream history of UI events, including button presses, taps, gestures, voice activated triggers, and/or any other user input.

[0623] Store device sensor data (such as location, time, positional orientation, motion, light level, sound level, and the like) which might be correlated with interactions with the assistant.

[0624] According to specific embodiments, multiple instances or threads of short term personal memory component(s) **1052** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software.

[0625] According to different embodiments, one or more different threads or instances of short term personal memory component(s) **1052** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of short term personal memory component(s) **1052**. For example, short term personal memory component(s) **1052** may be invoked when there is a user session with the embodiment of assistant **1002**, on at least one input form or action by the user or response by the system.

[0626] In at least one embodiment, a given instance of short term personal memory component(s) **1052** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. For example, short term personal memory component(s) **1052** may access data from long-term personal memory components(s) **1054** (for example, to obtain user identity and personal preferences) and/or data from the local device about time and location, which may be included in short term memory entries.

[0627] Referring now to FIGS. **43**A and **43**B, there are shown screen shots depicting an example of the use of short term personal memory component(s) **1052** to maintain dialog context while changing location, according to one embodiment. In this example, the user has asked about the local weather, then just says "in new york". Screen **4301** shows the initial response, including local weather. When the user says "in new york", assistant **1002** uses short term personal memory component(s) **1052** to access the dialog context and thereby determine that the current domain is weather forecasts. This enables assistant **1002** to interpret the new utterance "in new york" to mean "what is the weather forecast in New York this coming Tuesday?". Screen **4302** shows the appropriate response, including weather forecasts for New York.

[0628] In the example of FIGS. **43**A and **43**B, what was stored in short term memory was not only the words of the input "is it going to rain the day after tomorrow?" but the system's semantic interpretation of the input as the weather domain and the time parameter set to the day after tomorrow.

Long-Term Personal Memory Component(s) **1054**

[0629] In at least one embodiment, long-term personal memory component(s) **1054** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0630] To persistently store the personal information and data about a user, including for example his or her preferences, identities, authentication credentials, accounts, addresses, and the like;

[0631] To store information that the user has collected by using the embodiment of assistant **1002**, such as the equivalent of bookmarks, favorites, clippings, and the like;

[0632] To persistently store saved lists of business entities including restaurants, hotels, stores, theaters and other venues. In one embodiment, long-term personal memory component(s) **1054** saves more than just the names or URLs, but also saves the information sufficient to bring up a full listing on the entities including phone numbers, locations on a map, photos, and the like;

[0633] To persistently store saved movies, videos, music, shows, and other items of entertainment;

[0634] To persistently store the user's personal calendar (s), to do list(s), reminders and alerts, contact databases, social network lists, and the like;

[0635] To persistently store shopping lists and wish lists for products and services, coupons and discount codes acquired, and the like;

[0636] To persistently store the history and receipts for transactions including reservations, purchases, tickets to events, and the like.

[0637] According to specific embodiments, multiple instances or threads of long-term personal memory component(s) **1054** may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. For example, in at least some embodiments, various aspects, features, and/or functionalities of long-term personal memory component(s) **1054** may be performed, implemented and/or initiated using one or more databases and/or files on (or associated with) clients **1304** and/or servers **1340**, and/or residing on storage devices.

[0638] According to different embodiments, one or more different threads or instances of long-term personal memory component(s) **1054** may be initiated in response to detection of one or more conditions or events satisfying one or more different types of minimum threshold criteria for triggering initiation of at least one instance of long-term personal memory component(s) **1054**. Various examples of conditions or events which may trigger initiation and/or implementation of one or more different threads or instances of long-term personal memory component(s) **1054** may include, but are not limited to, one or more of the following (or combinations thereof):

[0639] Long term personal memory entries may be acquired as a side effect of the user interacting with an embodiment of assistant **1002**. Any kind of interaction with the assistant may produce additions to the long term personal memory, including browsing, searching, finding, shopping, scheduling, purchasing, reserving, communicating with other people via an assistant.

[0640] Long term personal memory may also be accumulated as a consequence of users signing up for an account or service, enabling assistant **1002** access to accounts on other services, using an assistant **1002** service on a client device with access to other personal information databases such as calendars, to-do lists, contact lists, and the like.

[0641] In at least one embodiment, a given instance of long-term personal memory component(s) **1054** may access and/or utilize information from one or more associated databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices, which may be located, for example, at client(s) **1304** and/or server(s) **1340**. Examples of different types of data which may be accessed by long-term personal memory component(s) **1054** may include, but are not limited to data from other personal information databases such as contact or friend lists, calendars, to-do lists, other list managers, personal account and wallet managers provided by external services **1360**, and the like.

[0642] Referring now to FIGS. **44**A through **44**C, there are shown screen shots depicting an example of the use of long term personal memory component(s) **1054**, according to one embodiment. In the example, a feature is provided (named "My Stuff"), which includes access to saved entities such as restaurants, movies, and businesses that are found via interactive sessions with an embodiment of assistant **1002**. In screen **4401** of FIG. **44**A, the user has found a restaurant. The user taps on Save to My Stuff **4402**, which saves information about the restaurant in long-term personal memory component(s) **1054**.

[0643] Screen **4403** of FIG. **44**B depicts user access to My Stuff. In one embodiment, the user can select among categories to navigate to the desired item.

[0644] Screen **4404** of FIG. **44**C depicts the My Restaurant category, including items previously stored in My Stuff.

Automated Call and Response Procedure

[0645] Referring now to FIG. **33**, there is shown a flow diagram depicting an automatic call and response procedure, according to one embodiment. The procedure of FIG. **33** may be implemented in connection with one or more embodiments of intelligent automated assistant **1002**. It may be appreciated that intelligent automated assistant **1002** as depicted in FIG. **1** is merely one example from a wide range of intelligent automated assistant system embodiments which may be implemented. Other embodiments of intelligent automated assistant systems (not shown) may include additional, fewer and/or different components/features than those illustrated, for example, in the example intelligent automated assistant **1002** depicted in FIG. **1**.

[0646] In at least one embodiment, the automated call and response procedure of FIG. **33** may be operable to perform and/or implement various types of functions, operations, actions, and/or other features such as, for example, one or more of the following (or combinations thereof):

[0647] The automated call and response procedure of FIG. **33** may provide an interface control flow loop of a conversational interface between the user and intelligent

automated assistant **1002**. At least one iteration of the automated call and response procedure may serve as a ply in the conversation. A conversational interface is an interface in which the user and assistant **1002** communicate by making utterances back and forth in a conversational manner.

[0648] The automated call and response procedure of FIG. **33** may provide the executive control flow for intelligent automated assistant **1002**. That is, the procedure controls the gathering of input, processing of input, generation of output, and presentation of output to the user.

[0649] The automated call and response procedure of FIG. **33** may coordinate communications among components of intelligent automated assistant **1002**. That is, it may direct where the output of one component feeds into another, and where the overall input from the environment and action on the environment may occur.

[0650] In at least some embodiments, portions of the automated call and response procedure may also be implemented at other devices and/or systems of a computer network.

[0651] According to specific embodiments, multiple instances or threads of the automated call and response procedure may be concurrently implemented and/or initiated via the use of one or more processors **63** and/or other combinations of hardware and/or hardware and software. In at least one embodiment, one or more or selected portions of the automated call and response procedure may be implemented at one or more client(s) **1304**, at one or more server(s) **1340**, and/or combinations thereof.

[0652] For example, in at least some embodiments, various aspects, features, and/or functionalities of the automated call and response procedure may be performed, implemented and/or initiated by software components, network services, databases, and/or the like, or any combination thereof.

[0653] According to different embodiments, one or more different threads or instances of the automated call and response procedure may be initiated in response to detection of one or more conditions or events satisfying one or more different types of criteria (such as, for example, minimum threshold criteria) for triggering initiation of at least one instance of automated call and response procedure. Examples of various types of conditions or events which may trigger initiation and/or implementation of one or more different threads or instances of the automated call and response procedure may include, but are not limited to, one or more of the following (or combinations thereof):

[0654] a user session with an instance of intelligent automated assistant **1002**, such as, for example, but not limited to, one or more of:

[0655] a mobile device application starting up, for instance, a mobile device application that is implementing an embodiment of intelligent automated assistant **1002**;

[0656] a computer application starting up, for instance, an application that is implementing an embodiment of intelligent automated assistant **1002**;

[0657] a dedicated button on a mobile device pressed, such as a "speech input button";

[0658] a button on a peripheral device attached to a computer or mobile device, such as a headset, telephone handset or base station, a GPS navigation system, consumer appliance, remote control, or any other device with a button that might be associated with invoking assistance;

[0659] a web session started from a web browser to a website implementing intelligent automated assistant **1002**;

[0660] an interaction started from within an existing web browser session to a website implementing intelligent automated assistant **1002**, in which, for example, intelligent automated assistant **1002** service is requested;

[0661] an email message sent to a modality server **1426** that is mediating communication with an embodiment of intelligent automated assistant **1002**;

[0662] a text message is sent to a modality server **1426** that is mediating communication with an embodiment of intelligent automated assistant **1002**;

[0663] a phone call is made to a modality server **1434** that is mediating communication with an embodiment of intelligent automated assistant **1002**;

[0664] an event such as an alert or notification is sent to an application that is providing an embodiment of intelligent automated assistant **1002**.

[0665] when a device that provides intelligent automated assistant **1002** is turned on and/or started.

[0666] According to different embodiments, one or more different threads or instances of the automated call and response procedure may be initiated and/or implemented manually, automatically, statically, dynamically, concurrently, and/or combinations thereof. Additionally, different instances and/or embodiments of the automated call and response procedure may be initiated at one or more different time intervals (e.g., during a specific time interval, at regular periodic intervals, at irregular periodic intervals, upon demand, and the like).

[0667] In at least one embodiment, a given instance of the automated call and response procedure may utilize and/or generate various different types of data and/or other types of information when performing specific tasks and/or operations. This may include, for example, input data/information and/or output data/information. For example, in at least one embodiment, at least one instance of the automated call and response procedure may access, process, and/or otherwise utilize information from one or more different types of sources, such as, for example, one or more databases. In at least one embodiment, at least a portion of the database information may be accessed via communication with one or more local and/or remote memory devices. Additionally, at least one instance of the automated call and response procedure may generate one or more different types of output data/information, which, for example, may be stored in local memory and/or remote memory devices.

[0668] In at least one embodiment, initial configuration of a given instance of the automated call and response procedure may be performed using one or more different types of initialization parameters. In at least one embodiment, at least a portion of the initialization parameters may be accessed via communication with one or more local and/or remote memory devices. In at least one embodiment, at least a portion of the initialization parameters provided to an instance of the automated call and response procedure may correspond to and/or may be derived from the input data/information.

[0669] In the particular example of FIG. **33**, it is assumed that a single user is accessing an instance of intelligent automated assistant **1002** over a network from a client application with speech input capabilities. The user is interested in find-

ing a good place for dinner at a restaurant, and is engaging intelligent automated assistant **1002** in a conversation to help provide this service.

[0670] In step **100**, the user is prompted to enter a request. The user interface of the client offers several modes of input, as described in connection with FIG. **26**. These may include, for example:

[0671] an interface for typed input, which may invoke an active typed-input elicitation procedure as illustrated in FIG. **11**;

[0672] an interface for speech input, which may invoke an active speech input elicitation procedure as illustrated in FIG. **22**.

[0673] an interface for selecting inputs from a menu, which may invoke active GUI-based input elicitation as illustrated in FIG. **23**.

[0674] One skilled in the art will recognize that other input modes may be provided.

[0675] In one embodiment, step **100** may include presenting options remaining from a previous conversation with assistant **1002**, for example using the techniques described in the active dialog suggestion input elicitation procedure described in connection with FIG. **24**.

[0676] For example, by one of the methods of active input elicitation in step **100**, the user might say to assistant **1002**, "where may I get some good Italian around here?" For example, the user might have spoken this into a speech input component. An embodiment of an active input elicitation component **1094** calls a speech-to-text service, asks the user for confirmation, and then represents the confirmed user input as a uniform annotated input format **2690**.

[0677] An embodiment of language interpreter component **1070** is then called in step **200**, as described in connection with FIG. **29**. Language interpreter component **1070** parses the text input and generates a list of possible interpretations of the user's intent **290**. In one parse, the word "italian" is associated with restaurants of style Italian; "good" is associated with the recommendation property of restaurants; and "around here" is associated with a location parameter describing a distance from a global sensor reading (for example, the user's location as given by GPS on a mobile device).

[0678] In step **300**, the representation of the user's intent **290** is passed to dialog flow processor **1080**, which implements an embodiment of a dialog and flow analysis procedure as described in connection with FIG. **32**. Dialog flow processor **1080** determines which interpretation of intent is most likely, maps this interpretation to instances of domain models and parameters of a task model, and determines the next flow step in a dialog flow. In the current example, a restaurant domain model is instantiated with a constrained selection task to find a restaurant by constraints (the cuisine style, recommendation level, and proximity constraints). The dialog flow model indicates that the next step is to get some examples of restaurants meeting these constraints and present them to the user.

[0679] In step **400**, an embodiment of the flow and service orchestration procedure **400** is invoked, via services orchestration component **1082**. It invokes a set of services **1084** on behalf of the user's request to find a restaurant. In one embodiment, these services **1084** contribute some data to a common result. Their data are merged and the resulting list of restaurants is represented in a uniform, service-independent form.

[0680] In step **500**, output processor **1092** generates a dialog summary of the results, such as, "I found some recommended Italian restaurants near here." Output processor **1092** combines this summary with the output result data, and then sends the combination to a module that formats the output for the user's particular mobile device in step **600**.

[0681] In step **700**, this device-specific output package is sent to the mobile device, and the client software on the device renders it on the screen (or other output device) of the mobile device.

[0682] The user browses this presentation, and decides to explore different options. If the user is done **790**, the method ends. If the user is not done **490**, another iteration of the loop is initiated by returning to step **100**.

[0683] The automatic call and response procedure may be applied, for example to a user's query "how about mexican food?". Such input may be elicited in step **100**. In step **200**, the input is interpreted as restaurants of style Mexican, and combined with the other state (held in short term personal memory **1052**) to support the interpretation of the same intent as the last time, with one change in the restaurant style parameter. In step **300**, this updated intent produces a refinement of the request, which is given to service orchestration component(s) **1082** in step **400**.

[0684] In step **400** the updated request is dispatched to multiple services **1084**, resulting in a new set of restaurants which are summarized in dialog in **500**, formatted for the device in **600**, and sent over the network to show new information on the user's mobile device in step **700**.

[0685] In this case, the user finds a restaurant of his or her liking, shows it on a map, and sends directions to a friend.

[0686] One skilled in the art will recognize that different embodiments of the automated call and response procedure (not shown) may include additional features and/or operations than those illustrated in the specific embodiment of FIG. **33**, and/or may omit at least a portion of the features and/or operations of automated call and response procedure illustrated in the specific embodiment of FIG. **33**.

Constrained Selection

[0687] In one embodiment, intelligent automated assistant **1002** uses constrained selection in its interactions with the user, so as to more effectively identify and present items that are likely to be of interest to the user.

[0688] Constrained selection is a kind of generic task. Generic tasks are abstractions that characterize the kinds of domain objects, inputs, outputs, and control flow that are common among a class of tasks. A constrained selection task is performed by selecting items from a choice set of domain objects (such as restaurants) based on selection constraints (such as a desired cuisine or location). In one embodiment, assistant **1002** helps the user explore the space of possible choices, eliciting the user's constraints and preferences, presenting choices, and offering actions to perform on those choices such as to reserve, buy, remember, or share them. The task is complete when the user selects one or more items on which to perform the action.

[0689] Constrained selection is useful in many contexts: for example, picking a movie to see, a restaurant for dinner, a hotel for the night, a place to buy a book, or the like. In general, constrained selection is useful when one knows the category and needs to select an instance of the category with some desired properties.

[0690] One conventional approach to constrained selection is a directory service. The user picks a category and the system offers a list of choices. In a local directory, one may constrain the directory to a location, such as a city. For instance, in a "yellow pages" service, users select the book for a city and then look up the category, and the book shows one or more items for that category. The main problem with a directory service is that the number of possibly relevant choices is large (e.g., restaurants in a given city).

[0691] Another conventional approach is a database application, which provides a way to generate a choice set by eliciting a query from the user, retrieving matching items, and presenting the items in some way that highlights salient features. The user browses the rows and columns of the result set, possibly sorting the results or changing the query until he or she finds some suitable candidates. The problem with the database service is that it may require the user to operationalize their human need as a formal query and to use the abstract machinery of sort, filter, and browse to explore the resulting data. These are difficult for most people to do, even with graphical user interfaces.

[0692] A third conventional approach is open-ended search, such as "local search". Search is easy to do, but there are several problems with search services that make them difficult for people to accomplish the task of constrained selection. Specifically:

[0693] As with directory search, the user may not just enter a category and look at one or more possible choice, but must narrow down the list.

[0694] If the user can narrow the selection by constraints, it is not obvious what constraints may be used (e.g., may I search for places that are within walking distance or are open late?)

[0695] It is not clear how to state constraints (e.g., is it called cuisine or restaurant type, and what are the possible values?)

[0696] Multiple preferences conflict; there is usually no objectively "best" answer to a given situation (e.g., I want a place that is close by and cheap serving gourmet food with excellent service and which is open until midnight).

[0697] Preferences are relative, and they depend on what is available. For example, if the user may get a table at a highly rated restaurant, he or she might choose it even though it is expensive. In general, though, the user would prefer less expensive options.

[0698] In various embodiments, assistant **1002** of the present invention helps streamline the task of constrained selection. In various embodiments, assistant **1002** employs database and search services, as well as other functionality, to reduce the effort, on the part of the user, of stating what he or she is looking for, considering what is available, and deciding on a satisfactory solution.

[0699] In various embodiments, assistant **1002** helps to make constrained selection simpler for humans in any of a number of different ways.

[0700] For example, in one embodiment, assistant **1002** may operationalize properties into constraints. The user states what he or she wants in terms of properties of the desired outcome. Assistant **1002** operationalizes this input into formal constraints. For example, instead of saying "find one or more restaurants less than 2 miles from the center of Palo Alto whose cuisine includes Italian food" the user may just say "Italian restaurants in palo alto". Assistant **1002** may also operationalize qualities requested by the user that are not parameters to a database. For example, if the user requests romantic restaurants, the system may operationalize this as a text search or tag matching constraint. In this manner, assistant **1002** helps overcome some of the problems users may otherwise have with constrained selection. It is easier, for a user, to imagine and describe a satisfactory solution than to describe conditions that would distinguish suitable from unsuitable solutions.

[0701] In one embodiment, assistant **1002** may suggest useful selection criteria, and the user need only say which criteria are important at the moment. For example, assistant **1002** may ask "which of these matter: price (cheaper is better), location (closer is better), rating (higher rated is better)?" Assistant **1002** may also suggest criteria that may require specific values; for example, "you can say what kind of cuisine you would like or a food item you would like".

[0702] In one embodiment, assistant **1002** may help the user make a decision among choices that differ on a number of competing criteria (for example, price, quality, availability, and convenience).

[0703] By providing such guidance, assistant **1002** may help users in making multiparametric decisions in any of several ways:

[0704] One is to reduce the dimensionality of the space, combining raw data such as ratings from multiple sources into a composite "recommendation" score. The composite score may take into account domain knowledge about the sources of data (e.g., Zagat ratings may be more predictive of quality than Yelp).

[0705] Another approach is to focus on a subset of criteria, turning a problem of "what are all the possible criteria to consider and how to they combine?" into a selection of the most important criteria in a given situation (e.g., "which is more important, price or proximity?").

[0706] Another way to simply the decision making is to assume default values and preference orders (e.g., all things being equal, higher rated and closer and cheaper are better). The system may also remember users' previous responses that indicate their default values and preferences.

[0707] Fourth, the system may offer salient properties of items in the choice set that were not mentioned in the original request. For example, the user may have asked for local Italian food. The system may offer a choice set of restaurants, and with them, a list of popular tags used by reviewers or a tag line from a guide book (e.g., "a nice spot for a date" "great pasta"). This could let people pick out a specific item and complete the task. Research shows that most people make decisions by evaluating specific instances rather than deciding on criteria and rationally accepting the one that pops to the top. It also shows that people learn about features from concrete cases. For example, when choosing among cars, buyers may not care about navigation systems until they see that some of the cars have them (and then the navigation system may become an important criterion). Assistant **1002** may present salient properties of listed items that help people pick a winner or that suggest a dimension along which to optimize.

Conceptual Data Model

[0708] In one embodiment, assistant **1002** offers assistance with the constrained selection task by simplifying the con-

ceptual data model. The conceptual data model is the abstraction presented to users in the interface of assistant **1002**. To overcome the psychological problems described above, in one embodiment assistant **1002** provides a model that allows users to describe what they want in terms of a few easily recognized and recalled properties of suitable choices rather than constraint expressions. In this manner, properties can be made easy to compose in natural language requests (e.g., adjectives modifying keyword markers) and be recognizable in prompts ("you may also favor recommended restaurants . . . "). In one embodiment, a data model is used that allows assistant **1002** to determine the domain of interest (e.g., restaurants versus hotels) and a general approach to guidance that may be instantiated with domain-specific properties.

[0709] In one embodiment, the conceptual data model used by assistant **1002** includes a selection class. This is a representation of the space of things from which to choose. For example, in the find-a-restaurant application, the selection class is the class of restaurants. The selection class may be abstract and have subclasses, such as "things to do while in a destination". In one embodiment, the conceptual data model assumes that, in a given problem solving situation, the user is interested in choosing from a single selection class. This assumption simplifies the interaction and also allows assistant **1002** to declare its boundaries of competence ("I know about restaurants, hotels, and movies" as opposed to "I know about life in the city").

[0710] Given a selection class, in one embodiment the data model presented to the user for the constrained selection task includes, for example: items; item features; selection criteria; and constraints.

[0711] Items are instances of the selection class.

[0712] Item features are properties, attributes, or computed values that may be presented and/or associated with at least one item. For example, the name and phone number of a restaurant are item features. Features may be intrinsic (the name or cuisine of a restaurant) or relational (e.g., the distance from one's current location of interest). They may be static (e.g., restaurant name) or dynamic (rating). They may be composite values computed from other data (e.g., a "value for money" score). Item features are abstractions for the user made by the domain modeler; they do not need to correspond to underlying data from backend services.

[0713] Selection criteria are item features that may be used to compare the value or relevance of items. That is, they are ways to say which items are preferred. Selection criteria are modeled as features of the items themselves, whether they are intrinsic properties or computed. For example, proximity (defined as distance from the location of interest) is a selection criterion. Location in space-time is a property, not a selection criterion, and it is used along with the location of interest to compute the distance from the location of interest.

[0714] Selection criteria may have an inherent preference order. That is, the values of any particular criterion may be used to line up items in a best first order. For example, the proximity criterion has an inherent preference that closer is better. Location, on the other hand, has no inherent preference value. This restriction allows the system to make default assumptions and guide the selection if the user only mentions the criterion. For example, the user interface might offer to "sort by rating" and assume that higher rated is better.

[0715] One or more selection criteria are also item features; they are those features related to choosing among possible items. However, item features are not necessarily related to a preference (e.g., the names and phone numbers of restaurants are usually irrelevant to choosing among them).

[0716] In at least one embodiment, constraints are restrictions on the desired values of the selection criteria. Formally, constraints might be represented as set membership (e.g., cuisine type includes Italian), pattern matches (e.g., restaurant review text includes "romantic"), fuzzy inequalities (e.g., distance less than a few miles), qualitative thresholds (e.g., highly rated), or more complex functions (e.g., a good value for money). To make things simple enough for normal humans, this data model reduces at least one or more constraints to symbolic values that may be matched as words. Time and distance may be excluded from this reduction. In one embodiment, the operators and threshold values used for implementing constraints are hidden from the user. For example, a constraint on the selection criteria called "cuisine" may be represented as a symbolic value such as "Italian" or "Chinese". A constraint on rating is "recommended" (a binary choice). For time and distance, in one embodiment assistant **1002** uses proprietary representations that handle a range of inputs and constraint values. For example, distance might be "walking distance" and time might be "tonight"; in one embodiment, assistant **1002** uses special processing to match such input to more precise data.

[0717] In at least one embodiment, some constraints may be required constraints. This means that the task simply cannot be completed without this data. For example, it is hard to pick a restaurant without some notion of desired location, even if one knows the name.

[0718] To summarize, a domain is modeled as selection classes with item features that are important to users. Some of the features are used to select and order items offered to the user—these features are called selection criteria. Constraints are symbolic limits on the selection criteria that narrow the set of items to those that match.

[0719] Often, multiple criteria may compete and constraints may match partially. The data model reduces the selection problem from an optimization (finding the best solution) to a matching problem (finding items that do well on a set of specified criteria and match a set of symbolic constraints). The algorithms for selecting criteria and constraints and determining an ordering are described in the next section.

Methodology for Constrained Selection

[0720] In one embodiment, assistant **1002** performs constrained selection by taking as input an ordered list of criteria, with implicit or explicit constraints on at least one, and generating a set of candidate items with salient features. Computationally, the selection task may be characterized as a nested search: first, identify a selection class, then identify the important selection criteria, then specify constraints (the boundaries of acceptable solutions), and search through instances in order of best-fit to find acceptable items.

[0721] Referring now to FIG. **45**, there is shown an example of an abstract model **4500** for a constrained selection task as a nested search. In the example assistant **1002** identifies **4505** a selection call among all local search types **4501**. The identified class is restaurant. Within the set of all restaurants **4502**, assistant **1002** selects **4506** criteria. In the example, the criterion is identified as distance. Within the set of restaurants in PA **4503**, assistant **1002** specifies **4507** constraints for the search. In the example, the identified constraint is "Italian cuisine"). Within the set of Italian restaurants in PA **4504**, assistant **4508** selects items for presentation to the user.

[0722] In one embodiment, such a nested search is what assistant **1002** does once it has the relevant input data, rather than the flow for eliciting the data and presenting results. In one embodiment, such control flow is governed via a dialog between assistant **1002** and the user which operates by other procedures, such as dialog and task flow models. Constrained selection offers a framework for building dialog and task flow models at this level of abstraction (that is, suitable for constrained selection tasks regardless of domain).

[0723] Referring now to FIG. **46**, there is shown an example of a dialog **4600** to help guide the user through a search process, so that the relevant input data can be obtained.

[0724] In the example dialog **4600**, the first step is for the user to state the kind of thing they are looking for, which is the selection class. For example, the user might do this by saying "dining in palo alto". This allows assistant **1002** to infer **4601** the task and domain.

[0725] Once assistant **1002** has understood the task and domain binding (selection class=restaurants), the next step is to understand which selection criteria are important to this user, for example by soliciting **4603** criteria and/or constraints. In the example above, "in palo alto" indicates a location of interest. In the context of restaurants, the system may interpret a location as a proximity constraint (technically, a constraint on the proximity criterion). Assistant **1002** explains what is needed, receives input. If there is enough information to constrain the choice set to a reasonable size, then assistant **1002** paraphrases the input and presents **4605** one or more restaurants that meet the proximity constraint, sorted in some useful order. The user can then select **4607** from this list, or refine **4606** the criteria and constraints. Assistant **1002** reasons about the constraints already stated, and uses domain-specific knowledge to suggest other criteria that might help, soliciting constraints on these criteria as well. For example, assistant **1002** may reason that, when recommending restaurants within walking distance of a hotel, the useful criteria to solicit would be cuisine and table availability.

[0726] The constrained selection task is complete when the user selects **4607** an instance of the selection class. In one embodiment, additional follow-on tasks **4602** are enabled by assistant **1002**. Thus, assistant **1002** can offer services that indicate selection while providing some other value. Examples **4608** booking a restaurant, setting a reminder on a calendar, and/or sharing the selection with others by sending an invitation. For example, booking a restaurant certainly indicates that it was selected; other options might be to put the restaurant on a calendar or send in invitation with directions to friends.

[0727] Referring now to FIG. **47**, there is shown a flow diagram depicting a method of constrained selection according to one embodiment. In one embodiment, assistant **1002** operates in an opportunistic and mixed-initiative manner, permitting the user to jump to the inner loop, for instance, by stating task, domain, criteria, and constraints one or more at once in the input.

[0728] The method begins **4701**. Input is received **4702** from the user, according to any of the modes described herein. If, based on the input, the task not known, assistant **1002** requests **4705** clarifying input from the user.

[0729] In step **4717**, assistant **1002** determines whether the user provides additional input. If so, assistant **1002** returns to step **4702**. Otherwise the method ends **4799**.

[0730] If, in step **4703**, the task is known, assistant **1002** determines **4704** whether the task is constrained selection. If not, assistant **1002** proceeds **4706** to the specified task flow.

[0731] If, in step **4704**, the task is constrained selection, assistant **1002** determines **4707** whether the selection class can be determined. If not, assistant **1002** offers **4708** a choice of known selection classes, and returns to step **4717**.

[0732] If, in step **4707**, the selection class can be determined, assistant **1002** determines **4709** whether all required constraints can be determined. If not, assistant **1002** prompts **4710** for required information, and returns to step **4717**.

[0733] If, in step **4709**, all required constants can be determined, assistant **1002** determines **4711** whether any result items can be found, given the constraints. If there are no items that meet the constraints, assistant **1002** offers **4712** ways to relax the constraints. For example, assistant **1002** may relax the constraints from lowest to highest precedence, using a filter/sort algorithm. In one embodiment, if there are items that meet some of the constraints, then assistant **1002** may paraphrase the situation (outputting, for example, "I could not find Recommended Greek restaurants that deliver on Sundays in San Carlos. However, I found 3 Greek restaurants and 7 Recommend restaurants in San Carlos."). In one embodiment, if there are no items that match any constraints, then assistant **1002** may paraphrase this situation and prompt for different constraints (outputting, for example, "Sorry, I could not find any restaurants in Anytown, Texas. You may pick a different location."). Assistant **1002** returns to step **4717**.

[0734] If, in step **4711**, result items can be found, assistant **1002** offers **4713** a list of items. In one embodiment, assistant **1002** paraphrases the currently specified criteria and constraints (outputting, for example, "Here are some recommended Italian restaurants in San Jose." (recommended=yes, cuisine=Italian, proximity=<in San Jose>)). In one embodiment, assistant **1002** presents a sorted, paginated list of items that meet the known constraints. If an item only shows some of the constraints, such a condition can be shown as part of the item display. In one embodiment, assistant **1002** offers the user ways to select an item, for example by initiating another task on that item such as booking, remembering, scheduling, or sharing. In one embodiment, on any given item, assistant **1002** presents item features that are salient for picking instances of the selection class. In one embodiment, assistant **1002** shows how the item meets a constraint; for example, Zagat rating of 5 meets the Recommended=yes constraint, and "1 mile away" meets the "within walking distance of an address" constraint. In one embodiment, assistant **1002** allows the user to drill down for more detail on an item, which results in display of more item features.

[0735] Assistant **1002** determines **4714** whether the user has selected an item. If the user selects an item, the task is complete. Any follow-on task is performed **4715**, if there is one, and the method ends **4799**.

[0736] If, in step **4714**, the user does not select an item, assistant **1002** offers **4716** the user ways to select other criteria and constraints and returns to step **4717**. For example, given the currently specified criteria and constraints, assistant **1002** may offer criteria that are most likely to constrain the choice set to a desired size. If the user selects a constraint value, that constraint value is added to the previously determined constraints when steps **4703** to **4713** are repeated.

[0737] Since one or more criteria may have an inherent preference value, selecting the criteria may add information to the request. For example, allowing the user to indicate that

positive reviews are valued allows assistant **1002** to sort by this criterion. Such information can be taken into account when steps **4703** to **4713** are repeated.

[0738] In one embodiment, assistant **1002** allows the user to raise the importance of a criterion that is already specified, so that it would be higher in the precedence order. For example, if the user asked for fast, cheap, highly recommended restaurants within one block of their location, assistant **1002** may request that the user chooses which of these criteria are more important. Such information can be taken into account when steps **4703** to **4713** are repeated.

[0739] In one embodiment, the user can provide additional input at any point while the method of FIG. **47** is being performed. In one embodiment, assistant **1002** checks periodically or continuously for such input, and, in response, loops back to step **4703** to process it.

[0740] In one embodiment, when outputting an item or list of items, assistant **1002** indicates, in the presentation of items, the features that were used to select and order them. For example, if the user asked for nearby Italian restaurants, such item features for distance and cuisine may be shown in the presentation of the item. This may include highlighting matches, as well as listing selection criteria that were involved in the presentation of an item.

Example Domains

[0741] Table 1 provides an example of constrained selection domains that may be handled by assistant **1002** according to various embodiments.

TABLE 1

	Based on these criteria								
Select a	Location	Price	Availability	Type	Quality	Name	Services	special search	general search
Restaurant	proximity	affordability	open tables	cuisine	rating by guide, review	restaurant name	delivery	menu items	keywords
Hotel	proximity	price range	available rooms	motel, hotel, B&B, . . .	rating by guide, review	hotel name	amenities		keywords
Movie	theatre proximity		show times	genre	rating by review	movie title		actors, etc	
Local Business	proximity			business category	rating by review	business name			keywords
Local event	venue proximity		by date			event title			keywords
concert	venue proximity		by tour schedule	music genre		band name		band members	keywords
CD, book, DVD to buy		price range	online, in store, etc	download, physical	popularity	album or song name		artist, title, etc.	keywords

Filtering and Sorting Results

[0742] In one embodiment, when presenting items that meet currently specified criteria and constraints, a filter/sort methodology can be employed. In one embodiment selection constraints may serve as both filter and sort parameters to the underlying services. Thus, any selection criterion can be used to determine which items are in the list, and to compute the order in which to paginate and show them. Sort order for this task is akin to relevance rank in search. For example, proximity is a criterion with symbolic constraint values such as "within driving distance" and a general notion of sorting by distance. The "driving distance" constraint might be used to select a group of candidate items. Within that group, closer items might be sorted higher in the list.

[0743] In one embodiment, selection constraints and associated filtering and sorting are at discrete "levels", which are functions of both the underlying data and the input from the user. For example, proximity is grouped into levels such as "walking distance", "taxi distance", "driving distance". When sorting, one or more items within walking distance are treated as if they were the same distance. The input from the user may come into play in the way he or she specifies a constraint. If the user enters "in palo alto", for example, then one or more items within the Palo Alto city limits are perfect matches and are equivalent. If the user enters, "near the University Avenue train station" then the match would depend on a distance from that address, with the degree of match dependent on the selection class (e.g., near for restaurants is different than near for hotels). Even within a constraint that may be specified with a continuous value, a discretization may be applied. This may be important for sorting operations, so that multiple criteria may participate in determining the best-first ordering.

[0744] In one embodiment, the item list—those items that are considered "matching" or "good enough"—may be shorter or longer than the number of items shown on one "page" of the output. Generally, items in the first page are given the most attention, but conceptually there is a longer list, and pagination is simply a function of the form factor of the output medium. This means, for instance, that if the user is offered a way to sort or browse the items by some criterion, then it is the entire set of items (more than one page worth) that is sorted or browsed.

[0745] In one embodiment, there is a precedence ordering among selection criteria. That is, some criteria may matter more than others in the filter and sort. In one embodiment, those criteria selected by the user are given higher precedence than others, and there is a default ordering over one or more criteria. This allows for a general lexicographic sort. The assumption is that there is a meaningful a priori precedence. For example, unless the user states otherwise, it may be more important for a restaurant to be close than to be inexpensive. In one embodiment, the a priori precedence ordering is

domain-specific. The model allows for user-specific preferences to override the domain defaults, if that is desired.

[0746] Since the values of constraints can represent several internal data types, there are different ways for constraints to match, and they may be specific to the constraint. For example, in one embodiment:

[0747] Binary constraints match one or more or none. For example, whether a restaurant is "Fast" might be either true or not.

[0748] Set membership constraints match one or more or none based on a property value. For example, cuisine=Greek means the set of cuisines for a restaurant includes Greek.

[0749] Enumeration constraints match at a threshold. For example, a rating criterion might have constraint values rated, highly-rated, or top-rated. Constraining to highly-rated would also match top-rated.

[0750] Numeric constraints match at a threshold that may be criterion specific. For example, "open late" might be a criterion, and the user might ask for places open after 10:00 pm. This kind of constraint may be slightly out of scope for the constrained selection task, since it is not a symbolic constraint value. However, in one embodiment, assistant **1002** recognizes some cases of numeric constraints like this, and maps them to threshold values with symbolic constraints (e.g., "restaurants in palo alto open now"->"here are 2 restaurants in palo alto that are open late").

[0751] Location and time are handled specially. A constraint on proximity might be a location of interest specified at some level of granularity, and that determines the match. If the user specifies a city, then city-level matching is appropriate; a ZIP code may allow for a radius. Assistant **1002** may also understand locations that are "near" other locations of interest, also based on special processing. Time is relevant as a constraint value of criteria that have threshold value based on a service call, such as table availability or flights within a given time range.

[0752] In one embodiment, constraints can be modeled so that there is a single threshold value for selection and a small set of discrete values for sorting. For example, the affordability criterion might be modeled as a roughly binary constraint, where affordable restaurants are any under some threshold price range. When the data justify multiple discrete levels for selection, constraints can be modeled using a gradient of matching. In one embodiment two levels of matching (such as strong and weak matching) may be provided; however, one skilled in the art will recognize that in other embodiments, any number of levels of matching can be provided. For example, proximity may be matched with a fuzzy boundary, so that things that are near the location of interest may match weakly. The operational consequence of a strong or weak match is in the filter/sort algorithm as described below.

[0753] For at least one criterion, an approach to matching and default thresholds can be established, if relevant. The user may be able to say just the name of the constraint, a symbolic constraint value, or a precise constraint expression if it is handled specially (such as time and location).

[0754] An ideal situation for constrained selection occurs when the user states constraints that result in a short list of candidates, one or more of which meet the constraints. The user then chooses among winners based on item features. In many cases, however, the problem is over- or under-constrained. When it is over-constrained, there are few or no items that meet the constraints. When it is under-constrained, there are so many candidates that examining the list is not expedient. In one embodiment, the general constrained selection model of the present invention is able to handle multiple constraints with robust matching and usually produce something to choose from. Then the user may elect to refine their criteria and constraints or just complete the task with a "good enough" solution.

Method

[0755] In one embodiment, the following method is used for filtering and sorting results:

[0756] 1. Given an ordered list of selection criteria selected by the user, determine constraints on at least one.

[0757] a. If the user specified a constraint value, use it. For example, if the user said "greek food" the constraint is cuisine=Greek. If the user said "san Francisco" the constraint is In the City of San Francisco. If the user said "south of market" then the constraint is In the Neighborhood of SoMa.

[0758] b. Otherwise use a domain- and criteria-specific default. For example, if the user said "a table at some that place" he or she is indicating that the availability criterion is relevant, but he or she did not specify a constraint value. The default constraint values for availability might be some range of date times such as tonight and a default party size of 2.

[0759] 2. Select a minimum of N results by specified constraints.

[0760] a. Try to get N results at strong match.

[0761] b. If that fails, try to relax constraints, in reverse precedence order. That is, match at strong level for one or more of the criteria except the last, which may match at a weak level. If there is no weak match for that constraint, then try weak matches up the line from lowest to highest precedence.

[0762] c. Then repeat the loop allowing failure to match on constraints, from lowest to highest precedence.

[0763] 3. After getting a minimum choice set, sort lexicographically over one or more criteria (which may include user-specified criteria as well as other criteria) in precedence order.

[0764] a. Consider the set of user-specified criteria as highest precedence, then one or more remaining criteria in their a priori precedence. For example, if the a priori precedence is (availability, cuisine, proximity, rating), and the user gives constraints on proximity and cuisine, then the sort precedence is (cuisine, proximity, availability, rating).

[0765] b. Sort on criteria using discrete match levels (strong, weak, none), using the same approach as in relaxing constraints, this time applied the full criteria list.

[0766] i. If a choice set was obtained without relaxing constraints, then one or more of the choice set may "tie" in the sort because they one or more match at strong levels. Then, the next criteria in the precedence list may kick in to sort them. For example, if the user says cuisine=Italian, proximity=In San Francisco, and the sort precedence is (cuisine, proximity, availability, rating),

then one or more the places on the list have equal match values for cuisine and proximity. So the list would be sorted on availability (places with tables available bubble to the top). Within the available places, the highest rated ones would be at the top.

[0767] ii. If the choice set was obtained by relaxing constraints, then one or more of the fully matching items are at the top of the list, then the partially matching items. Within the matching group, they are sorted by the remaining criteria, and the same for the partially matching group. For example, if there were only two Italian restaurants in San Francisco, then the available one would be shown first, then the unavailable one. Then the rest of the restaurants in San Francisco would be shown, sorted by availability and rating.

Precedence Ordering

[0768] The techniques described herein allow assistant **1002** to be extremely robust in the face of partially specified constraints and incomplete data. In one embodiment, assistant **1002** uses these techniques to generate a user list of items in best-first order, i.e. according to relevance.

[0769] In one embodiment, such relevance sorting is based on an a priori precedence ordering. That is, of the things that matter about a domain, a set of criteria is chosen and placed in order of importance. One or more things being equal, criteria higher in the precedence order may be more relevant to a constrained selection among items than those lower in the order. Assistant **1002** may operate on any number of criteria. In addition, criteria may be modified over time without breaking existing behaviors.

[0770] In one embodiment, the precedence order among criteria may be tuned with domain-specific parameters, since the way criteria interact may depend on the selection class. For example, when selecting among hotels, availability and price may be dominant constraints, whereas for restaurants, cuisine and proximity may be more important.

[0771] In one embodiment, the user may override the default criteria ordering in the dialog. This allows the system to guide the user when searches are over-constrained, by using the ordering to determine which constraints should be relaxed. For example, if the user gave constraints on cuisine, proximity, recommendation, and food item, and there were no fully matching items, the user could say that food item was more important than recommendation level and change the mix so the desired food item matches were sorted to the top.

[0772] In one embodiment, when precedence order is determined, user-specified constraints take precedence over others. For example, in one embodiment, proximity is a required constraint and so is always specified, and further has precedence over other unselected constraints. Therefore it does not have to be the highest precedence constraint in order to be fairly dominant. Also, many criteria may not match at one or more unless a constraint is given by the user, and so the precedence of these criteria only matters within user-selected criteria. For example, when the user specifies a cuisine it is important to them, and otherwise is not relevant to sorting items.

[0773] For example, the following is a candidate precedence sorting paradigm for the restaurant domain:

[0774] 1. cuisine* (not sortable unless a constraint value is given)

[0775] 2. availability* (sortable using a default constraint value, e.g., time)

[0776] 3. recommended

[0777] 4. proximity* (a constraint value is always given)

[0778] 5. affordability

[0779] 6. may deliver

[0780] 7. food item (not sortable unless a constraint value, e.g., a keyword, is given)

[0781] 8. keywords (not sortable unless a constraint value, e.g., a keyword, is given)

[0782] 9. restaurant name

[0783] The following is an example of a design rationale for the above sorting paradigm:

[0784] If a user specifies a cuisine, he or she wants it to stick.

[0785] One or more things being equal, sort by rating level (it is the highest precedence among criteria than may be used to sort without a constraint).

[0786] In at least one embodiment, proximity may be more important than most things. However, since it matches at discrete levels (in a city, within a radius for walking and the like), and it is always specified, then most of the time most matching items may "tie" on proximity.

[0787] Availability (as determined by a search on a website such as opentable.com, for instance) is a valuable sort criterion, and may be based on a default value for sorting when not specified. If the user indicates a time for booking, then only available places may be in the list and the sort may be based on recommendation.

[0788] If the user says they want highly recommended places, then it may sort above proximity and availability, and these criteria may be relaxed before recommendation. The assumption is that if someone is looking for nice place, they may be willing to drive a bit farther and it is more important than a default table availability. If a specific time for availability is specified, and the user requests recommended places, then places that are both recommended and available may come first, and recommendation may relax to a weak match before availability fails to match at one or more.

[0789] The remaining constraints except for name are one or more based on incomplete data or matching. So they are weak sort heuristics by default, and when they are specified the match one or more-or-none.

[0790] Name may be used as a constraint to handle the case where someone mentions the restaurant by name, e.g., find one or more Hobee's restaurants near Palo Alto. In this case, one or more items may match the name, and may be sorted by proximity (the other specified constraint in this example).

Domain Modeling: Mapping Selection Criteria to Underlying Data

[0791] It may be desirable to distinguish between the data that are available for computation by assistant **1002** and the data used for making selections. In one embodiment, assistant **1002** uses a data model that reduces the complexity for the user by folding one or more kinds of data used to distinguish among items into a simple selection criteria model. Internally, these data may take several forms. Instances of the selection class can have intrinsic properties and attributes (such as cuisine of a restaurant), may be compared along dimensions (such as the distance from some location), and may be dis-

covered by some query (such as whether it matches a text pattern or is available at a given time). They may also be computed from other data which are not exposed to the user as selection criteria (e.g., weighted combinations of ratings from multiple sources). These data are one or more relevant to the task, but the distinctions among these three kinds of data are not relevant to the user. Since the user thinks in terms of features of the desired choice rather than in properties and dimensions, assistant **1002** operationalizes these various criteria into features of the items. Assistant **1002** provides a user-facing domain data model and maps it to data found in web services.

[0792] One type of mapping is an isomorphism from underlying data to user-facing criteria. For example, the availability of tables for reservations as seen by the user could be exactly what an online reservation website, such as opentable.com, offers, using the same granularity for time and party size.

[0793] Another type of mapping is a normalization of data from one or more services to a common value set, possibly with a unification of equivalent values. For example, cuisines of one or more restaurants may be represented as a single ontology in assistant **1002**, and mapped to various vocabularies used in different services. That ontology might be hierarchical, and have leaf nodes pointing to specific values from at least one service. For example, one service might have a cuisine value for "Chinese", another for "Szechuan", and a third for "Asian". The ontology used by assistant **1002** would cause references to "Chinese food" or "Szechuan" to semantically match one or more of these nodes, with confidence levels reflecting the degree of match.

[0794] Normalization might also be involved when resolving differences in precision. For example, the location of a restaurant may be given to the street level in one service but only to city in another. In one embodiment, assistant **1002** uses a deep structural representation of locations and times that may be mapped to different surface data values.

[0795] In one embodiment, assistant **1002** uses a special kind of mapping for open-ended qualifiers (e.g., romantic, quiet) which may be mapped to matches in full text search, tags, or other open-textured features. The name of the selection constraint in this case would be something like "is described as".

[0796] In at least one embodiment, constraints may be mapped to operational preference orderings. That is, given the name of a selection criterion and its constraint value, assistant **1002** is able to interpret the criterion as an ordering over possible items. There are several technical issues to address in such a mapping. For example:

[0797] Preference orderings may conflict. The ordering given by one constraint may be inconsistent or even inversely correlated with the ordering given by another. For example, price and quality tend to be in opposition. In one embodiment, assistant **1002** interprets constraints chosen by the user in a weighted or otherwise combined ordering that reflects the user's desires but is true to the data. For example, the user may ask for "cheap fast food French restaurants within walking distance rated highly". In many locations, there may not be any such restaurant. However, in one embodiment, assistant **1002** may show a list of items that tries to optimize for at least one constraint, and explain why at least one is listed. For example, item one might be "highly rated French cuisine" and another "cheap fast food within walking distance".

[0798] Data may be used as either hard or soft constraints. For example, the price range of a restaurant may be important to choosing one, but it may be difficult to state a threshold value for price up-front. Even seemingly hard constraints like cuisine may be, in practice, soft constraints because of partial matching. Since, in one embodiment, assistant **1002** using a data modeling strategy that seeks to flatten one or more criteria into symbolic values (such as "cheap" or "close"), these constraints may be mapped into a function that gets the criteria and order right, without being strict about matching specific threshold values. For symbolic criteria with clear objective truth values, assistant **1002** may weight the objective criteria higher than other criteria, and make it clear in the explanation that it knows that some of the items do not strictly match the requested criteria.

[0799] Items may match some but not one or more constraints, and the "best fitting" items may be shown.

[0800] In general, assistant **1002** determines which item features are salient for a domain, and which may serve as selection criteria, and for at least one criteria, possible constraint values. Such information can be provided, for example, via operational data and API calls.

Paraphrase and Prompt Text

[0801] As described above, in one embodiment assistant **1002** provides feedback to show it understands the user's intent and is working toward the user's goal by producing paraphrases of its current understanding. In the conversational dialog model of the present invention, the paraphrase is what assistant **1002** outputs after the user's input, as a preface (for example, paraphrase **4003** in FIG. **40**) or summary of the results to follow (for example, list **3502** in FIG. **35**). The prompt is a suggestion to the user about what else they can do to refine their request or explore the selection space along some dimensions.

[0802] In one embodiment, the purposes of paraphrase and prompt text include, for example:

[0803] to show that assistant **1002** understands the concepts in the user's input, not just the text;

[0804] to indicate the boundaries of assistant's **1002** understanding;

[0805] to guide the user to enter text that is required for the assumed task;

[0806] to help the user explore the space of possibilities in constrained selection;

[0807] to explain the current results obtained from services in terms of the user's stated criteria and assistant's **1002** assumptions (for example, to explain the results of under- and over-constrained requests).

[0808] For example, the following paraphrase and prompt illustrates several of these goals:

[0809] User input: indonesian food in menlo park

[0810] System Interpretation:

[0811] Task=constrainedSelection

[0812] SelectionClass=restaurant

[0813] Constraints:

[0814] Location=Menlo Park, Calif.

[0815] Cuisine=Indonesian (known in ontology)

[0816] Results from Services: no strong matches

[0817] Paraphrase: Sorry, I can't find any Indonesian restaurants near MenloPark.

[0818] Prompt: You could try other cuisines or locations.

[0819] Prompt Under Hypertext Links:

[0820] Indonesian: You can try other food categories such as Chinese, or a favorite food item such as steak.

[0821] MenloPark: Enter a location such as a city, neighborhood, street address, or "near" followed by a landmark.

[0822] Cuisines: Enter a food category such as Chinese or Pizza.

[0823] Locations: Enter a location: a city, zip code, or "near" followed by the name of a place.

[0824] In one embodiment, assistant **1002** responds to user input relatively quickly with the paraphrase. The paraphrase is then updated after results are known. For example, an initial response may be "Looking for Indonesian restaurants near MenloPark . . . " Once results are obtained, assistant **1002** would update the text to read, "Sorry, I can't find any Indonesian restaurants near MenloPark. You could try other cuisines or locations." Note that certain items are highlighted (indicated here by underline), indicating that those items represent constraints that can be relaxed or changed.

[0825] In one embodiment, special formatting/highlighting is used for key words in the paraphrase. This can be helpful to facilitate training of the user for interaction with intelligent automated assistant **1002**, by indicating to the user which words are most important to, and more likely to be recognized by, assistant **1002**. User may then be more likely to use such words in the future.

[0826] In one embodiment, paraphrase and prompt are generated using any relevant context data. For example, any of the following data items can be used, alone or in combination:

[0827] The parse—a tree of ontology nodes bound to their matching input tokens, with annotations and exceptions. For each node in the parse, this may include the node's metadata and/or any tokens in the input that provide evidence for the node's value.

[0828] The task, if known

[0829] The selection class.

[0830] The location constraint, independent of selection class.

[0831] Which required parameters are unknown for the given selection class (e.g., location is a required constraint on restaurants).

[0832] The name of a named entity in the parse that is an instance of the selection class, if there is one (e.g., a specific restaurant or movie name.)

[0833] Is this a follow-up refinement or the beginning of a conversation? (Reset starts a new conversation.)

[0834] Which constraints in the parse are bound to values in the input that changed their values? In other words, which constraints were just changed by the latest input?

[0835] Is the selection class inferred or directly stated?

[0836] Sorted by quality, relevance, or proximity?

[0837] For each constraint specified, how well was it matched?

[0838] Was refinement entered as text or clicking?

[0839] In one embodiment, the paraphrase algorithm accounts for the query, domain model **1056**, and the service results. Domain model **1056** contains classes features including metadata that is used to decide how to generate text. Examples of such metadata for paraphrase generation include

[0840] IsConstraint={true|false}

[0841] IsMultiValued={true|false}

[0842] ConstraintType={EntityName, Location, Time, CategoryConstraint, AvailabilityConstraint, BinaryConstraint, SearchQualifier, GuessedQualifier}

[0843] DisplayName=string

[0844] DisplayTemplateSingular=string

[0845] DisplayTemplatePlural=string

[0846] GrammaticalRole={AdjectiveBeforeNoun, Noun,ThatClauseModifer}

[0847] For example, a parse might contain these elements:

[0848] Class: Restaurant

[0849] IsConstraint=false

[0850] DisplayTemplateSingular="restaurant"

[0851] DisplayTemplatePlural="restaurants"

[0852] GrammaticalRole=Noun

[0853] Feature: RestaurantName (example: "Il Fornaio")

[0854] IsConstraint=true

[0855] IsMultiValued=false

[0856] ConstraintType=EntityName

[0857] DisplayTemplateSingular="named $1"

[0858] DisplayTemplatePlural="named $1"

[0859] GrammaticalRole=Noun

[0860] Feature: RestaurantCuisine (example: "Chinese")

[0861] IsConstraint=true

[0862] IsMultiValued=false

[0863] ConstraintType=CategoryConstraint

[0864] GrammaticalRole=AdjectiveBeforeNoun

[0865] Feature: RestaurantSubtype (example: "café")

[0866] IsConstraint=true

[0867] IsMultiValued=false

[0868] ConstraintType=CategoryConstraint

[0869] DisplayTemplateSingular="$1"

[0870] DisplayTemplatePlural="$1s"

[0871] GrammaticalRole=Noun

[0872] Feature: RestaurantQualifiers (example: "romantic")

[0873] IsConstraint=true

[0874] IsMultiValued=true

[0875] ConstraintType=SearchQualifier

[0876] DisplayTemplateSingular="is described as $1"

[0877] DisplayTemplatePlural="are described as $1"

[0878] DisplayTemplateCompact="matching $1"

[0879] GrammaticalRole=Noun

[0880] Feature: FoodType (example: "burritos")

[0881] IsConstraint=true

[0882] IsMultiValued=false

[0883] ConstraintType=SearchQualifier

[0884] DisplayTemplateSingular="serves $1"

[0885] DisplayTemplatePlural="serve $1"

[0886] DisplayTemplateCompact="serving $1"

[0887] GrammaticalRole=ThatClauseModifer

[0888] Feature: IsRecommended (example: true)

[0889] IsConstraint=true

[0890] IsMultiValued=false

[0891] ConstraintType=BinaryConstraint

[0892] DisplayTemplateSingular="recommended"

[0893] DisplayTemplatePlural="recommended"

[0894] GrammaticalRole=AdjectiveBeforeNoun

[0895] Feature: RestaurantGuessedQualifiers (example: "spectacular")

[0896] IsConstraint=true

[0897] IsMultiValued=false

[0898] ConstraintType=GuessedQualifier

[0899] DisplayTemplateSingular="matches $1 in reviews"

[0900] DisplayTemplatePlural="match $1 in reviews"

[0901] DisplayTemplateCompact="matching $1"

[0902] GrammaticalRole=ThatClauseModifer

[0903] In one embodiment, assistant **1002** is able to handle unmatched input. To handle such input, domain model **1056** can provide for nodes of type GuessedQualifier for each selection class, and rules that match otherwise unmatched words if they are in the right grammatical context. That is, GuessedQualifiers are treated as miscellaneous nodes in the parse which match when there are words that are not found in the ontology but which are in the right context to indicate that that are probably qualifiers of the selection class. The difference between GuessedQualifiers and SearchQualifiers is that the latter are matched to vocabulary in the ontology. This distinction allows us to paraphrase that assistant **1002** identified the intent solidly on the SearchQualifiers and can be more hesitant when echoing back the GuessedQualifiers.

[0904] In one embodiment, assistant **1002** performs the following steps when generating paraphrase text:

[0905] 1. If the task is unknown, explain what assistant **1002** can do and prompt for more input.

[0906] 2. If the task is a constrained selection task and the location is known, then explain the domains that assistant **1002** knows and prompt for the selection class.

[0907] 3. If the selection class is known but a required constraint is missing, then prompt for that constraint. (for example, location is required for constrained selection on restaurants)

[0908] 4. If the input contains an EntityName of the selection class, then output "looking up"<name> in <location>.

[0909] 5. If this is the initial request in a conversation, then output "looking for" followed by the complex noun phrase that describes the constraints.

[0910] 6. If this is a follow-up refinement step in the dialog,

[0911] a. If the user just completed a required input, then output "thanks" and then paraphrase normally. (This happens when there is a required constraint that is mapped to the user input.)

[0912] b. If the user is changing a constraint, acknowledge this and then paraphrase normally.

[0913] c. If the user typed in the proper name of an instance of the selection class, handle this specially.

[0914] d. If the user just added an unrecognized phrase, then indicate how it will be folded in as search. If appropriate, the input may be dispatched to a search service.

[0915] e. If the user is just adding a normal constraint, then output "OK", and paraphrase normally.

[0916] 7. To explain results, use the same approach for paraphrase. However, when the results are surprising or unexpected, then explain the results using knowledge about the data and service. Also, when the query is over- or underconstrained, prompt for more input.

Grammar for Constructing Complex Noun Phrases

[0917] In one embodiment, when paraphrasing **734** a constrained selection task query, the foundation is a complex noun phrase around the selection class that refers to the current constraints. Each constraint has a grammatical position, based on its type. For example, in one embodiment, assistant **1002** may construct a paraphrase such as:

[0918] recommended romantic Italian restaurants near MenloPark with opentablesfor2 that serve osso buco and are described as "quiet"

[0919] A grammar to construct this is

```
<paraphraseNounClause> := <binaryConstraint> <searchQualifier>
  <categoryConstraint> <itemNoun> <locationConstraint>
  <availabiltyConstraint> <adjectivalClauses>
```

[0920] <binaryConstraint>:==single adjective that indicates the presence or absence of a BinaryConstraint (e.g., recommended (best), affordable (cheap))

[0921] It is possible to list more than one in the same query.

[0922] <searchQualifier>:==a word or words that match the ontology for a qualifier of the selection class, which would be passed into a search engine service. (e.g., romantic restaurants, funny movies).

[0923] Use when ConstraintType=Search Qualifier.

[0924] <categoryConstraint>:==an adjective that identifies the genre, cuisine, or category of the selection class (e.g., Chinese restaurant or R-rated file). It is the last prefix adjective because it is the most intrinsic. Use for features of type CategoryConstraint and GrammaticalRole=AdjectiveBeforeNoun.

```
<itemNoun> := <namedEntityPhrase> | <selectionClass> |
  <selectionClassSubType>
```

[0925] find the most specific way to display the noun. NamedEntity <SubType <Class

[0926] <selectionClass>:==a noun that is the generic name for the selection class (e.g., restaurant, movie, place)

[0927] <selectionClassSubType>:==a noun phrase that is the subtype of the selection class if it is known (e.g., diner, museum, store, bar for the selection class local business). Use for features in which ConstraintType=CategoryConstraint and GrammaticalRole=AdjectiveBeforeNoun.

```
<namedEntityPhrase> := <entityName> |
  "the" (<selectionClass> | <selectionClassSubType>)
```

[0928] <entityName>:==the proper name of an instance of the selection class (e.g., "Il Formaio" "Animal House" "Harry's Bar")

[0929] <locationConstraint>:
==<locationPreposition><locationName>

[0930] <locationPreposition>:=="in", "near", "at", and the like

[0931] <locationName>:==city, street address, landmark, or something for GPS like "your current location"

[0932] <availabilityConstraint>:==the availability constraint expressed as a prepositional phrase that comes after the noun (e.g., "with open tables", "with seats available", "available online"). It comes right after the noun to give it salience.

<adjectivalClauses> :== <modiferVerbPhrase> | "that" <modiferVerbPhrase> "and" <modiferVerbPhrase>

[0933] <modiferVerbPhrase>:=a verb phrase that expresses a search-keyword style constraint on the selection class (e.g., restaurants that "are described as quiet", "serve meat after 11", "match 'tragically hip' in reviews"; movies that "contain violence", "star Billy Bob Thornton"). Include all constraints in the parse whose GrammaticalRole=ThatClauseModifer, using the "that . . . and" variant if there are more than one. Use the DisplayTemplatePlural to generate the "that" clauses, putting the GuessedQualifier last. If there is only one such constraint, use the DisplayTemplateCompact variant.

[0934] Table 2 provides some examples of paraphrases that may be provided in response to the first input for a task, according to one embodiment.

TABLE 2

Paraphrases in response to first input

Input	Initial Paraphrase
tie house	Sorry, I don't understand "tie house". I know about restaurants, hotels, and other local businesses, flights, weather, and buying things online. What are you looking for, and where?
palo alto	I can help you find restaurants, businesses, events, and things to do in Palo Alto. What are you looking for?
food	Looking for a place to eat? Where? (examples)
Italian food	I can tell you about Italian restaurants in your area. Where would you like to eat?
burrito	I can help you find places that serve burritos in your area. Where would you like to eat?
best mexican food	I can help you find good Mexican restaurants in your area. Where are you?
restaurants palo alto	Looking for restaurants near Palo Alto.
Italian palo alto	Looking for Italian restaurants near Palo Alto.
thai food in menlo park	Looking for Thai restaurants near Menlo Park
indonesian food in menlo park	Looking for Indonesian restaurants near Menlo Park
best mexican food in Palo Alto	Looking for recommended Mexican restaurants near Palo Alto.
fast food menlo park	Looking for fast food restaurants near Menlo Park
healthy food in menlo park	Looking for places near Menlo Park that serve healthy food.
dive bar in menlo park	Looking for dive bars near Menlo Park.
pool room in menlo park	Looking for pool rooms near Menlo Park.
hookah room in menlo park	Looking for places near Menlo Park described as "hookah room".
spicy food in sf	Looking for places near Menlo Park that serve spicy food.
burgers in Menlo park	Looking for places near Menlo Park that serve burgers.
belly dancing place in sf	Looking for places near San Francisco described as "belly dancing".
belly dancing place in menlo park	Looking for places near Menlo Park described as "belly dancing".
juke joint in sf	Looking for joints near San Francisco described as "juke".
tie house Menlo park	Looking for places near Menlo Park described as "tie house".
thai house palo alto	Looking up Thai House in Palo Alto
il fornaio menlo park	Looking up II Fornaio in Palo Alto
il fornao in Menlo park	Looking for places near Palo Alto described as "il fornao".
cliff house in Menlo park	Looking up Cliff House in Palo Alto
romantic restaurant in palo alto	Looking for romantic restaurants in Palo Alto
romantic restaurant in palo alto	Looking for romantic restaurants in Palo Alto
quiet romantic restaurant in palo alto	Looking for romantic restaurants in Palo Alto described as "quiet"
quiet romantic restaurant in palo alto	Looking for romantic restaurants in Palo Alto described as "quiet"
tragically hip restaurant in palo alto	Looking for restaurants in Palo Alto described as "tragically hip"
tragically hip restaurant in palo alto	Looking for restaurants in Palo Alto described as "tragically hip"
table for two tonight in palo alto	Checking availability for tonight at restaurants near Palo Alto.
table for two italian food tonight in palo alto	Checking availability for tonight at Italian restaurants near Palo Alto.
il fornaio palo alto 6pm for 2	Checking availability for tables tonight at II Fornaio in Palo Alto.
table for 2 at thai house in PA	Checking availability for tables tonight at Thai House in Palo Alto.

TABLE 2-continued

Paraphrases in response to first input	
Input	Initial Paraphrase
pizza delivery in menlo park	
I want the best quiet romantic italian restaurant for osso buco in or near menlo park	OK, looking for recommended romantic italian restaurants near Menlo Park that serve osso buco and are described as "quiet".
Book me a table for 2 at the best quiet romantic italian restaurant for osso buco in or near menlo park	OK, looking for recommended romantic italian restaurants near Menlo Park with open tables for 2 that serve osso buco and are described as "quiet".

Refining a Query about Places to Eat

[0935] Table 3 provides some examples of paraphrases in response to a situation where user's intent to find a place to eat is known, but they have not selected a particular place to eat. They are facing a list of restaurants, and they provide refinements.

TABLE 3

Paraphrases in response to refinements	
Refinements	
menlo park (was unknown)	Thanks, now looking for restaurants in Menlo Park.
cheapest	OK, looking for affordable restaurants near Menlo Park.
chinese (was italian)	OK, changing cuisine to Chinese. Looking for Chinese restaurants near Menlo Park.

TABLE 3-continued

Paraphrases in response to refinements	
Refinements	
pizza (adding food item)	OK, looking for Italian restaurants near Menlo Park that serve pizza.
san francisco (was menlo park)	OK, changing location to San Francisco. Looking for Italian restaurants near San Francisco that serve pizza.
tragically hip	OK, I'll search reviews for "tragically hip".
quiet romantic	OK, looking for romantic restaurants in Palo Alto described as "quiet"
cliff house	Cliff House sounds familiar. I'll look it up in Palo Alto.

[0936] Table 4 provides some examples of results summaries that are provided once results are obtained.

TABLE 4

Results Summaries		
Input	# Results	Results Summary
tie house	N/A	N/A (no call to services)
palo alto	N/A	N/A (no call to services)
food	N/A	N/A (no call to services)
Italian food	N/A	N/A (no call to services)
burrito	N/A	N/A (no call to services)
best mexican food	N/A	N/A (no call to services)
restaurants palo alto	many	Here are some recommended restaurants near Palo Alto. Do you prefer any particular food type? Would you like a reservation?
Italian palo alto	some	Here are some Italian restaurants near Palo Alto. To reserve a table, click on available times.
thai food in menlo park	some	Here are some Thai restaurants near Menlo Park. To reserve a table, click on available times.
indonesian food in menlo park	0	Sorry, I can't find any Indonesian restaurants near Menlo Park. You could try other cuisines or locations.
best mexican food in Palo Alto	some	Here are some recommended Mexican restaurants near Palo Alto.
fast food menlo park	some	OK, here are some fast food restaurants near Menlo Park.
healthy food in menlo park	some	Here are some places near Menlo Park that might serve healthy food, based on reviews.
dive bar in menlo park	some	Here are some bars near Menlo Park. I don't see anything specific to "dive bar". Are there other ways of saying it?
pool room in menlo park	some	Here are some places in Menlo Park that are described as pool rooms. You can add other words to describe what you want.
hookah room in menlo park	0	I can't find places Menlo Park matching "hookah room". Interested in restaurants, hotels, or local businesses?
spicy food in sf	weak match	Here are some places to eat in San Francisco. I don't know which serve spicy food. Could it go by another name?

TABLE 4-continued

	Results Summaries	
	# Results	Results Summary
burgers in Menlo park	some	You might find burgers at these places to eat near Menlo Park.
belly dancing place in sf	some	Here are some places in Menlo Park that are described as "belly dancing". You can add other words to describe what you want.
belly dancing place in menlo park	0	I can help you find restaurants, businesses, events, and things to do in San Francisco. What are some other words for "belly dancing place"?
juke joint in sf	0	I can help you find restaurants, businesses, events, and things to do in San Francisco. What are some other words for "juke joint"?
tie house Menlo park	0	Sorry, I don't know about "tie house" in Menlo Park. Any other names for it?
thai house palo alto	1	Got it: Thai House restaurant in Palo Alto, CA. Want to reserve a table? When and for how many?
il fornaio menlo park	weak match	There is a Il Fornaio restaurant in Palo Alto, which is near Menlo Park.
il fornao in Menlo park	spell match	Did you mean Il Fornaio in Palo Alto? If not, try different words.
cliff house in Menlo park	0	I couldn't find Cliff House in Palo Alto. Could it go by a different name or be in a different city?
romantic restaurant in palo alto	some	Here are some restaurants near Palo Alto that are described in reviews as romantic.
romantic restaurant in palo alto	0	Here are restaurants near Palo Alto. Searching reviews did not find any described as romantic. Any other preferences?
quiet romantic restaurant in palo alto	some	Here are some restaurants near Palo Alto that match "quiet romantic" in reviews.
quiet romantic restaurant in palo alto	0	Here are some restaurants near Palo Alto. Searching reviews did not find any matching "quiet romantic". Any other preferences?
tragically hip restaurant in palo alto	some	Here are some restaurants near Palo Alto that match "tragically hip" in reviews.
tragically hip restaurant in palo alto	0	Here are some restaurants near Palo Alto. Searching reviews did not find any matching "tragically hip". Any other preferences?
table for two tonight in palo alto	some	Here are restaurants with open tables from 6pm to 8pm tonight for a party of two in Palo Alto, CA.
table for two italian food tonight in palo alto	some	Here are restaurants with open tables from 6pm to 8pm tonight for a party of two in Palo Alto, CA.
il fornaio palo alto 6pm for 2	1	Here are table options at Il Fornaio restaurant in Palo Alto, CA at 6pm today for a party of 2.
table for 2 at thai house in PA	1	Here is the information for Thai House in Palo Alto, CA. You can call them at the phone number listed.
pizza delivery in menlo park		Here are places that serve pizza in menlo park. I do not have delivery options at this time. Click on restaurant phone numbers to call for delivery.
I want the best quiet romantic italian restaurant for osso buco in or near menlo park	weak match, sorted by availability	Here are the best matches for recommended romantic italian restaurants near Menlo Park that serve osso buco and are described as "quiet". Click on available times to reserve a table.
Book me a table for 2 at the best quiet romantic italian restaurant for osso buco in or near menlo park	weak match, sorted by availability	Here are the best matches for recommended romantic italian restaurants near Menlo Park with open tables for 2 that serve osso buco and are described as "quiet". Click on available times to reserve a table.
refinements		
menlo park (was unknown)	some	Here are some recommended restaurants near Menlo Park. Do you prefer any particular food type?
cheapest	some	I found 4 affordable restaurants near Menlo Park.
chinese (was italian)		I found 4 Chinese restaurants near Menlo Park.
pizza (adding food item)	some	I found 4 Italian restaurants near Menlo Park that serve pizza.
san francisco (was menlo park)	some	I found 4 Italian restaurants near San Francisco.
tragically hip	some	I found 4 restaurants near Palo Alto that match "tragically hip" in reviews.
quiet romantic	some	Here are some restaurants near Palo Alto that match "quiet romantic" in reviews.
cliff house	0	I couldn't find Cliff House in Palo Alto. Could it go by a different name or be in a different city?

[0937] Table 5 provides some examples of prompts that are provided when users click on active links.
Prompts when Users Click on Active Links

TABLE 5

Prompts when users click on active links

Anchor Text	Prompt Text	Notes
Location, where	Enter a location: a city, zip code, or "near" followed by the name of a place.	This prompt might be used when the user has not specified a location yet.
Palo Alto	Enter a location such as a city, neighborhood, street address, or "near" followed by a landmark.	This prompt might be used when the user is changing locations.
food type	Enter a food category such as Chinese or Pizza.	Merge food type and cuisine can be merged
Italian	You can try other food categories such as Chinese, or a favorite food item such as steak.	User already said Italian. Assistant 1002 is helping the user explore alternatives. If it is a food item, it dominates over cuisine.
reservation	Enter the day and time to reserve a table, such as "tomorrow at 8".	Prompting for a reservation
healthy food	You can also enter menu items or cuisines	Known food type
spicy food	You can also enter menu items or cuisines	Unknown food type
restaurants	What kind of restaurant? (e.g., Chinese, Pizza)	Clicking on the restaurants link should insert the word "restaurant" on the end of the text input.
businesses	You can find local florists, ATMs, doctors, drug stores, and the like What kind of business are you looking for?	Clicking on the businesses link should add to the machine readable tag that this is a local search
events	You can discover upcoming converts, shows, and the like What interests you?	
things to do	Music, art, theater, sports, and the like What kind of thing would you like to do in this area?	
hotels	I can help you find an available hotel room. Any preferences for amenities or location?	
weather	Enter a city, and I'll tell you what the weather is like there.	If location is known, just show the weather data
buying things	I can help you find music, movies, books, electronics, toys, and more -- and buy it from Amazon. What are you looking for?	

Suggesting Possible Responses in a Dialog

[0938] In one embodiment, assistant **1002** provides contextual suggestions. Suggestions a way for assistant **1002** to offer the user options to move forward from his or her current situation in the dialog. The set of suggestions offered by assistant **1002** depends on context, and the number of suggestions offered may depend on the medium and form factor. For example, in one embodiment, the most salient suggestions may be offered in line in the dialog, an extended list of suggestions ("more") may be offered in a scrollable menu, and even more suggestions are reachable by typing a few characters and picking from autocomplete options. One skilled in the art will recognize that other mechanisms may be used for providing suggestions.

[0939] In various embodiments, different types of suggestions may be provided. Examples of suggestion types include:

[0940] options to refine a query, including adding or removing or changing constraint values;

[0941] options to repair or recover from bad situations, such as "not what I mean" or "start over" or "search the web";

[0942] options to disambiguate among;

[0943] interpretations of speech;

[0944] interpretations of text, including spell correction and semantic ambiguity;

[0945] context-specific commands, such as "show these on a map" or "send directions to my date" or "explain these results";

[0946] suggested cross-selling offers, such as next steps in meal or event planning scenarios;

[0947] options to reuse previous commands, or parts of them.

[0948] In various embodiments, the context that determines the most relevant suggestions may be derived from, for example:

[0949] dialog state

[0950] user state, including, for example:

[0951] static properties (name, home address, etc)

[0952] dynamic properties (location, time, network speed)

[0953] interaction history, including, for example:

[0954] query history

[0955] results history

[0956] the text that has been entered so far into autocomplete.

[0957] In various embodiments, suggestions may be generated by any mechanism, such as for example:

[0958] paraphrasing a domain, task, or constraint based on the ontology model;

[0959] prompting in autocomplete based on the current domain and constraints;

[0960] paraphrasing ambiguous alternative interpretations;

[0961] alternative interpretations of speech-to-text;

[0962] hand authoring, based on special dialog conditions.

[0963] According to one embodiment, suggestions are generated as operations on commands in some state of completion. Commands are explicit, canonical representations of requests, including assumptions and inferences, based on attempted interpretations on user input. In situations where the user input is incomplete or ambiguous, suggestions are an attempt to help the user adjust the input to clarify the command.

[0964] In one embodiment, each command is an imperative sentence having some combination of a

[0965] command verb (imperative such as "find" or "where is");

[0966] domain (selection class such as "restaurants");

[0967] constraint(s) such as location=Palo Alto and cuisine=Italian.

[0968] These parts of a command (verb, domain, constraints) correspond to nodes in the ontology.

[0969] A suggestion, then, may be thought of as operations on a command, such as setting it, changing it, or declaring that it is relevant or not relevant. Examples include:

[0970] setting a command verb or domain ("find restaurants")

[0971] changing a command verb ("book it", "map it", "save it")

[0972] changing a domain ("looking for a restaurant, not a local business")

[0973] stating that a constraint is relevant ("try refining by cuisine")

[0974] choosing a value for a constraint ("Italian", "French", and the like)

[0975] choosing a constraint and value together ("near here", "tables for 2")

[0976] stating that a constraint value is wrong ("not that Boston")

[0977] stating that a constraint is not relevant ("ignore the expense")

[0978] stating the intent to change a constraint value ("try a different location")

[0979] changing a constraint value ("Italian, not Chinese")

[0980] adding to a constraint value ("and with a pool, too")

[0981] snapping a value to grid ("Los Angeles, not los angelos")

[0982] initiating a new command, reusing context ([after movies] "find nearby restaurants", "send directions to my friend")

[0983] initiating a command that is "meta" to context ("explain these results")

[0984] initiating a new command, resetting or ignoring context ("start over", "help with speech")

[0985] A suggestion may also involve some combination of the above. For example:

[0986] "the movie Milk not [restaurants serving] the food item milk"

[0987] "restaurants serving pizza, not just pizza joints"

[0988] "The place called Costco in Mountain View, I don't care whether you think it is a restaurant or local business"

[0989] "Chinese in mountain view" [a recent query]

[0990] In one embodiment, assistant **1002** includes a general mechanism to maintain a list of suggestions, ordered by relevance. The format in which a suggestion is offered may differ depending on current context, mode, and form factor of the device.

[0991] In one embodiment, assistant **1002** determines which constraints to modify by considering any or all of the following factors:

[0992] Consider whether the constraint has a value;

[0993] Consider whether the constraint was inferred or explicitly stated;

[0994] Consider its salience (suggestionIndex).

[0995] In one embodiment, assistant **1002** determines an output format for the suggestion. Examples of output formats include:

[0996] change domain:

[0997] if autocomplete option "find restaurants", then "try something different"

[0998] else [was inferred] "not looking for restaurants"

[0999] change name constraint:

[1000] if name was inferred, offer alternative ambiguous interpretation"

[1001] stuff into autocomplete the entity names from current results

[1002] different name

[1003] consider that it wasn't a name lookup (remove constraint)—maybe offer category in place of it

[1004] "not named"

[1005] "not in Berkeley"

[1006] "some other day"

[1007] not that sense of (use ambiguity alternatives)

[1008] inferred date: "any day, I don't need a reservation"

[1009] In one embodiment, assistant **1002** attempts to resolve ambiguities via suggestions. For example, if the set of current interpretations of user intent is too ambiguous **310**, then suggestions are one way to prompt for more information **322**. In one embodiment, for constrained selection tasks, assistant **1002** factors out common constraints among ambiguous interpretations of intent **290** and presents the differences among them to the user. For example, if the user input includes the word "café" and this word could match the name of a restaurant or the type of restaurant, then assistant **102** can ask "did you mean restaurants named 'café' or 'café restaurants'?"

[1010] In one embodiment, assistant **1002** infers constraints under certain situations. That is, for constrained selection tasks, not all constraints need be mentioned explicitly in the user input; some can be inferred from other information available in active ontology **1050**, short term memory **1052**, and/or other sources of information available to assistant **1002**. For example:

[1011] Inferring domain or location

[1012] Default assumption, like location

[1013] Weakly matched constraint (fuzzy, low salience location, etc)

[1014] Ambiguous criteria (match to constraint value without prefix (name vs. category, often ambiguous)

[1015] In cases where the assistant **1002** infers constraint values, it may also offer these assumptions as suggestions for the user to overrule. For example, it might tell the user "I assumed you meant around here. Would you like to look at a different location?"

[1016] The present invention has been described in particular detail with respect to possible embodiments. Those of skill in the art will appreciate that the invention may be practiced in other embodiments. First, the particular naming of the components, capitalization of terms, the attributes, data structures, or any other programming or structural aspect is not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, formats, or protocols. Further, the system may be implemented via a combination of hardware and software, as described, or entirely in hardware elements, or entirely in software elements. Also, the particular division of functionality between the various system components described herein is merely exemplary, and not mandatory; functions performed by a single system component may instead be performed by multiple components, and functions performed by multiple components may instead be performed by a single component.

[1017] In various embodiments, the present invention can be implemented as a system or a method for performing the above-described techniques, either singly or in any combination. In another embodiment, the present invention can be implemented as a computer program product comprising a nontransitory computer-readable storage medium and computer program code, encoded on the medium, for causing a processor in a computing device or other electronic device to perform the above-described techniques.

[1018] Reference in the specification to "one embodiment" or to "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[1019] Some portions of the above are presented in terms of algorithms and symbolic representations of operations on data bits within a memory of a computing device. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps (instructions) leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic or optical signals capable of being stored, transferred, combined, compared and otherwise manipulated. It is convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. Furthermore, it is also convenient at times, to refer to certain arrangements of steps requiring physical manipulations of physical quantities as modules or code devices, without loss of generality.

[1020] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "displaying" or "determining" or the like, refer to the action and processes of a computer system, or similar electronic computing module and/or device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[1021] Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that the process steps and instructions of the present invention can be embodied in software, firmware and/or hardware, and when embodied in software, can be downloaded to reside on and be operated from different platforms used by a variety of operating systems.

[1022] The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computing device selectively activated or reconfigured by a computer program stored in the computing device. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Further, the computing devices referred to herein may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

[1023] The algorithms and displays presented herein are not inherently related to any particular computing device, virtualized system, or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will be apparent from the description provided herein. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references above to specific languages are provided for disclosure of enablement and best mode of the present invention.

[1024] Accordingly, in various embodiments, the present invention can be implemented as software, hardware, and/or other elements for controlling a computer system, computing device, or other electronic device, or any combination or plurality thereof. Such an electronic device can include, for example, a processor, an input device (such as a keyboard, mouse, touchpad, trackpad, joystick, trackball, microphone, and/or any combination thereof), an output device (such as a screen, speaker, and/or the like), memory, long-term storage (such as magnetic storage, optical storage, and/or the like), and/or network connectivity, according to techniques that are well known in the art. Such an electronic device may be portable or nonportable. Examples of electronic devices that may be used for implementing the invention include: a mobile phone, personal digital assistant, smartphone, kiosk, desktop computer, laptop computer, tablet computer, consumer electronic device, consumer entertainment device; music player; camera; television; set-top box; electronic gaming unit; or the like. An electronic device for implementing the present inven-

tion may use any operating system such as, for example, iOS or MacOS, available from Apple Inc. of Cupertino, Calif., or any other operating system that is adapted for use on the device.

[1025] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of the above description, will appreciate that other embodiments may be devised which do not depart from the scope of the present invention as described herein. In addition, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the claims.

What is claimed is:

1. An automated assistant operating on a computing device, the assistant comprising:

an input device, for receiving user input;

a language interpreter component, for interpreting the received user input to derive a representation of user intent;

a dialog flow processor component, for identifying at least one domain, at least one task, and at least one parameter for the task, based at least in part on the derived representation of user intent;

a services orchestration component, for calling at least one service for performing the identified task;

an output processor component, for rendering output based on data received from the at least one called service, and further based at least in part on a current output mode; and

an output device, for outputting the rendered output.

2. The automated assistant of claim **1**, further comprising:

an active input elicitation component, for generating at least one prompt to actively elicit input from a user;

wherein the output device outputs the generated at least one prompt.

3. The automated assistant of claim **2**, wherein:

the active input elicitation component generates the at least one prompt to actively elicit input from a user via a conversational interface; and

the input device receives the user input via the conversational interface.

4. The automated assistant of claim **2**, wherein the active input elicitation component performs constrained selection to generate the at least one prompt, by: obtaining an ordered list of criteria;

obtaining, for at least one of the obtained criteria, obtaining at least one constraint; and

based on the at least one criterion and the at least one constraint, generating a set of candidate items.

5. The automated assistant of claim **2**, further comprising:

an active ontology, comprising representations of concepts and relationships among concepts;

wherein the active input elicitation component generates the at least one prompt using at least a subset of the representations in the active ontology.

6. The automated assistant of claim **2**, wherein:

the active input elicitation component generates the at least one prompt to actively elicit input from a user via multiple input modes; and

the input device receives the user input via the multiple input modes.

7. The automated assistant of claim **6**, wherein the multiple input modes comprise at least one selected from the group consisting of:

typed input;

spoken input; and

input provided via a graphical user interface.

8. The automated assistant of claim **2**, further comprising an event detector, for detecting at least one event, and wherein the active input elicitation component generates the at least one prompt responsive to at least one detected event.

9. The automated assistant of claim **1**, further comprising:

a task flow models component, for identifying at least one task flow model representing steps for performing the identified at least one task;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the task flow models component.

10. The automated assistant of claim **9**, further comprising:

an active ontology, comprising representations of concepts and relationships among concepts;

wherein the at least one task flow model is organized according to the active ontology.

11. The automated assistant of claim **9**, wherein the dialog flow processor component performs constrained selection to identify the at least one task flow model.

12. The automated assistant of claim **1**, further comprising:

a domain models component, comprising at least one representation of a domain;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the domain models component.

13. The automated assistant of claim **12**, wherein each representation of a domain comprises at least one selected from the group consisting of:

a representation of at least one concept;

a representation of at least one entity; and

a representation of at least one relationship.

14. The automated assistant of claim **12**, further comprising:

an active ontology, comprising representations of concepts and relationships among concepts;

wherein the at least one representation of a domain is organized according to the active ontology.

15. The automated assistant of claim **12**, wherein the domain models component uses modeling abstractions from constrained selection to define at least one representation of a domain.

16. The automated assistant of claim **1**, further comprising:

a dialog flow models component, comprising representations of steps taken in a conversation between the assistant and a user;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the dialog flow models component.

17. The automated assistant of claim **1**, further comprising:

a service capability models component, comprising representations of capabilities of services;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the service capability models component.

18. The automated assistant of claim **17**, wherein the services orchestration component selects at least one service to call based on at least one service model.

19. The automated assistant of claim **17**, wherein the service capability models component comprises a declarative model of at least one service, and wherein the services orchestration component dynamically selects among available services according to the declarative model and further according to the derived representation of user intent.

20. The automated assistant of claim **1**, further comprising:

a vocabulary database, comprising associations between words and concepts;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the vocabulary database.

21. The automated assistant of claim **1**, further comprising:

a domain entity database, comprising data describing domain entities;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the domain entity database to obtain data describing domain entities.

22. The automated assistant of claim **1**, further comprising:

a short term memory component, for storing data describing at least one user interaction with the automated assistant;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the short term memory component to obtain data describing at least one user interaction with the automated assistant.

23. The automated assistant of claim **22**, wherein the language interpreter component interprets the received user input based at least in part on information from the short term memory component.

24. The automated assistant of claim **1**, further comprising:

a long term memory component, for storing at least one selected from the group consisting of:

personal information associated with a user;

information collected by the user;

saved lists of entities relevant to the user;

transaction history associated with the user;

and wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component interfaces with the long term memory component to obtain data stored therein.

25. The automated assistant of claim **24**, wherein the language interpreter component interprets the received user input based at least in part on information from the long term memory component.

26. The automated assistant of claim **1**, wherein the dialog flow processor component performs constrained selection to identify at least one task and at least one parameter for the task.

27. The automated assistant of claim **26**, wherein the dialog flow processor component performs constrained selection by:

obtaining an ordered list of criteria;

obtaining, for at least one of the obtained criteria, obtaining at least one constraint; and

based on the at least one criterion and the at least one constraint, identifying at least one task and at least one parameter for the task.

28. The automated assistant of claim **1**, wherein the services orchestration component performs constrained selection to call at least one service.

29. The automated assistant of claim **28**, wherein the services orchestration component performs constrained selection by:

obtaining an ordered list of criteria;

obtaining, for at least one of the obtained criteria, obtaining at least one constraint; and

based on the at least one criterion and the at least one constraint, calling at least one service.

30. The automated assistant of claim **1**, further comprising:

an active ontology, comprising representations of concepts and relationships among concepts.

31. The automated assistant of claim **30**, wherein the language interpreter component interprets the received user input using at least a subset of the representations in the active ontology.

32. The automated assistant of claim **30**, wherein the dialog flow processor component identifies the at least one domain, task, and parameter using at least a subset of the representations in the active ontology.

33. The automated assistant of claim **30**, wherein the services orchestration component calls at least one service using at least a subset of the representations in the active ontology.

34. The automated assistant of claim **30**, wherein the output processor component renders output using at least a subset of the representations in the active ontology.

35. The automated assistant of claim **30**, wherein the active ontology is used as a semantic unifier for at least one database used by the automated assistant.

36. The automated assistant of claim **30**, wherein the active ontology is instantiated for each user session.

37. The automated assistant of claim **30**, wherein the active ontology comprises at least one selected from the group consisting of:

at least one domain model;

at least one dialog flow model;

at least one service model; and

at least one task model.

38. The automated assistant of claim **1**, further comprising:

an event monitor, for receiving data concerning events;

wherein the dialog flow processor component identifies at least one task and at least one parameter for the task based at least in part on received data concerning at least one event.

39. The automated assistant of claim **1**, wherein the input device comprises:

at least one microphone for receiving speech input;

and wherein the assistant further comprises:

a speech-to-text component, for translating the received speech input into text.

40. The automated assistant of claim **1**, wherein the input device receives text input.

41. The automated assistant of claim **1**, wherein the language interpreter disambiguates among alternative parse results for the received user input.

42. The automated assistant of claim **1**, wherein the services orchestration component calls at least one service for performing an operation on the computing device.

43. The automated assistant of claim **1**, wherein the services orchestration component calls at least one service for performing an operation via a computing network.

44. The automated assistant of claim **1**, wherein the services orchestration component calls at least one service for performing an operation via the Internet.

45. The automated assistant of claim **1**, wherein the services orchestration component receives results from the called at least one service, and unifies the received results.

46. The automated assistant of claim **1**, wherein the automated assistant operates on at least one selected from the group consisting of:

a telephone;
a smartphone;
a tablet computer;
a laptop computer;
a personal digital assistant;
a desktop computer;
a kiosk;
a consumer electronic device;
a consumer entertainment device;
a music player;
a camera;
a television;
an electronic gaming unit; and
a set-top box.

47. The automated assistant of claim **1**, wherein:

the output processor component paraphrases the at least one representation of user intent; and
the output device outputs the paraphrased representation of user intent.

48. The automated assistant of claim **1**, wherein at least one of the language interpreter component, the dialog flow processor component, the services orchestration component, and the output processor component obtain parameters from a detected context of the computing device.

49. The automated assistant of claim **48**, wherein the detected context of the computing device comprises at least one selected from the group consisting of:

a current software application and its current state;
a current location;
a current environmental condition detected via at least one environmental sensor;
a usage history; and
information describing the user.

50. The automated assistant of claim **1**, wherein the output device outputs the rendered output using a combination of at least two output modes.

51. The automated assistant of claim **50**, wherein at least two output modes comprise at least two selected from the group consisting of:

text output;
graphical output;
sound output;
synthetic speech output;
sampled speech output; and
actuator output.

52. An automated assistant operating on a computing device, the assistant comprising:

an input device, for receiving user input;
an active ontology, comprising representations of concepts and relationships among concepts;
a language interpreter component, for interpreting the received user input to derive a representation of user intent, wherein the language interpreter component obtains representations of concepts and relationships among concepts from the active ontology to derive the representation of user intent;
an output processor component, for rendering output based on the derived representation of user intent, and further based at least in part on a current output mode; and
an output device, for outputting the rendered output.

53. The automated assistant of claim **52**, wherein the language interpreter component parses the received user input based on the representations of concepts and relationships among concepts from the active ontology to derive the representation of user intent.

54. The automated assistant of claim **52**, wherein the language interpreter component disambiguates among at least two possible interpretations of the user input, based on the representations of concepts and relationships among concepts from the active ontology.

55. The automated assistant of claim **52**, further comprising:

an active input elicitation component, for generating at least one prompt to actively elicit input from a user;
wherein the output device outputs the generated at least one prompt.

56. The automated assistant of claim **55**, wherein the active input elicitation component generates the at least one prompt based on representations of concepts and relationships among concepts obtained from the active ontology.

57. The automated assistant of claim **55**, wherein the active input elicitation component automatically completes the user input based on representations of concepts and relationships among concepts obtained from the active ontology.

58. The automated assistant of claim **52**, further comprising:

a services orchestration component, for calling at least one service for performing a task based on the derived representation of user intent;
wherein the services orchestration component identifies services to be called based on the representations of concepts and relationships among concepts from the active ontology.

59. An automated assistant operating on a computing device, the assistant comprising:

an input device, for receiving user input;
a language interpreter component, for interpreting the received user input to derive a representation of user intent;
an active input elicitation component, for:
generating at least one prompt to actively elicit input from a user via a conversational interface; and
generating a paraphrase of the user intent;
an output processor component, for:
summarizing a plurality of results based on the derived representation of user intent; and

generating output representing the summarized plurality of results, and further based at least in part on a current output mode; and

an output device, for outputting the paraphrase of the user intent and for outputting the generated output.

60. The automated assistant of claim **59**, wherein:

the input device comprises a speech input device, for receiving spoken input;

the output device further outputs text representing the spoken input.

61. A method for implementing an automated assistant on a computing device having at least one processor, the method comprising:

at an input device, receiving user input;

at a processor, interpreting the received user input to derive a representation of user intent;

at the processor, identifying at least one domain, at least one task, and at least one parameter for the task, based at least in part on the derived representation of user intent;

at the processor, calling at least one service for performing the identified task;

at the processor, rendering output based on data received from the at least one called service, and further based at least in part on a current output mode; and

at an output device, outputting the rendered output.

62. The method of claim **61**, further comprising:

at the processor, generating at least one prompt to actively elicit user input; and

at the output device, outputting the generated at least one prompt.

63. The method of claim **61**, wherein receiving user input comprises:

prompting the user via a conversational interface; and

receiving user input via the conversational interface.

64. The method of claim **61**, further comprising:

paraphrasing the at least one representation of user intent; and

outputting, at the output device, the paraphrased representation of user intent.

65. The method of claim **61**, further comprising:

generating a plurality of alternative parse results for the received user input; and

disambiguating among the alternative parse results.

66. The method of claim **65**, wherein disambiguating among the alternative parse results comprises:

identifying at least two competing interpretations of the user input;

at the output device, outputting a prompt to request additional information from the user;

at the input device, receiving additional information from the user; and

resolving the ambiguity based on the additional information.

67. The method of claim **61**, wherein processing the received user input comprises processing the user input using at least one selected from the group consisting of:

data from a short term memory describing at least one previous interaction in a current session; and

data from a long term memory describing at least one characteristic of the user.

68. A computer program product for implementing an automated assistant on a computing device having at least one processor, the method comprising:

a nontransitory computer-readable storage medium; and

computer program code, encoded on the medium, for causing at least one processor to perform the steps of:

receiving user input;

interpreting the received user input to derive a representation of user intent;

identifying at least one domain, at least one task, and at least one parameter for the task, based at least in part on the derived representation of user intent;

calling at least one service for performing the identified task;

rendering output based on data received from the at least one called service, and further based at least in part on a current output mode; and

outputting the rendered output.

69. The computer program product of claim **68**, further comprising computer program code for causing the at least one processor to perform the steps of:

generating at least one prompt to actively elicit user input; and

outputting the generated at least one prompt.

70. The computer program product of claim **68**, wherein the computer program code for receiving user input comprises computer program code for causing the at least one processor to perform the steps of:

prompting the user via a conversational interface; and

receiving user input via the conversational interface.

71. The computer program product of claim **68**, further comprising computer program code for causing the at least one processor to perform the steps of:

paraphrasing the at least one representation of user intent; and

outputting, at the output device, the paraphrased representation of user intent.

72. The computer program product of claim **68**, further comprising computer program code for causing the at least one processor to perform the steps of:

generating a plurality of alternative parse results for the received user input; and

disambiguating among the alternative parse results.

73. The computer program product of claim **72**, wherein the computer program code for disambiguating among the alternative parse results comprises computer program code for causing the at least one processor to perform the steps of:

identifying at least two competing interpretations of the user input;

at the output device, outputting a prompt to request additional information from the user;

at the input device, receiving additional information from the user; and

resolving the ambiguity based on the additional information.

74. The computer program product of claim **68**, wherein the computer program code for processing the received user input comprises computer program code for causing the at least one processor to perform the steps of processing the user input using at least one selected from the group consisting of:

data from a short term memory describing at least one previous interaction in a current session; and

data from a long term memory describing at least one characteristic of the user.

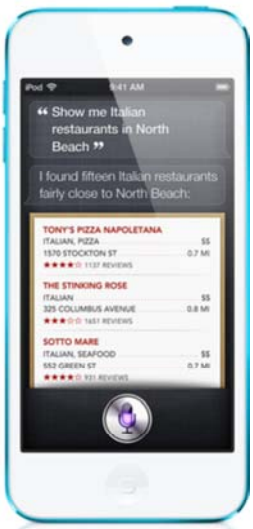
* * * * *

EXHIBIT D

iOS

Siri. Beta
Your wish is its command.

Siri lets you use your voice to send messages, schedule meetings, place phone calls, and more.* Ask Siri to do things just by talking the way you talk. Siri is so easy to use and does so much, you'll keep finding more and more ways to use it.



It understands what you say. And knows what you mean.

Talk to Siri as you would to a person. Say something like "Tell my wife I'm running late" or "Remind me to call the vet." Siri not only understands what you say, it's smart enough to know what you mean. So when you ask "Any good burger joints around here?" Siri will reply "I found a number of burger restaurants near you." Then you can say "Hmm. How about tacos?" Siri remembers that you just asked about restaurants, so it will look for Mexican restaurants in the neighborhood. And Siri is proactive, so it will question you until it finds what you're looking for.

It helps you do the things you do every day.

Siri makes everyday tasks less tasking. It figures out which apps to use for which requests, and it finds answers to queries through sources like Yelp and

Siri Frequently Asked Questions

Read answers to common questions and learn how Siri works. See the Siri FAQ

WolframAlpha. It plays the songs you want to hear, gives you directions, wakes you up, even tells you the score of last night's game. All you have to do is ask.

Know the score.

Ask Siri for baseball, basketball, football, hockey, and soccer scores as well as schedules, rosters, and stats.

Choose a movie.

Ask Siri to get showtimes, buy tickets from Fandango, look up movie facts, play trailers, show you reviews, and more.

Find a restaurant.

Ask Siri to search by different criteria or a combination. Siri gets you photos, reviews, and reservations.

iOS takes dictation.

Here's another amazing way to get things done: just use your voice. Instead of typing, tap the microphone icon on the keyboard. Then say what you want to say and Siri listens. Tap Done, and your words are converted to text. Use dictation to write messages, take notes, search the web, and more. Dictation also works with third-party apps, so you can update your Facebook status, tweet, or write and send Instagrams.

Eyes free.

Apple is working with car manufacturers to integrate Siri into select voice control systems. Through the voice command button on your steering wheel, you'll be able to ask Siri questions without taking your eyes off the road. To minimize distractions even more, your iOS device's screen won't light up. With the Eyes Free feature, ask Siri to call people, select and play music, hear and compose text messages, use Maps and get directions, read your notifications, find calendar information, add reminders, and more. It's just another way Siri helps you get things done, even when you're behind the wheel.

*Siri is available in Beta only on iPhone 4S, iPhone 5, iPad with Re ina display, iPad mini, and iPod touch (5th genera ion) and requires Internet access. Siri may not be available in all languages or in all areas, and features may vary by area. Cellular data charges may apply.

Some features may not be available for all countries or all areas. Click here to see complete list.

EXHIBIT E

Search Results

siri Search

About 69 results found for 'siri'.

iPhone 5
It's so thin and so light, yet iPhone 5 features a larger display, a faster chip, the latest wireless technology, an 8MP iSight camera, and more.
Learn more

Siri FAQ
Get answers to all your questions about Siri on iPhone 4S.
Learn more

iOS 6 - Use your voice to do even more with Siri.
Siri on Apple iPhone, iPad, and iPod touch lets you use your voice to send messages, make calls, set reminders, and more. Just speak naturally. Siri understands.
http //www.apple com/ios/siri/

iOS - Siri Frequently Asked Questions
Siri is the intelligent personal assistant that helps you get things done just by asking. Use your voice to send messages, schedule meetings and more.
http //www.apple com/ios/siri/siri-faq/

iOS 6 Feature Availability
Find out what features of iOS 6 are available on iPhone, iPad, and iPod touch in your country
http //www.apple com/ios/feature-availability/

iPhone 5 - Learn some helpful iPhone tips and tricks.
iPhone is capable of some pretty incredible things. Things you might not even know. Here are some tips and tricks you'll master in no time.
http //www.apple com/iphone/iphone-5/tips/

iPhone 4S - Learn some helpful iPhone tips and tricks.
iPhone is capable of some pretty incredible things. Things you might not even know. Here are some tips and tricks you'll master in no time.
http //www.apple com/iphone/tips/

iPhone 5 - The best of everything. Built right in.
Find your way in Maps. Shoot 8MP photos and 1080p HD video. Talk to Siri. Make FaceTime video calls. Browse the web. Check email. And more.
http //www.apple com/iphone/built-in-apps/

iPad mini - Features
iPad mini is the full iPad experience. It has a vivid display, it's fast and fluid, it works with apps made for iPad, and you can hold it in one hand.
http //www.apple com/ipad-mini/features/

iOS 6 - iPhone, iPad, and iPod touch get 200+ new ...

Apple Store Results

iPhone 4S 16GB White
$549 00

iPhone 4S - Unlocked (GSM) - 16GB Black
$549 00

iPhone 4S 16GB White
$549 00

iPhone 4S 16GB Black
$549 00

iPhone 4S 16GB Black
$549 00

View more Apple Store results

iTunes Results

Mega Snake
Tibor Takács
Horror

American Bandits Frank and Jesse Jame...
Fred Olen Ray
Western

Headshot
Pen-Ek Ratanaruang
Foreign

Chocolate (2008)
Prachya Pinkaew
Action & adventure

Sesame Street Elmo's Christmas Countd...
Gary Halvorson
Kids & family

View more results in iTunes

iOS 6 free software update has all-new Maps, more Siri, Facebook integra ion, and over 200 other new features that keep it the most advanced mobile operating system.
http //www.apple com/ios/whats-new/

iPad - Features
iPad with Retina display now features an A6X chip, FaceTime HD camera, and faster Wi−Fi. All of which make iPad capable of more than you ever imagined.
http //www.apple com/ipad/features/

iOS 6 - All-new, Apple-designed Maps with 3D views.
Maps gives you turn-by-turn spoken directions, interactive 3D views, and Flyover. All in a vector-based interface that scales and zooms with ease.
http //www.apple com/ios/maps/

iPhone 5 - It's so much more. And so much less, too.
It’s so thin and so light, yet iPhone 5 features a larger display, a faster chip, the latest wireless technology, an 8MP iSight camera, and more.
http //www.apple com/iphone/features/

iPod touch - Built-In Apps
There's an app to play music, an app to text a friend, an app to browse the web, even an app to get more apps. And it's all built into iPod touch.
http //www.apple com/ipod-touch/built-in-apps/

iPad mini - Built-in apps
iPad mini can do thousands of things a tablet PC or e-reader can’t. And because of Multi−Touch, it does things better than your computer ever could.
http //www.apple com/ipad-mini/built-in-apps/

iPad - Built-in apps
iPad can do thousands of things a tablet PC or e-reader can’t. And because of Multi−Touch, it does things better than your computer ever could.
http //www.apple com/ipad/built-in-apps/

iOS 6 - The world's most advanced mobile operating ...
With hundreds of amazing features, iOS 6 free software upgrade is the world’s most advanced and easy-to-use mobile operating system.
http //www.apple com/ios/what-is/

iPad - iOS
With its easy-to-use interface, amazing features, and rock-solid stability, Apple’s mobile operating system, iOS, is the foundation of iPad.
http //www.apple com/ipad/ios/

iPhone 5 - Read all about the amazing iOS.
Read about iOS 6. And its easy-to-use interface, amazing features, and rock-solid stability.
http //www.apple com/iphone/ios/

iPod touch - iOS
With its easy-to-use interface, amazing features, and rock-solid stability, Apple’s mobile operating system, iOS, is the foundation of iPod touch.
http //www.apple com/ipod-touch/ios/

iPad mini - iOS
With its easy-to-use interface, amazing features, and rock-solid stability, Apple's mobile operating system, iOS, is the foundation of iPad mini.
http //www.apple com/ipad-mini/ios/

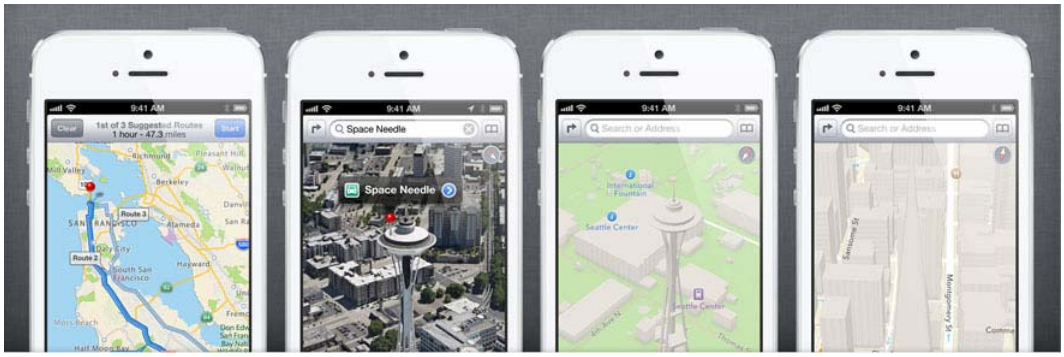
iPod touch - Features
Grab a panorama shot, ask Siri to update your Facebook status, play some music, video call your BFF, and slay some baddies. iPod touch is loads of fun.
http //www.apple com/ipod-touch/features/

Prev 1 2 3 4 Next

EXHIBIT F

iPhone 5 Tips and Tricks

These simple tips and tricks will help you travel from one destination to the next, get more done with Siri, and make the things you do every day even easier.



Get there with Quick Route.

When Maps drops a pin on a location, you can get turn-by-turn directions to that location almost instantly. Just tap the Quick Route button next to the location's name.

See up close with Flyover.

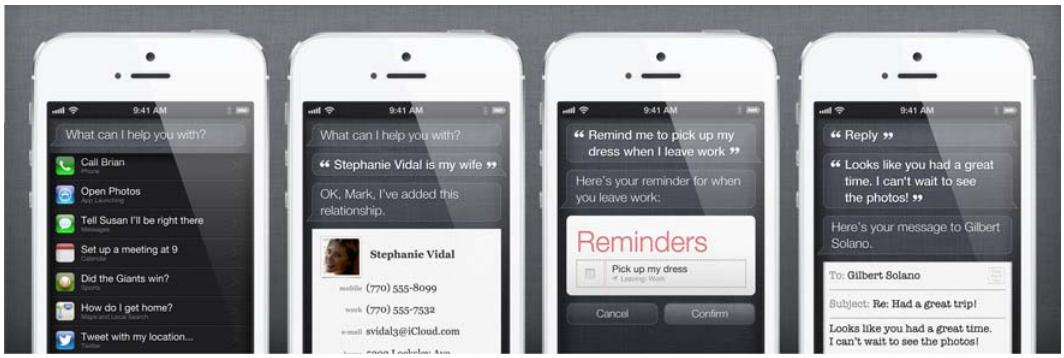
Switch to Flyover view in Maps by tapping the 3D button. Now you can explore cities from the air as you zoom, pan, and rotate around landmarks.

Tilt and rotate with two fingers.

As you're looking at a map, use two fingers to tilt or rotate the view. Maps keeps the names of the streets and places where they belong. So everything's easy to read, and you won't get lost.

Head north in a tap.

Rotate a map a few times, and you could lose your north-south orientation. But it's easy to get it back. Just tap the compass icon in the upper-right corner.



See how to use Siri.

When you press and hold the Home button, Siri asks, "What can I help you with?" Tap the "i" button to see a detailed list of the ways Siri can help you get things done.

Tell Siri who's who.

Tell Siri about your relationships, such as "Erin is my wife" or "Rick is my dad." Then you can say "Text my wife" or "Call dad" and Siri knows who you mean.

Set your locations.

Be sure to enter your home and work addresses in Contacts. That way, Siri can remind you to do things when you leave or arrive at either place.

Siri writes email.

You can dictate a quick response to an email. Just say "Reply," then tell Siri your message.

iPhone 5 takes dictation.

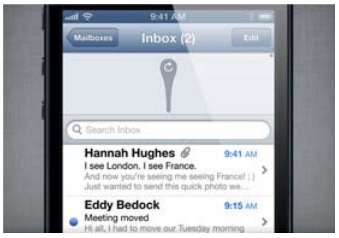
Instead of typing an email, tap the microphone icon on the keyboard and start talking. Tap Done, and your words will appear as text. Use dictation to write messages, take notes, update your Facebook status, tweet, and more.

Quick, take a photo.

The camera icon is always there on your lock screen. Just swipe up to open the Camera app.

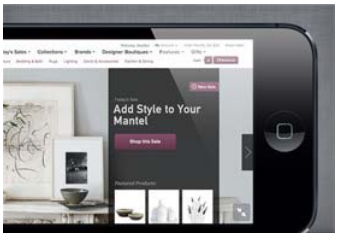
Switch directions in Panorama.

In panorama mode, you can easily shoot from left to right or right to left. Just tap the arrow to switch the direction.



Refresh your inbox with a swipe.

Anticipating an important email? Refresh your mailboxes in an instant with a quick downward swipe.



Take websites full screen.

When you really want to see the whole picture, turn your iPhone to landscape and tap the full-screen icon to view web pages without distractions.

See what's up.

Notification Center lets you know about missed messages, calendar invitations, friend requests, and more. New notifications appear on top of the screen. Swipe down to see a summary of recent notifications.

Set reminders.

Assign due dates to items on your to-do list and Reminders will send you alerts. Add locations, and you'll get alerts when you leave or when you arrive. You can also choose priority levels and write notes.

Tweet it. Post it.

Sign in once under Settings, and you can tweet or update your Facebook status directly from Safari, Photos, Camera, and Maps. Or ask Siri to do it for you.

Broadcast live with AirPlay.

With AirPlay Mirroring, you can share exactly what's on your iPhone 5 with your HDTV connected to an Apple TV. Just double-click the Home button, swipe all the way to the right, and select AirPlay Mirroring.

Customize your keyboard.

Create your own personal dictionary, including shortcuts for each word. So your keyboard not only autocorrects, it knows exactly what you want to say as you type. In Settings, tap General > Keyboard > Add New Shortcut. From there you can add new phrases and assign optional shortcuts to them.

Create an event fast.

Touch anywhere on the calendar to create an event. To change your schedule, just drag events around. Tap for day, month, or list view. And rotate to landscape to see an entire week.

Everything has an alert.

Assign a specific tone or create a custom vibration for new mail, Calendar alerts, tweets, and reminders. In Settings, tap Sounds to assign each a tone or vibration.

Locate your lost iPhone.

If you've lost your iPhone, Find My iPhone can help you locate it. Just go to Settings > iCloud and turn on the Find My iPhone feature.

Save images from the web.

In Safari, touch and hold an image to save it to your Camera Roll or copy it to paste into an MMS or email.

Give your texts some character.

Access all sorts of smileys, animals, shapes, and other peculiarities from your emoji-enabled keyboard. Just go to Settings > General > Keyboard and tap International Keyboards.

Get some peace and quiet.

In Settings, turn on Do Not Disturb to avoid dealing with all incoming calls and notifications. Enable it manually or schedule a recurring time. Or allow calls from your favorites or specific contact groups.

Scroll to the top fast.

In Safari, Mail, Contacts, and many other apps, tap the status bar — which displays the network information, time, and battery level — to scroll quickly to the top.

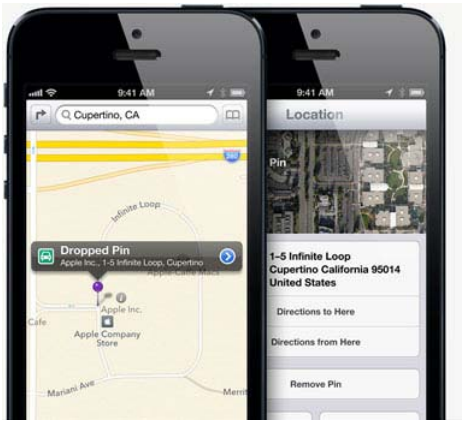
Add a new keyboard and select Emoji. Now when you type, just tap the globe button and choose the emoji that fits the mood.

Lock the screen orientation.

Double-click the Home button to bring up the multitasking bar, then swipe from left to right. Now tap the portrait orientation lock once to turn it on and again to turn it off.

Drop a pin.

In Maps, touch and hold anywhere on a map to drop a pin so you can find an address, get directions, or share it with a friend for a meet-up.



Pinch to zoom.

In the Camera app, pinch to zoom in and out of a scene up to 5x. You can also use the zoom slider.

Add PDFs to iBooks.

From a Mail message or a web page, touch and hold the PDF icon or link, then select "Open in iBooks."

Lighten the load. In your wallet.

Passbook stores your boarding passes, gift cards, coupons, and more. So when you're boarding at the gate or checking out at the café, just open Passbook, find the pass you need, and have it scanned — right from your iPhone.

Create web clips.

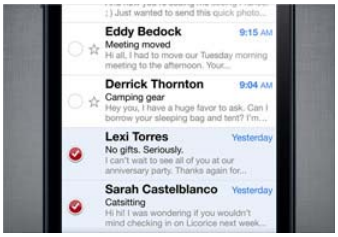
To add a website to your Home screen, visit the page in Safari and tap the Share button at the bottom of the Safari window. Then tap Add to Home Screen.

Learn some keyboard tricks.

- Tap the space bar twice, and iPhone adds a period and capitalizes the next word.
- To enter a number or symbol quickly, touch and hold 123, then select the key you want. Lifting your finger returns you to the alphabet keyboard.
- Touch and hold a letter to reveal a list of special characters.

Scrub through audio and video.

When you're watching a video or listening to music or a podcast, the scrubber bar lets you skip to any point along the timeline. You can adjust the scrub rate from high-speed to fine scrubbing by sliding your finger down as you drag the playhead along the scrubber bar.



Keep your inbox clean.

In Mail, you can delete or move messages in batches. From your inbox, tap Edit, select the messages you want to organize, then tap Delete or Move.

Take a picture of your screen.

Press and hold the Sleep/Wake button, then press the Home button. Your screen flashes and the picture appears in your Camera Roll.

Create a playlist.

In the Music app, tap Playlists, then tap Add Playlist and give it a name. Now tap any song or video to add it to the playlist. You can add individual songs, entire albums, or all songs by an artist.

Tap to focus the camera.

While shooting video or photos, tap the screen where you want to focus. iPhone also adjusts the exposure and white balance automatically.

Display character count in text messages.

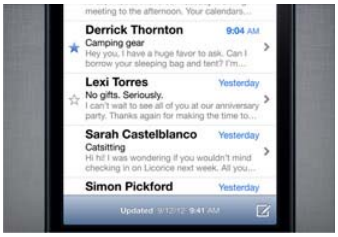
In Settings, tap Messages, then tap the Character Count switch. The count appears as you type when your message exceeds two lines. You may want to do this when carrier fees apply.

Print wirelessly from iPhone.

AirPrint makes it easy to print email and web pages from your iPhone to your AirPrint-enabled printer. To print an email, just tap the Reply icon and select Print. To print a web page, tap the Share button and select Print. You can also print photos, documents, and more.

Share from Notification Center.

Notification Center not only shows you what's up with you, it can tell your social circle what's up with you. Tweet or update your Facebook status right from Notification Center.



Get back to your draft.

In Mail, touch and hold the Compose button to switch to your list of saved message drafts.

Read the iPhone User Guide.

For more tips, tricks, and instructions, tap the Bookmarks icon in Safari, then select iPhone User Guide.

Some features may not be available for all countries or all areas. Click here to see complete list.