The SystemX Natural Language Interface: Design, Implementation and Evaluation

Nick Cercone, Paul McFetridge, Fred Popowich,

Dan Fass, Chris Groeneboer, Gary Hall

CSS-IS TR 93-03

Centre for Systems Science
Simon Fraser University
Burnaby, British Columbia
CANADA V5A 1S6

November 1993
(revised January 1994)

Abstract

SystemX is a natural language interface (NLI) to relational databases. The system has been used with a customer service statistical database, a research grants database, and an environmental database. Distinctive features of the NLI are its use of Head-Driven Phrase Structure Grammar (HPSG), treatment of complex noun phrases, integration of natural language with other input modes, use with a statistical database, and field-testing and evaluation in a real world application.

1.0 Introduction

Natural language interfaces (NLIs) allow people to communicate with machines in a natural language such as English. One application of NLIs is as “front-ends” to databases, enabling non-technical people to directly access information stored in databases. Advantages of NLIs as front-ends include relieving users of the need to know the structure of the database ([20]); relieving users of the need to know a database manipulation language, allowing users to express queries directly without a transformation or mapping to some other representation ([5], [32]); and making the system more convenient and flexible for users ([20]).

SystemX is an NLI to relational databases. Questions containing ordinary English are translated into database queries expressed in SQL (short for Structured Query Language), the standard computer language for manipulating relational databases. SystemX is different from other NLIs for the following reasons.

- Statistical and non-statistical databases are used.

SystemX's main target is a statistical database called RCIOPS owned by Rogers Cablesystems, a major Canadian cablevision company. The database describes, among other things, Rogers’ customer service operations, and underlies the company’s executive information system (EIS). Statistical databases and EISs are common in larger companies because executives are typically interested in summary information. The Rogers EIS provides on-line access to pre-formatted reports which are built using statistics derived from summarizing the company’s operational data.

SystemX has also been used as a front end to a statistical research grants database from NSERC (the National Science and Engineering Research Council of Canada) and a non-statistical environmental database from Environment Canada.

- Head-Driven Phrase Structure Grammar (HPSG) and a special-purpose semantics is used.

The parser of SystemX — its natural language understanding component — uses the HPSG formalism [23] to build semantic representations using a notation that is closely tied to the structure of relational databases. The semantic representations are then translated into a formalism-neutral logical form and then into SQL.

- Natural language is integrated with other input methods.

Input to SystemX is via a menu which allows queries as (a) natural language sentences, (b) menu items, or (c) a combination of menu items and fragments of natural language.

- Development is for an industrial application using real-world field-testing and evaluation.

SystemX is being developed to provide freer, less restricted access to Rogers’ RCIOPS database than the access provided by their current EIS, particularly the customer service parts of the database. SystemX is the result of extensive consultation with several potential users within Rogers. Through extensive interviews with the Vice-President of Customer Service for Rogers’ Western division, we have learned a significant portion of the sublanguage used by the cablevision industry to describe its customer service operations. We discovered that complex compound nouns were used extensively in the sublanguage and have developed a sophisticated treatment of them using HPSG and the NLI's semantics. Further extensions to SystemX, also influenced by the interaction with Rogers, are currently being programmed. The improvements include a better user interface and a distributed client/ server architecture.

The remainder of the paper is organized as follows. Certain key design issues behind our NLI are presented in section 2. Aspects of the developed system are described in sections 3-5. Section 3 describes SystemX's architecture. Section 4 shows an extended example of use of the system. Section 5 outlines our field test and its results. Section 6 reiterates the main features of systemX, discusses its merits, and offers some concluding remarks.

2.0 Design Issues

The team developing SystemX is committed to a view of design as an evolutionary process. This view has been articulated by Dasgupta [8] in the following way (see Figure 1). Design begins with a problem specified as a set of requirements which are often imprecise, incomplete, or both. Critical tests are devised to decide whether the design fits the requirements. Identification of a misfit between design and requirements may indicate the need for modifications to the design or requirements. Another critical test is devised to decide whether the (possibly new) design fits the (possibly modified) requirements. The process is repeated until a fit is found between requirements and design. However, requirements may change or a misfit may surface at any time in the future. In this sense design is an evolutionary process and evaluation, viewed as critical testing and misfit identification, is an inextricable part of the process.

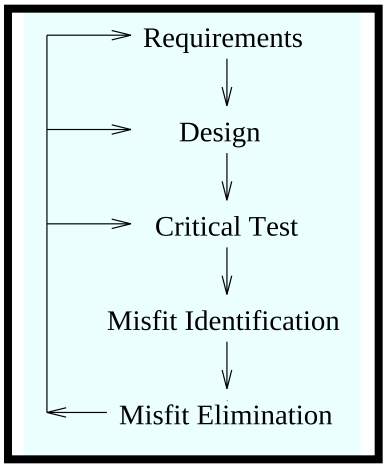


FIGURE 1. The design process.

This section describes various design concerns behind the evolution of SystemX. Section 2.1outlines a number of prior theoretical concerns; Section 2.2describes influential properties of the main application of SystemX to the Rogers RCIOPS statistical database. The discussion of evaluation issues will be deferred to Section 5.

2.1 Theoretical Concerns

2.1.1 Positive Aspects of the First System X

The current SystemX is the second natural language interface to a relational database developed at Simon Fraser University. The first NLI — also called System X ([6], [18], [19]) but spelt as two words *System* and *X* — had quite extensive coverage of English and could handle passives, imperatives, possessives, relative clauses, prepositional phrases and quantification. The old version of System X contained many positive design issues and features which have been incorporated into the new SystemX. These issues and features are: the provision of ad hoc access and portability, and a proven system architecture.

Ad hoc access. Natural language allows users to formulate queries using a form of communication very familiar to them with a freedom unmatched by other input modes such as menus. See [4], [5], [33] for discussion of the merits of natural language as an interface mode.

Portability. NLIs can be made more portable in a number of ways (see [7],[12], [13],[21]). The most important way is promoting easy attachment of databases from different domains. The original System X focussed on portability between databases and attempted to minimize the knowledge that humans must provide by maximizing the knowledge that the system can discover for itself by analyzing the database (see [6]). The same view is adopted in the new SystemX.

Proven system architecture. System X was of a modular design, containing a natural language understanding module and a database query module. The natural language understanding module had three components: a lexical analyzer (which analyzed words), a parser (which did syntax processing), and a semantic interpreter (which did semantic interpretation and produced canonical semantic representations). The database query module contained a component that translated the canonical semantic representations into a logical form (LF). A second component translated the logical form into SQL. Different versions of this component would translate the logical form into other database languages. The same basic architecture is used in the new SystemX.

2.1.2 Use of Head-Driven Phrase Structure Grammar and New Semantics

One major difference between the old and new versions of SystemX is that the natural language understanding module has been updated with a more modern grammar formalism and a new semantics. Head-Driven Phrase Structure Grammar (HPSG), the new grammar formalism, is one of the best known “unification-based” grammar formalisms. Unification-based grammars are grammars whose central operation is unification, a powerful matching technique. Two descriptions unify if they do not contain conflicting information and the result of unification is a description containing all the information from both descriptions.

Although HPSG borrows freely many positive features from other previous unification-based grammars, it has a number of distinctive features. For one, a uniform knowledge representation — attribute-value matrices — are used to represent lexical entries and grammar rules and principles. For another, the grammar is head-driven, i.e., the head constituent of a phrase is central. There is evidence that head-driven parsing strategies are more efficient than other parsing strategies [2].

A semantic representation which makes use of feature structures to describe database relations [16] is incorporated directly into the HPSG feature structures used by the parser. So syntactic and semantic processing can be done in parallel. The HPSG semantic representations are then converted into logical forms which are then passed to an adapted version of the database query module from the old System X.

2.2 Properties of the Application

The application chosen for SystemX was to provide ad hoc access to the RCIOPS database underlying the Rogers Cablesystems' EIS, particularly, its customer service information. The EIS provides on-line access to pre-formatted reports built from statistics derived from summarizing the company's operational data. SystemX provides on-line, direct access to the statistical database on which the preformatted reports are based. This application has been a big influence on the design of SystemX in three main ways: its task domain (Section 2.2.1), database domain (Section 2.2.2), and linguistic domain (Section 2.2.3).

2.2.1 The Task Domain

Knowledge about the task domain was obtained through interviews with Rogers' Vice-President of Customer Service for its Western Region, and other company personnel. The VP was selected as target user for the system. In these interviews, we learned about the company's business operations, and its use of databases and information retrieval. Presently, graphs containing key monthly statistics must be requested from Rogers' database people in Toronto. It often took three weeks for the requested information to come back graphed. By then, our user would sometimes have forgotten why he had asked for the information! With SystemX, not only can its users obtain the information they want in minutes, they can follow up on some interesting-looking trend or anomaly there and then, without having to wait another three weeks for feedback!

Some of the graphs the VP requested from Toronto never vary, so we opted to provide these as "canned" menu items. In these interviews, we were told about the kinds of queries the VP would like to formulate, notably, what kinds of statistics he sought and the forms of location and time restrictions he needed. Through the interviews, we came to see that SystemX should provide not only natural language on its own, but also natural language integrated with menus so that our VP could build certain requests through a combination of menu selection and typed natural language.

2.2.2 The Database Domain

The Rogers RCIOPS database is a statistical database, that is, each table in the database contains one or more *category attributes* (columns) whose values define sets of entities of a single type, and one or more *statistical attributes* (columns) whose values summarize those sets. Some of these statistical attributes are grouped together as *statistical tables.*

There are six subdomains within the customer service domain. Ordered in terms of complexity from simplest to most complex, they are: manpower, plant, subscribers, outages, service calls and telephone. Figure 1 shows these six subdomains and, in the case of the service calls and telephone subdomains, further major subareas within them.

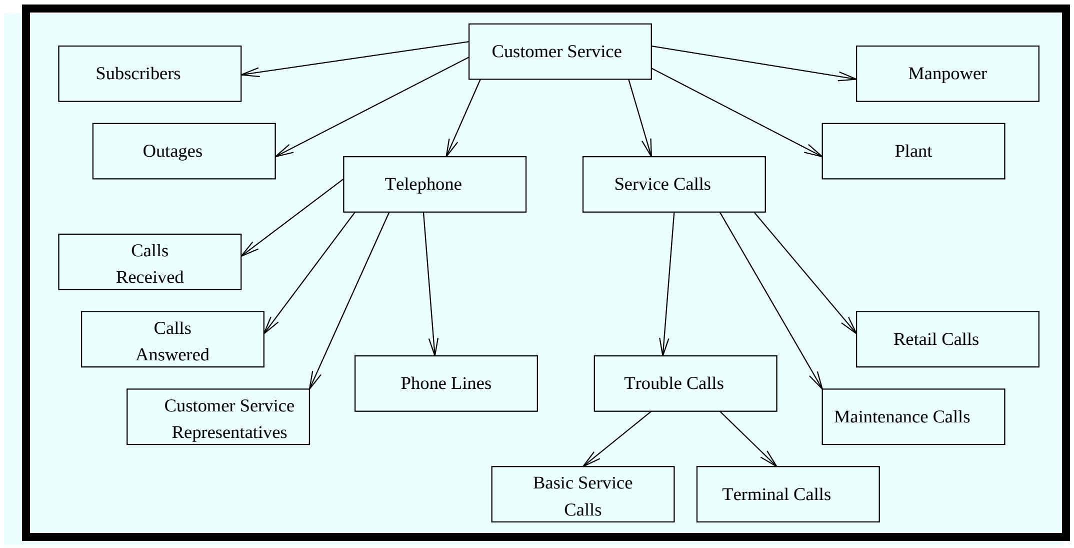


FIGURE 2. The RCIOPS cablevision customer service domain.

Two category attributes appear in every statistical table in the RCIOPS database: `S_Date` and `Location`. The `S_Date` attribute refers to time information. The `Location` attribute refers to codes that represent all of Canada, two regions (Western and Eastern), and individual divisions (such as Vancouver, part of the Western Region). The subscribers, outages, service calls and telephone subdomains all have their own subdomain-specific category attributes, which we shall come to shortly.

Plant. The simplest subdomain is plant which consists of one statistical table, `ACMPLANT`, containing one statistical attribute, `Size_Kms`, and no subdomain-specific category attribute.

Manpower. The next simplest is manpower which also consists of one statistical table, ACMMAN, again containing just one statistical attribute, `Man_Count`. However, there is a subdomain-specific category attribute, `Man_Code`, which refers to two types of employee.

Subscribers. The subscribers subdomain, the next most complicated, has one statistical table (`ACMSUBS`) containing two statistical attributes and one subdomain-specific category attribute. The two statistical attributes are `Subs_Bud`, which refers to the number of cable subscribers budgeted for a time-frame, and `Subs`, the number of subscribers served during that time-frame. The subdomain-specific category attribute, Subs_Code, divides subscribers into *basic* and *terminal* categories.

Outages. The outages subdomain has one statistical table (`ACMREL`), three statistical attributes and one subdomain-specific category attribute. The three statistical attributes are `Outages` (number of outages), `Outage_Mins` (number of minutes of outage), and `Fails_Corrected` (percentage of failures corrected within four hours). The subdomain-specific category attribute, Outage_Code, divides outages into *scheduled* and *unscheduled* ones.

Service calls. The above four subdomains only have one statistical table each. The remaining subdomains, service calls and telephone, each have multiple statistical tables. The service calls subdomain contains three statistical tables, each containing two statistical attributes. There are also two subdomain-specific category attributes. The three tables are ACMSERVC, ACMSERVCGRP and ACMSERVCTOT which offer views of the service call subdomain at three levels of granularity. The finest level of granularity is 26 individual types of service call (ACMSERVC). The intermediate level is five groupings of service calls (ACMSERVCGRP) shown in Figure 1: basic service calls, maintenance calls, trouble calls, terminal calls, and retail calls. The coarsest level is all service calls collectively (ACM-SERVCTOT).

The subdomain-specific category attribute for ACMSERVC is Servcall_Code, which contains codes for the 26 types of service calls such as *subscriber (sub) owned equipment (eqp) faults* and *drop problems outside.* Correspondingly, ACMSERVCGRP has the category attribute Servcgp_Code, which contains codes for the five groupings of service call, such as *basic service calls* and *maintenance calls.* ACMSERVCTOT has no subdomain-specific category attribute.

The statistical attributes for the three tables all refer to the same kinds of statistics, either the number of service calls or a ratio of the number of service calls per 1000 basic service subscribers. For ACMSERVC, the statistical attributes are called Servcall_Count and Servcall_Ratio. For ACMSERVCGRP, they are called Servc_Grp_Count and Servc_Grp_Ratio. For ACMSERVCTOT, they are called Tot_Servc_Count and Tot_Servc_Ratio.

As will be seen in the next section (Section 2.2.3), the sublanguage used to describe the three levels of granularity, the codes and the counts and ratios, causes a number of linguistic problems.

Telephone. The telephone subdomain has the most statistical tables and statistical attributes: four tables containing two, four, four, and six statistical attributes. These tables cover the four subareas shown in Figure 1: phone lines, customer service representatives, phone calls received and phone calls answered. There is one subdomain-specific category attribute, shared by all four tables, called Office_Code, which refers to two types of telephone office, *business* and *repair.*

The table ACMTELSLINE contains statistics about telephone lines into a particular office within two attributes, Fone_Lines (number of phone lines) and Trunk_Busy (number of hours trunk lines were busy).

ACMTELSCSR holds statistics about the customer service representatives (also known as CSRs) who answer the phones. The information is held within the four attributes Csr_Stats (number of representatives), Csr_Hours (number of hours representatives answered phones), Csr_Hours_Pr (number of hours paid/required), and No_Ft_Eqv (number of full-time equivalent representatives).

ACMTELSREC contains statistical information about phone calls received at a particular office. The four statistical attributes for the table are Calls_Rec (number of calls received), Calls_Rec_Bud (number of calls budgeted to be received), Vru_Abnd (number of calls abandoned while interacting with a voice response unit), and On_Hold_Abnd (number of calls abandoned while waiting to speak to someone).

Finally, the table ACMTELSANS has statistics about phone calls answered (at a particular office), held with the six attributes Calls_Ans (number of phone calls answered), Avg_Talk (average number of seconds called talked), Avg_Wait (average number of seconds called waited), Avg_Work (average number of seconds representatives worked after a call), Pct_20_Sec (percentage of phone calls answered within a 20 second standard), and `Pct_180_Sec` (ditto but for 180 seconds).

The sublanguage used in the telephone subdomain, like the service call subdomain, produced a host of linguistic problems, described in the next section.

2.2.3 The Natural Language Domain

During the interviews conducted with users about the task domain, they were asked how they would frame queries in natural language, if that facility were available. From their answers a basic corpus of 90 sentences was built. The corpus, which appears in Appendix A, shows a sublanguage for the cablevision customer service subdomain, and other subdomains such as accounts receivable.

Much of the sublanguage consisted of compound noun phrases that were technical and terse but lengthy nonetheless, often containing acronyms and abbreviations. For example, compound noun phrases included *the western region outage log summary*, *the co owned eqp fault breakdown ratios*, and *the TSR calls answered within 20 seconds.* Acronyms included *TSR* (short for *technical service representative*) and *VRU* (for *voice response unit*). The many abbreviations included *sub* (for *subscriber*), *co* (for *company*), and *eqp* (for *equipment)*.

Much work went into figuring out the meanings of words and phrases. Some of this was done in consultation with our interviewees. However, because checking every detail of the cablevision sublanguage would have been too time-consuming for our interviewees, we mapped out natural language phrases for referring to some of the available statistics, especially in the service call and telephone subdomains. We found that a number of words had multiple senses, for example, the word *calls* had different meanings in the service call and telephone subdomains, and the word *count* had different meanings in all six subdomains.

It rapidly became apparent that the main problem was to analyze the compound noun phrases while keeping down the sense ambiguity of words and hence the number of lexical entries in SystemX's lexicon. The analysis of compound nouns and sense ambiguity was complicated because interpretation was not always a matter of straightforward decompositional meaning. These problems are not just a property of the structure of the RCIOPS database, they are related to the general linguistic problems of *compositionality* and of *ambiguity.* In a strict compositional approach to natural language semantics, the meaning of a complex constituent would be composed of the meanings of the parts. Such an approach is far from adequate for our needs, and far from adequate for natural language in general. Let us now look at some specific classes or problematic constructions that were encountered in our corpus. In the following examples, ***bold-italic*** indicates the word that changes the meaning of the phrase.

Problem 1 (Non-compositionality): An extra word changes the statistical attribute(s) or statistical table.

This is a non-standard form of composition since the meaning of two words is formed by one meaning *overwriting* that of the other, rather than their meanings *combining.* An extra word can change the meaning of a phrase in one of two ways: (a) the addition of a new word can override the prior meaning of a phrase and (b) a word can have a meaning that must be suppressed in the presence of a prior word. This happens quite frequently in the telephone subdomain where tables often contain statistical attributes that have similar

English descriptions:

TSR hours	`(ACMTELSCSR Csr_Hours)`
TSR hours ***projected***	`(ACMTELSCSR Csr_Hours_Proj)`
calls answered	`(ACMTELSANS Calls_Ans)`
calls answered ***within 20 seconds***	`(ACMTELSANS Pct_20_Sec)`
calls answered ***within 180 seconds***	`(ACMTELSANS Pct_180_Sec)`
calls received	`(ACMTELSREC Calls_Rec)`
calls received ***(against) budget***	`(ACMTELSREC Calls_Rec_Bud)`

A different but related problem happens in the outage and subscriber subdomains because the words *outage* and *subscriber* refer both to the name of the subdomain and to certain statistical attributes within the subdomain. The problem is to figure out which use is which and control the meanings of words so that the correct interpretation is produced each time. Listed below are the interpretations we assigned to various phrases from the outages subdomain (the subscribers subdomain has very similar phrases):

outages = *outages count*	`(ACMSREL Outages)`
outages ***corrected*** = ***corrected*** *outages*	(`ACMSREL` Fails_Corrected)
outages ***scheduled*** = ***scheduled*** *outages*	(`ACMSREL` Outage_Code 10)
outages ***statistics***	(`ACMSREL` 3 statistical attributes)

Yet another related problem can be found in the service call subdomain. The presence of *basic* suppresses the meaning of *service call*: and changes the statistical table.

service call ...	`(ACMSERVCTOT)`
basic *service call ...*	`(ACMSERVCGRP Servc_Grp_Code 10)`

Problem 2 (Ambiguity, Non-compositionality): One word or phrase refers to two (or more) statistical tables, and an extra word or phrase selects the meaning.

This is common in both multiple-table subdomains, service calls and telephone. In the service call subdomain, a number of lexical items refer to two tables of different levels of granularity: `ACMSERVCTOT` (coarse-grained: all service calls collectively), `ACM-SERVCGRP` (medium-grained: five intermediate groupings of service calls) and `ACM-SERVC` (fine-grained: 26 categories).We interpret the phrase *service call*, in the absence of any other words, to refer to the medium-grained `ACMSERVCGRP` table or the fine-grained `ACMSERVCTOT`. Likewise, phrases like *basic service call, maintenance call,* and *retail call*, in isolation, can refer to either one of the five medium-grain groupings of service calls (`ACMSERVCGRP`) or the individual types of service call falling within that group (`ACM-SERVC`).

In the telephone subdomain, we interpret *tsr*, when paired with *performance* (and other such general words), as referring to all statistical attributes that concern customer service representatives. These attributes are drawn from three different tables.

The ambiguity of granularity of phrases in these subdomains is resolved through the presence (or absence) of certain words, often the last in a phrase. We interpret the attachment of the word *breakdown(s)* to a phrase to select the finer-grained of the two tables referred to by the phrase. So, for example, *trouble call breakdown* is interpreted as referring to the ACMSERVC table, rather then the coarser-grained ACMSERVCGRP.

Conversely, the words *total(s)*, or *summar*y, or the absence of any word, is taken to indicate the coarser-grained of two tables. Here is a summary of what has just been said:

service call	***total(s)***	coarse	ACMSERVCTOT
service call	***breakdown(s)***	medium	ACMSERVCGRP
trouble call	***total(s)***	medium	ACMSERVCGRP
trouble call	***breakdown(s)***	fine	ACMSERVC
drop problem outside	***breakdown(s)/total(s)***	fine	ACMSERVC

Likewise, the presence of *performance* in *tsr performance* indicates a reference to three tables whereas the presence of *hours* in *tsr hours* indicates a reference to a statistical attribute (Csr_Hours) from just one table (ACMTELSANS).

Problem 3 (Ambiguity): One word refers to a variety of category attribute information.

The phrases of the service call subdomain mentioned above refer not only to two different tables, but to two different category attributes for codes within those tables. The code values, however, are always the same, e.g., *maintenance call* refers to the table ACM-SERVCGRP and its Servc_Grp_Code value, which is 20, and also ACMSERVC and its Acmserve_Code value, which is also 20.

A different kind of example is the word *sub* (actually, an abbreviation) which occurs in two subdomains, subscribers and service calls. In the subscriber subdomain, it has one basic meaning. However, in the service call subdomain, it has at least three distinct meanings in *sub owned*, *sub education* and *sub not home*, referring to three different values of the category attribute Servcall_Code:

sub owned eqp faults	(ACMSERVC Servcall_Code 1)
sub education	(ACMSERVC Servcall_Code 4)
sub not home	(ACMSERVC Servcall_Code 7)

A further example, which occurs with the category attribute S_Date, is the word *last*, which has five different temporal meanings in the following phrases:

for last year = for the year prior to the current one.
for the last N years = all of the last -12xN months.
for (last) July = the s_date information from July = the July in the last 12 months.
for last month = the month prior to today.
*for the last N month*s = all of the last -1xN months.

Problem 4 (Ambiguity): Flexible adjunct attachment needed.

Some of the compound nouns we encountered were as long as twelve words, e.g., *the may 1992 eastern divisions sub owned eqp fault count total stats*. It became clear that we needed to assign some internal structure to the compounds in order to do further analysis. Furthermore, we needed a flexible approach to adjunct attachment that would allow the same adjunct as both premodifier and postmodifier. This was especially apparent with words that represent the category attributes S_Date and Location, e.g., *the Calgary outages*, *the outages for Calgary, the 1992 outages* and *the outages in 1992*. A flexible approach was also needed to handle the many alternative ways of requesting the same statistical attributes(s) in natural language. This can be seen below in the many synonymous expressions using the phrase *trouble calls*:

trouble calls = trouble call total(s) = trouble call activity total(s) = total(s) of trouble calls

`(ACMSERVCGRP Servc_Grp_Count Servc_Grp_Ratio)`

trouble call counts = trouble call count total(s)

`(ACMSERVCGRP Servc_Grp_Count)`

trouble call ratios = trouble call ratio total(s)

`(ACMSERVCGRP Servc_Grp_Ratio)`

trouble call activity breakdown(s) = breakdown(s) of trouble calls = breakdown(s) of trouble call activity

`(ACMSERVC Servcall_Count Servcall_Ratio)`

trouble call count breakdown(s) = breakdown(s) of trouble call counts

`(ACMSERVC Servcall_Count)`

trouble call ratio breakdown(s) = breakdown(s) of trouble call ratios

`(ACMSERVC Servcall_Ratio)`

Problem 5 (Non-compositionality): Percolation of adjunct information through compound noun phrases required.

In all the cases outlined above, it became apparent that we needed special techniques to percolate code values through lengthy compound noun phrases. Category attribute information — `S_Date, Location` and any other codes — must be passed through a complex nominal. Hence in *the Calgary outages* and *the outages in Calgary,* the `S_Date` code value (2800) must be passed through the phrase.

Likewise, in situations where an extra word changes the statistical attribute(s), such as *TSR hours* `(Csr_Hours)`and *TSR hours* ***projected*** `(Csr_Hours_Proj),` care must be taken that the information `Office_Code 20` contributed by *TSR* is not lost.

2.3 Our Approach

Given our theoretical concerns, and the practical concerns with the data, domain and tasks, we sought a solution based on the early version of System X. Within the HPSG formalism, we developed solutions to the syntactic and semantic problems which could then be incorporated into the whole natural language system. Although we used HPSG and a specific statistical database in designing the natural language interface, many of our techniques are applicable to other databases and other formalisms as we will see in our discussions in subsequent sections.

3.0 Implementation

SystemX is implemented on a Sun SPARC Station running SunOS, OpenWindows, Sicstus Prolog and Common Lisp. Figure 3 gives a high level graphic view of SystemX, which consists of a series of modules: HPSG-PL, Lisp Parser, TreeTool, LF conversion, semi-automatic construction of application-dependent knowledge bases, Oracle, and Gnuplot. The system's interface, described in Section 3.1, allows the user to input queries via a combination of natural language and menu selection. The interface accepts queries from

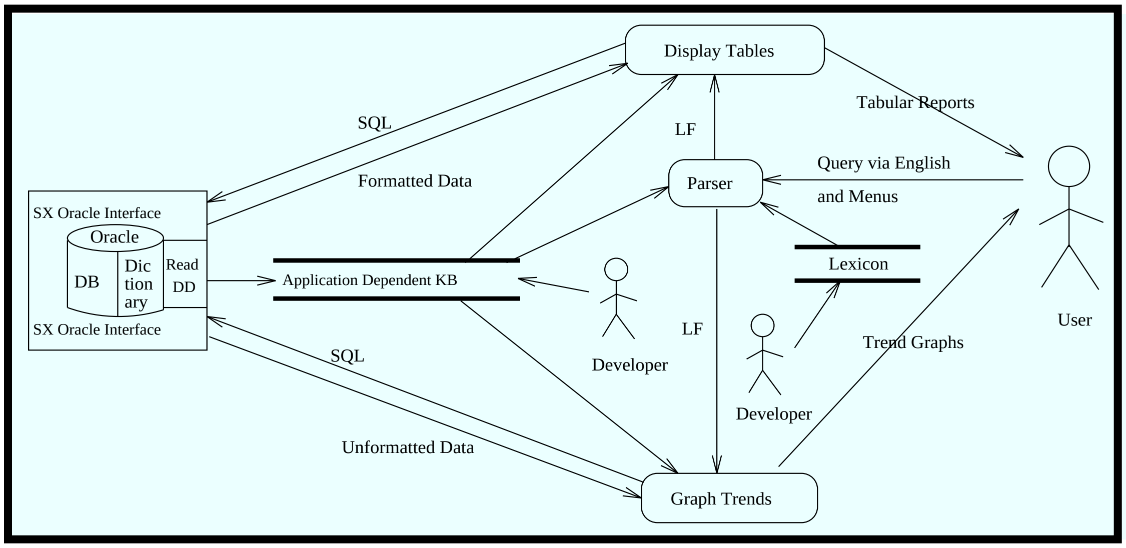


FIGURE 3. Overview of SystemX.

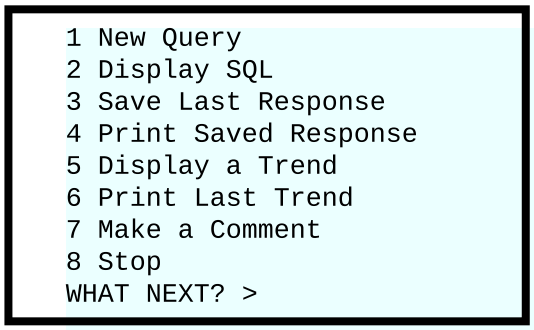
the user (on the right side of the figure) specified wholly or partly in English, translates them into SQL (via an intermediate language called LF), retrieves data from the database, and displays the data in the format (table or graphed trend) specified by the user in the query. The reports are obtained by typing a full query in natural language, including the statistic sought for a number of number of business units (or divisions) over a specified time period. Graphs are currently obtained by consulting a menu where natural language can be used to determine the base statistic.

The interface is customized to a given application via the creation of a *Lexicon* of English expressions related to the domain, used by the parser, and via the creation of an *Application Dependent Knowledge Base (ADKB)* containing information about the target (Oracle) database. The ADKB is constructed semi-automatically, with information extracted from the data dictionary of the database as well as information supplied by individuals familiar with the domain.

Natural language queries are processed by the parser, described in Section 3.2, which translates the query into a semantic representation. Section 3.3 describes the translation of semantic representations into logical form (LF), and Section 3.4 explains the translation of LF into SQL. Section 3.5 describes the system's output as tables and graphs.

3.1 The Interface

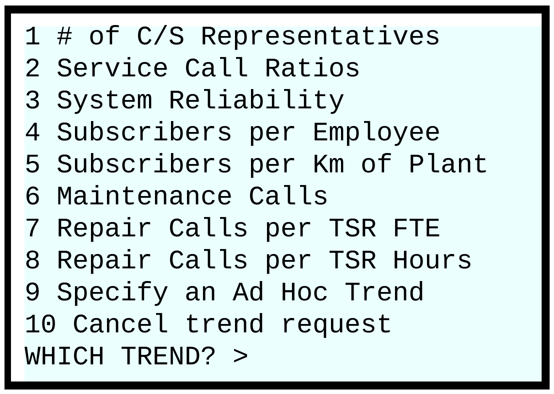
The main components of the interface are a main menu and a trend menu. Figure 4 shows the main menu. "New Query" is selected when natural language query input and tabular presentation of data is desired. "Display SQL" displays the SQL code generated by the most recent "New Query." "Save Last Response" saves the most recent table of data, and "Print Saved Responses" prints saved tables of data. "Display a Trend" brings up the trend menu and presents data in line-graph form. "Print Last Trend" prints the most recent trend. "Make a Comment" opens the on-line commenting facility. "Stop" is used to exit SystemX.



```
1 New Query
2 Display SQL
3 Save Last Response
4 Print Saved Response
5 Display a Trend
6 Print Last Trend
7 Make a Comment
8 Stop
WHAT NEXT? >
```

FIGURE 4. Main menu.

Figure 1 shows the trend menu which is selected through “Display a Trend” in the main menu. Users can elect to specify a statistic for graphing in natural language (menu



```
1 # of C/S Representatives
2 Service Call Ratios
3 System Reliability
4 Subscribers per Employee
5 Subscribers per Km of Plant
6 Maintenance Calls
7 Repair Calls per TSR FTE
8 Repair Calls per TSR Hours
9 Specify an Ad Hoc Trend
10 Cancel trend request
WHICH TREND? >
```

FIGURE 5. Trend menu.

item 9: “Specify an Ad Hoc Trend”) or choose among a set of canned queries (items 1-8 in the trend menu).

3.2 Natural Language to Semantic Representation

Natural language is translated into a semantic representation by a parser. To do this translation, the parser needs a description of the grammar for input queries, complete with a lexicon or dictionary which contains the words allowed in the input. The semantic representation is then converted to a logical form, and finally to SQL (see Section 3.3 and Section 3.4). The behaviour of the parser is summarized in Figure 6.

Two parsers have been developed, one written in Lisp by Paul McFetridge ([16]), and the other written in Prolog by Fred Popowich, Carl Vogel, and Sandi Kodric ([15], [27]). The two parsers are used to test competing ideas, which are sometimes easier to implement and test in one language than the other. Both parsers are chart parsers. Chart parsing is a type of parsing in which all syntactic structures which are built are placed on a single graph structure called a chart. A successful parse of a sentence is a list of edges and nodes that includes the start node of the chart, all the words in the sentence, and the end node.

3.2.1 Head-Driven Phrase Structure Grammar and Unification-Based Semantics

Head-Driven Phrase Structure Grammar (HPSG) is used to represent the linguistic

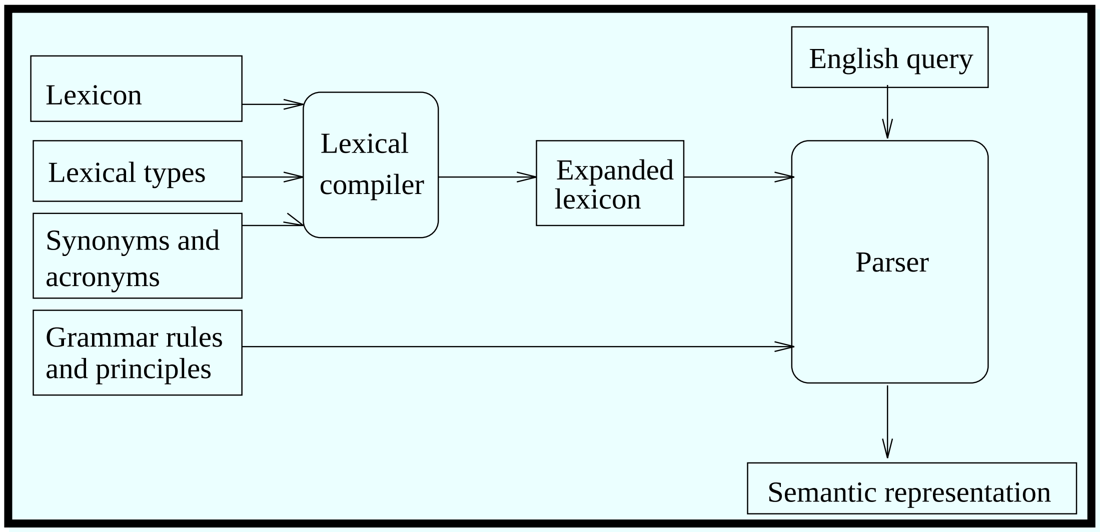


FIGURE 6. The parser.

knowledge needed by the system. HPSG is head-driven and uses four different classes of syntactic constituents: *head, adjunct*, *complement* and *filler.* An example of a head-adjunct structure is *scheduled outages* where *scheduled* is the adjunct and *outages* is the head (*scheduled* is here a nominal adjunct). A second example head-adjunct structure is *show manpower for calgary* where *for calgary* is either a verbal adjunct and *show manpower* is a head, or a nominal adjunct where *manpower* is the head. Complements include subjects, objects, determiners, verbal complements, and sentence complements. Heads share their head features with the mother. Complements discharge subcategorization requirements on the head. Fillers discharge binding requirements on the head (binding features provide information about long-distance dependencies for relative pronouns, interrogative expressions, and "missing elements" often called gaps or traces).

HPSG is considered a highly lexical grammar formalism since the vast majority of linguistic knowledge is represented in the lexicon rather than in the grammar rules. In [23], for example, only four grammar rules are introduced for handling a wide variety of English constructions. Our parser requires four grammar rules to cover the kind of constructions encountered in our natural language queries, plus a collection of *principles* which apply in conjunction with all grammar rules.

HPSG relies on a highly structured lexicon in which most linguistic generalizations are captured. The lexicon makes use of inheritance mechanisms, similar to those found in Object-Oriented programming languages, to capture these relationships and reduce redundancy. Furthermore, the "lexical inheritance" processing can be done off-line, as is done in the system, before any parsing is attempted, to produce an "expanded lexicon" which can be used by the parser. Using the expanded lexicon, the parser can then combine lexical entries according to the grammar rules using unification as its basic operation.

If the information of the lexicon is organized in a multiple inheritance hierarchy, it is easy to modify/update the lexicon. The appropriate class need only be changed, and the modification will propagate to all the related subclasses. This greatly simplifies the process of designing a lexicon. It minimizes the duplication of information.

We have developed a unification-based semantics for HPSG to treat the linguistic problems described in Section 2.2.3. It became clear that unification alone would not be sufficient to handle the decompositional problems described, so we developed a number of techniques for treating them: a context-selection and a priority mechanism. Much of the semantics resides in the lexicon; the rest is in some principles.

We shall describe HPSG and the semantics by considering first the lexicon (Section 3.2.2), then the grammar rules and principles (Section 3.2.3), paying special attention to the unification-based semantics. Although our discussion will focus on HPSG, our treatment of semantic information is also applicable to other unification-based formalisms (Section 3.2.4)

3.2.2 The Lexicon and Lexical Types

The lexicon consists of a *multiple inheritance hierarchy* of classes called *lexical types* (adopting the terminology of [23]). We call it a multiple inheritance hierarchy since a class may have more than one superclass. So, an instance of a class will inherit not only the information associated with its class, but also all the information associated with all of its superclasses. We do not consider hierarchies in which there is conflicting information. Issues related to the treatment of conflicting information and defaults are discussed in [15].

Each lexical type is represented by a *feature structure.* A feature structure, or equivalently an attribute value matrix or a directed acyclic graph, consists of a set of *feature value* pairs where the features are atomic and the values are either atomic or other feature structures.

The relationship between the different classes and subclasses is defined in terms of the subsumption relation for feature structures. The multiple inheritance hierarchy can be viewed as a *subsumption lattice*. Informally, a feature structure *A* subsumes a feature structure *B* if *B* contains all the information of *A* (and possibly more). Subsumption is closely related to unification ([23], [30]) — one can be defined in terms of the other. The unification of two structures *A* and *B* is a structure *C* that contains all the information of *A* and all the information of *B*, and nothing more. In the case where *A* and *B* contain conflicting information, unification is said to *fail*. Alternatively, one can say that the unification of *A* and *B* results in a feature structure, $\perp$, that contains an inconsistency. All feature structures subsume $\perp$.

The feature structures used by HPSG have features for phonological (PHON), syntactic (SYN) and semantic (SEM) information. In lexical entries, lists of feature structures are allowed to appear as the value of specific features. The PHON feature has as its value a list containing the orthographic representation of a lexical entry.

- **Syntactic Information**

SYN contains the syntactic information associated with the lexical entry. The value of this feature is a feature structure containing local (LOC) and nonlocal information. Nonlocal information is concerned with long distance dependencies and need not concern us here. Local features in general specify inherent syntactic properties of an entry, such as part of speech, inflection, case, and subcategorization (including the traditional notion of government). LOC information is broken down into HEAD, SUBCAT and LEX features.

HEAD features contain intrinsic syntactic information about a constituent like its category, verb form (finite, past participle, etc.), and prepositional form. The HEAD feature called HEADS [26] contains a set of descriptions, one for each construction that can be modified by the adjunct. For example, the HEADS feature for an adjective will contain a sign for a noun.

The SUBCAT field contains a list of feature structures, one for each complement that the head may be combined with in a derivation. For example, since a noun may be combined with a determiner to produce a noun phrase, there will be a feature structure for a determiner on the SUBCAT list of a noun. Similarly, *graph*, an intransitive verb, subcategorizes for only one NP (the subject). *In* subcategorizes for two NPs, the prepositional subject and object.

LEX is simply a binary value for that is usually assigned the value + for entries in the lexicon, and - for phrases formed from lexical entries. Note that if a phrase appears in the lexicon, then it will have a + value for its LEX feature. Also, some compound constructions formed from lexical entries can still be considered to be lexical, as specified by the grammar rules of HPSG that will be introduced in Section 3.2.3.

- **Semantic Information**

SEM contains semantic information about lexical entries and is comprised of features for TYPE, TABLES, FUNCTION and REFERENT. TYPE contains basic syntactico-semantic information used to assign structure to compound nominals, and to control ambiguity as discussed in [26]. TABLES contains representations of statistical tables, category attributes and statistical attributes from the RCIOPS database. FUNCTION contains the priority mechanism used to tackle some of the decompositional problems outlined in Section 2.2.3. REFERENT fulfils two functions: it percolates adjunct information through compound noun phrases and it contains the context-selection mechanism used for tackling other decompositional problems.

TYPE information

TYPE information consists of functional type (FTYPE) and domain type (DTYPE) information, both arranged as hierarchies. The functional type hierarchy embodies certain aspects of the Rogers statistical database design that appear to be of linguistic significance. We have noticed that the complex noun phrases used in queries to the Rogers database can be broken down into nominals and nominal modifiers that refer to statistical tables, category attributes, and statistical attributes. Some examples were given in Section 2.2.3.

The nominals and modifiers can be grouped into six classes: statistical type (STYPE), statistical set (SSET), modifier set (MSET), entity set (ESET), modifier (MOD), and premodifier (PMOD). Their hierarchical relationships are shown in Figure 7. (The FTYPE hierarchy has been ported, with some modification, to other databases.)

`statistical type` (e.g., *summary, sum, count, average,* and *ratio*)

Nouns from this class are heads which denote types of statistical measure such as averages and ratios.

`statistical set` (e.g., *statistics, log, performance, activity,* and *problem)*

Nouns from this class may be semantically vacuous, that is, we assume that all requests are for *some* set of statistics and these nouns may not carry any information that can help identify the particular statistics sought by a user. Expressions of this type may be used to modify a **statistical type**, e.g., l*og summary, performance count,* and *activity ratios*.

`modifier set` (e.g., *faults, call*s, and *lines*)

Expressions in this class denote “entity types” — sets of which are described by the statistics in the database. An expression of this type (partially) specifies which statistics are

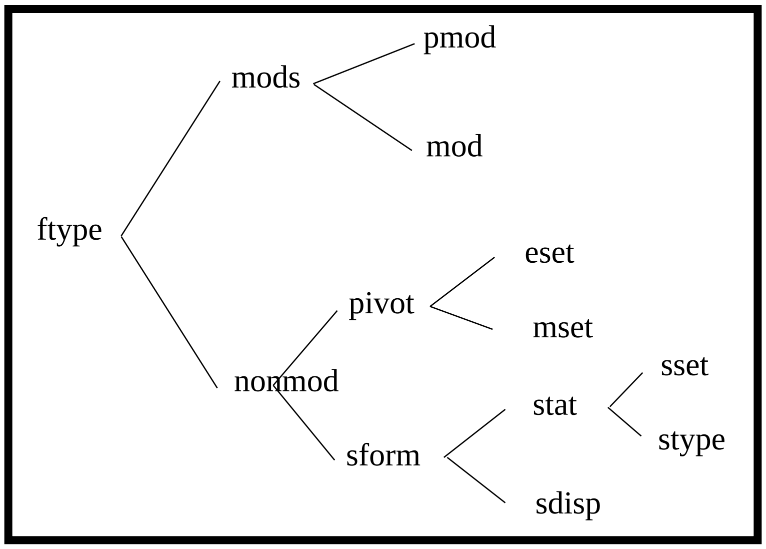


FIGURE 7. The functional type hierarchy.

being requested — either a statistical table or a statistical attribute — and can be viewed as a modifier of a **statistical set**, or a **statistical type**, though they generally require modification by an **entity set.**

`entity set` (e.g., *outage, reliability,subscriber,terminal,trouble,* and *csr*)

Expressions in this class also refer to either a statistical table or a statistical attribute and can be viewed as a modifier of a **modifier set,** or a **statistical set**, or a **statistical type** (e.g., *subscriber statistics, outage summary, trouble call performance,* and *phone call performance*).

`modifiers`(e.g., *region, division, service,* and *system*)

Expressions in this class refer to the attributes used to categorize entities into sets. Members of this class can be viewed as modifiers of (subclasses of) e**ntity sets**, e.g., *service call* and *system reliability*. There is a close correspondence between modifiers and category attribute names in the database.

`pre-modifiers`(e.g., *Western, Vancouver, basic,* and *terminal*)

Expressions in this class denote values of category attributes, and can be viewed as modifiers of (subclasses of) **modifiers**, e.g., *Western region, Western division,* and *basic service*.

A further class is **statistical display** (SDISP) words, concerned with how information is presented

The domain type hierarchy reflects the contents of the database. DTYPE information is used to describe the inherent semantic category of a lexical item and is also used in selectional restrictions. Part of the domain type hierarchy used in SystemX is shown in Figure 10. Some the hierarchy concerns number and the domain-general category attributes `S_Date` (time) and `Location`, but the largest part concerns the six database subdomains. This information about subdomains is used to isolate statistical tables and attributes. The densest parts of the hierarchy are those for the two multi-table subdomains, service calls and telephone.

TABLES information

The TABLES information consists of database information from a main table (MAIN)

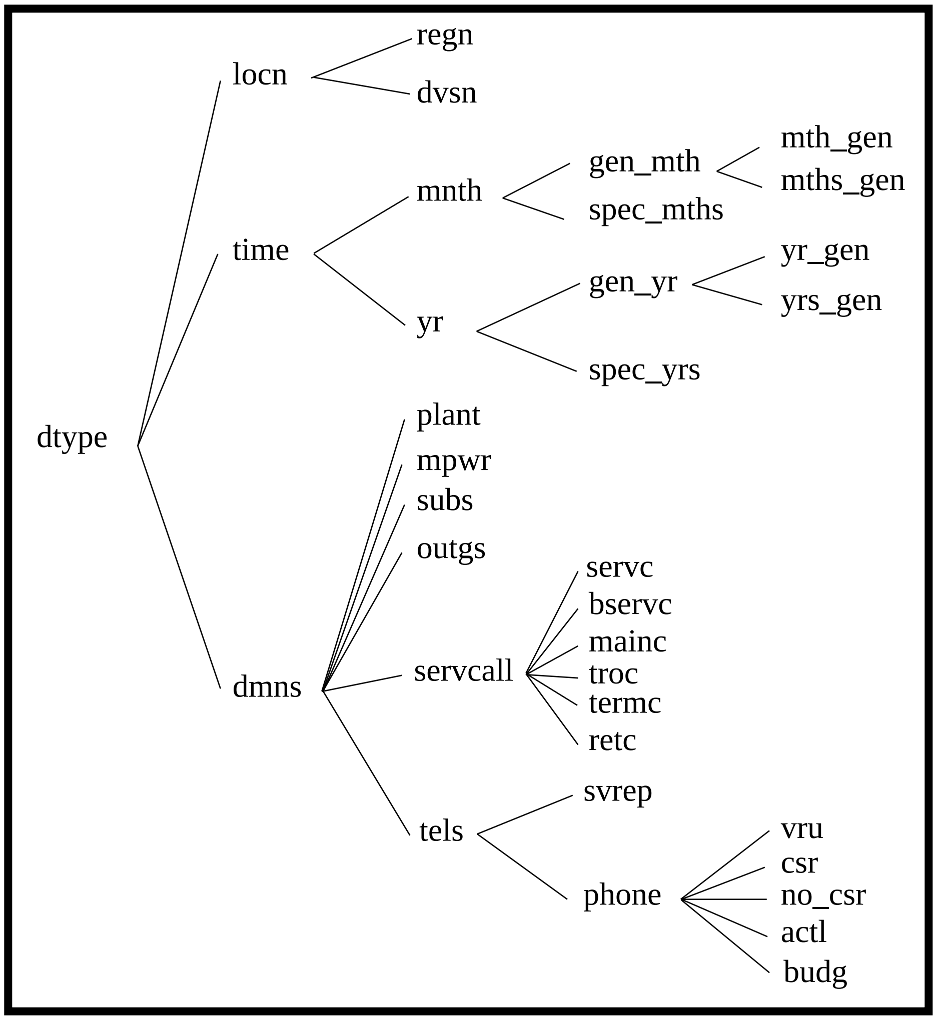


FIGURE 8. The domain type hierarchy.

plus a list of feature structures containing information from other tables (SUBTABLES). The feature structure associated with each table has an atomic valued feature for the table name, along with the COLUMNS feature which takes as its value a feature structure containing database column names and their values. This is illustrated in the feature structure shown in Figure 10 (based on Figure 6 from [7]). A "join" is indicated by identical variables prefixed with a "@" in the value of the LOCATION_CODE feature. NIL is a special "anonymous variable" which is used when the value is unknown but required.

FUNCTION information

We introduced the FUNCTION attribute for use by the Semantics Principle. The only possible values for this attribute are **priority** or **default**, but as with any other feature, its value may be left as unspecified. As will be seen in Section 3.2.3, the value of the FUNCTION feature will be used in a priority mechanism to allow meanings of adjuncts to override (or be overridden by) the meanings of their heads. The priority mechanism stems the proliferation of lexical entries and allows some meanings to be stronger than others, hence *projected* is given priority, thereby overriding *TSR hours* in *projected TSR hours*. With an adjunct as FUNCTION = **priority**, a standard HPSG treatment of the semantics of

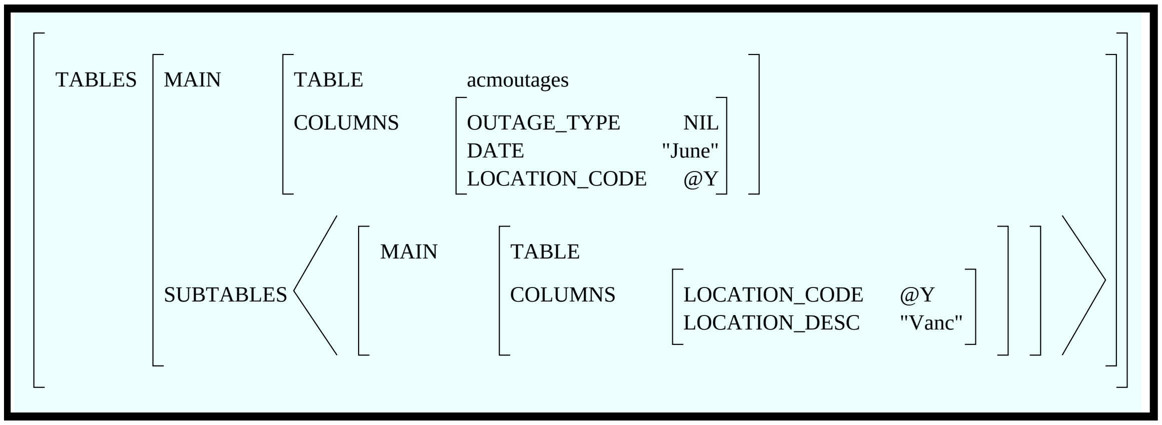


FIGURE 9. TABLE feature in a feature structure.

adjuncts can be used, i.e., the semantics of the mother is that of the adjunct. Similarly, if a head is FUNCTION = **default** and if the adjuncts TABLE feature is non-nil, then the semantics of the mother is that of the adjunct. In other words, if a head is FUNCTION = **default** then the adjunct is given priority. A head can also have FUNCTION = **priority** to appear either on an adjunct or on a head. Also, if a constituent is specified as FUNCTION = **priority** and it has a null (empty) semantics, then the unification fails.

REFERENT information

The REFERENT feature was originally intended to act much like the INDEX or INDICES feature introduced in [23]. However, it has evolved so that instead of representing a collection of semantic indices, it contains a collection of *potential* meanings associated with an expression. Recall that the *actual* meaning of an expression is contained within its TABLES feature. The value of a REFERENT feature is a feature structure where the feature names index the different possible meanings stored as the feature values.

The referent feature allows various possibilities contingent on the head which a nominal modifies. Each attribute in the referent corresponds to a statistical head. For example, *outage* has a REFERENT feature COUNT which has as its value the meaning of the expression *count* in the phrase *outage count*. As will be discussed in Section 3.2.3 in detail, this allows different information to be used, depending on context. This is used in the outages, subscribers, service calls and telephone subdomains.

Rather than write separate senses — and hence separate entries — for many words, we have written single entries, each containing a number of context-specific senses. There is one entry for *statistics*, which always combines with other words in the same way: its presence indicates that all table information should be given.

3.2.3 The Grammar Rules and Principles

- **Grammar Rules**

The grammar rules used in HPSG establish relationships between four different classes of syntactic constituents introduced in Section 3.2.1: *head (H), complement (C), adjunct (A)* and *filler* (F). Specifically, the rules state how the different information in one constituent is related to its sister and mother constituents in a derivation tree. Two grammar rules are used for combining heads with complements and two for combining heads with adjuncts. The grammar currently does not use any fillers. The grammar rules, which are shown below, are discussed in detail in [26].

$$[\text{SUBCAT} \ < [\,] >] \ \rightarrow \ H[\text{LEX} \ +, \ \text{INV} \ -], \ C^{*}$$

$$[\text{SUBCAT} \ < >] \ \rightarrow \ H[\text{LEX} \ -], \ C$$

$$[\text{SUBCAT} \ < [\ ] >, \ \text{LEX} \ \square] \ \rightarrow$$
$$H[\text{LEX} \ \square \ , A[\ \text{SUBCAT} < >, \ \text{LEX} \ +, \ \text{HEADS} \ <\ldots H \ldots >]$$

$$[\text{SUBCAT} < [\ ] >, \ \text{LEX} \ \square] \ \rightarrow$$
$$H[\text{LEX} \ \square \ ], A[\ \text{SUBCAT} < [\,] >, \ \text{LEX} \ \square \ , \ \text{HEADS} \ <\ldots H \ldots >]$$

The first rule combines a lexical head with everything but its final complement. It can combine a lexical head verb with all of its complements except its subject, and it can combine a preposition with its complement (object). This rule can also be used to convert a lexical head requiring only a single complement into a non-lexical constituent still requiring a single complement. The rule is used with lexical heads of numerous phrase types including verb phrases, common noun phrases, and prepositional phrases. This single rule captures the same generalizations as a set of traditional phrase structure rules which include the following:

VP $\rightarrow$ V, NP	VP $\rightarrow$ V, NP, NP
VP $\rightarrow$ V, PP	PP $\rightarrow$ P, NP
N’ $\rightarrow$ N	VP $\rightarrow$ V

The second rule combines a non-lexical head with its final complement. So it can combine a verb with its subject and a noun with its determiner, playing the role of traditional rules like:

S $\rightarrow$ NP, VP	NP $\rightarrow$ DET, N’

The third rule combines saturated lexical adjuncts ð— those with empty SUBCAT lists like adjectives, proper nouns and adverbs — with their heads. It can apply to either lexical or nonlexical heads, but the head must be unsaturated, specially having only one element on its subcategorization list. (Saturated adjuncts have empty SUBCAT lists and hence do not subcategorize for anything; nonsaturated adjuncts have something on their SUBCAT list.) The third rule thus combines adjectives with their heads, like in the phrase *western region*, and proper nouns with their heads as in *Vancouver outages*. Viewed in terms of traditional phrase structure rules, it can capture relationships like the following:

N $\rightarrow$ Adj, N	N’ $\rightarrow$ NP, N’

The final rule allows unsaturated heads to be modified by unsaturated adjuncts, like prepositional phrases and verb phrases. This rule requires the head and adjunct to either both be lexical, or both be nonlexical. Relationships captured by this rule include:

N’ $\rightarrow$ N’, PP	N’ $\rightarrow$ N’, VP
N’ $\rightarrow$ N’, N’	N $\rightarrow$ N, N
VP $\rightarrow$ VP, PP	

- **Principles**

Each of the rules, as presented above, is processed in conjunction with a collection of principles. Five principles are used. Four are taken from HPSG: the *Head Feature Princi-*

ple, *Subcategorization Principle*, *Constituent Ordering Principle*, and *Semantics Principle*. The first three principles are adopted unmodified; we have modified the Semantics Principle. The fifth principle is our own and is called the *Semantic Contribution Principle*.

Principles – syntactic

The Head Feature Principle, as described in [23], simply states that the HEAD feature of some compound constituent is identical to that of its head daughter in a derivation. The same feature structure is shared as the value of the HEAD feature in these two constituents.

The Subcategorization Principle states that the SUBCAT list of a compound constituent is formed by taking the SUBCAT list of its head daughter and removing from this list the feature structures that correspond (via unification) with its complement daughters. Thus the resulting constituent is still *looking for* those complements from its head daughters that have not yet been found.

The Constituent Ordering Principle is used to determine how the information contained in the PHON attributes of sister constituents is combined to form the phonology of the parent constituent. This principle makes use of *linear precedence* constraints which determine the left to right ordering of the different constituents. For our grammar, the only linear precedence constraints needed are that: (a) lexical heads precede their complements, (b) nonlexical heads follow their complements, (c) complements are ordered corresponding to their position on the SUBCAT list of the head daughter, (d) nominal or adjectival adjuncts precede their heads, and (e) verbal or prepositional adjuncts follow their heads. We shall return to discuss the Semantics Principle momentarily.

The interaction of all these principles along with the first three grammar rules can be seen in the derivation tree associated with the phrase *the unscheduled outages* in Figure 10. Notice that the head daughter (H) in every case shares the value of its HEAD feature with its mother (the Head Feature Principle). In cases where a compound constituent has no complements, the Subcategorization Principle ensures that they have the same SUBCAT lists. However, in the one case where we do have a complement daughter, notice that the parent's SUBCAT list does not contain the sign for this complement; the feature structure associated with the complement is unified with the corresponding sign on the subcategorization list of the head daughter.

In the above example, notice that there are two different classes of constructions: one in which a head is combined with zero or more complements (which we will call a *head-complement* structure), and one in which a head is combine with a single adjunct (a *head-adjunct* structure). This distinction is important for describing the relationship between the semantic information of a constituent and its daughters.

Principles – semantic

Our Semantics Principle describes how the REFERENT and TABLES information of a complex constituent is formed from that of its daughters. In a head-complement structure, the REFERENT of the complex structure is that of the head. In a head-adjunct structure, the REFERENT of the single adjunct is identified with the resulting constituent. If we were to adopt the treatment of semantic content proposed in [23] then we would get the same relationship for the REFERENT. The Semantics Principle of [23] (Chapter 8) states that the semantic content of a complex structure is "token identical" with that of the adjunct daughter in a head-adjunct structure, and with that of the head daughter otherwise. However, in determining the TABLES information of a complex structure, we desired

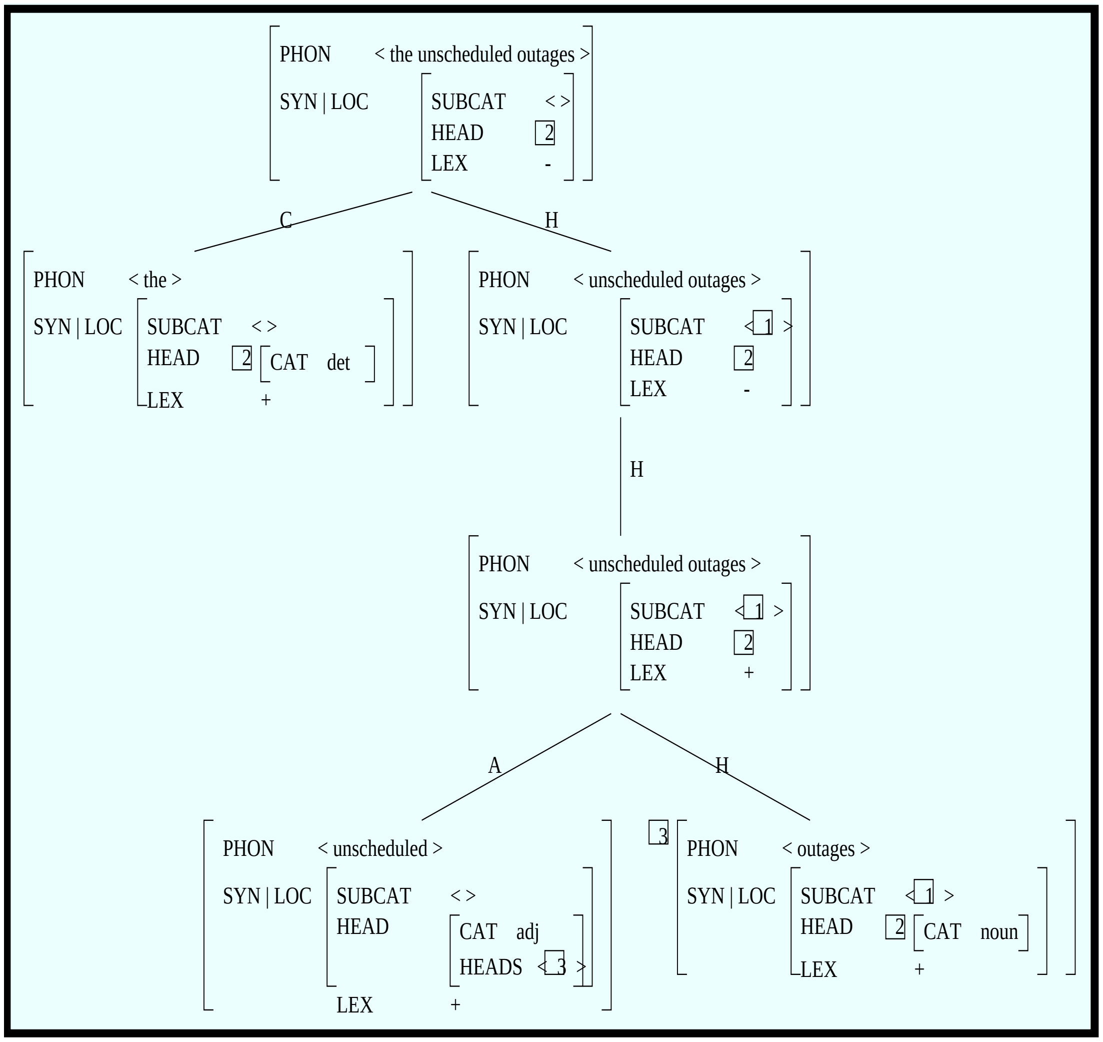


FIGURE 10. Interaction of grammar rules.

more flexibility than would be afforded by adopting Pollard and Sag's proposal.

The relationship between the TABLES informations of the complex constituent and its daughters depends on the values of FUNCTION feature on the daughters. It also depends on whether specific constituents have any information (in the way of a table name or a column value) to contribute to the resulting constituent. This additional dependency is reflected in the Semantic Contribution Principle, which was introduced in [26]. It requires the adjunct to have some semantic content in a head-adjunct structure; semantically void adjuncts are thus not allowed. An adjunct is considered to make some contribution as long as it has features for TABLE or for COLUMNS.

In a head-complement structure, the *table unify* procedure is used to combine the database information contained in all the tables of the constituents. However, we cannot just use simple unification of the corresponding feature structures. A more selective procedure is needed to combine all of the information from all of the constituents, making sure that the resulting semantic information is still meaningful with respect to the database.

Given two feature structures occurring as the value of the TABLES feature in two feature structures, we attempt to unify the two table structures that are the value of the MAIN feature in each of the two constituents. If one of the table structures is unspecified with respect to the value of the TABLE feature (the table name), then ensure that the columns in this underspecified table are a subset of those in the other table. If not, then do not allow them to be combined (this prevents us from creating structures which would be meaningless with respect to the database). If both tables do not have a table name specified or if they both do have a table name specified, then simply unify them. If the unification does not succeed, then try to find some pair of table structures in the TABLES or SUBTABLES that share some column name; look for a join. If a join is found, then all the tables and subtables are carried over into the SUBTABLES of the resulting constituent, with the MAIN table being the same as that of the head. The treatment of joins in general is discussed in [5] and [9].

In a head-adjunct structure, use is made of the FUNCTION feature for determining the value of the TABLES feature for the resulting constituent. Recall that the possible values for the FUNCTION feature are **default** or **priority**, or unspecified. The FUNCTION feature was introduced to stem proliferation of lexical entries and because we want some meanings to be stronger than others for situations.

The priority mechanism stems the proliferation of lexical entries and allows some meanings to be stronger than others. For example, the meaning of *projected* in *projected TSR hours* is given priority, thereby overriding the meaning of *TSR hours* to get the appropriate meaning as shown in Section 2.2.3. When an adjunct is FUNCTION = **priority**, a standard HPSG treatment of the semantics of adjuncts can be used, i.e., the semantics of the mother is that of the adjunct. Similarly, if a head is FUNCTION = **default** and if the adjunct has a value specified for its TABLE feature, then the semantics of the mother is that of the adjunct. In other words, if a head is FUNCTION = **default** then the adjunct is given priority. A complete description of how the FUNCTION feature is used in conjunction with the Semantics Principle is captured in the following algorithm which is used to determine the value of the TABLES feature for a complex constituent.

1. If the adjunct daughter has **priority**, then the TABLES information of the resulting constituent is token identical to that of the adjunct daughter. Note that this corresponds to the case covered by Pollard and Sag’s semantics principle.
2. If the head daughter has **priority**, then its TABLES information is used instead. Observe that if both constituents had **priority**, the previous case would have been selected.
3. If the adjunct is specified to be **default** and if the semantics of the head-daughter does specify a table name, then the TABLES information of the head is used.
4. If the head is **default** and if the adjunct specifies a table name, then the information from the adjunct is used.
5. Otherwise, *table unify* the TABLES information of all of the daughters.

Due to the nature of the parser’s grammar rules, in a head-adjunct structure it is possible for the adjunct to access all of the information contained in its head. This is because the adjuncts HEADS feature will contain a sign that is token identical to that of the head-daughter. Not only can the adjunct *examine* features associated with the head that it is modifying, but it can also *assign values* to features on the head. Depending on the context, the prospective adjunct may want to incorporate only selective information from its head, incorporate it into the adjunct's own semantics, and then use this as the semantics of the resulting constituent. The adjunct can do this by imposing a **default** value for the FUNCTION feature of its head. Conversely, in other cases the adjunct may not want its own semantic information to contribute toward the semantics of the resulting constituent. It can do this by specifying a **priority** value on its head.

Similarly, some lexical entries always override information of their sister constituents. These lexical entries can be specified as **priority** in the lexicon. Conversely, some lexical entries have a default interpretation which is always overridden when the entry is combined with some other constituent. Such entries will have a **default** value for their function feature. Note that the value of the FUNCTION feature is not passed up to the resulting constituent by any of its daughters; only the value of the HEAD feature is passed up from a head daughter to its mother. However, when an adjunct sets a value for the FUNCTION feature of its head, then the relationship is contained within the adjunct's HEADS feature, which is within the HEAD feature. Thus, this FUNCTION information can be passed up to more complex constituents indirectly.

When an adjunct overrides the meaning of its head, the head's category attribute information (`S_Date` and `Location` plus subdomain-specific category attributes) must be saved. This can be easily achieved as shown in the lexical type `dateloc` introduced in Figure 11. Note that the adjunct has access to all the information associated with its head

$$\begin{bmatrix} \text{SEM} & \begin{bmatrix} \text{FUNCTION} & \text{priority} \\ \text{TABLES} \mid \text{MAIN} \mid \text{COLUMNS} & \begin{bmatrix} \text{S\_DATE} & \boxed{1} \\ \text{LOCATION} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{SYN} \mid \text{LOC} \mid \text{HEAD} \mid \text{HEADS} & \left\langle \begin{bmatrix} \text{SEM} \mid \text{TABLES} \mid \text{MAIN} \mid \text{COLUMNS} & \begin{bmatrix} \text{S\_DATE} & \boxed{1} \\ \text{LOCATION} & \boxed{2} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

```
dateloc type @semsign with
 syn:loc:head:heads =
         [@semsign with
          sem:tables:main:columns:s_date = @var(D) with
          sem:tables:main:columns:location = @var(L)] with
 sem:tables:main:columns:s_date = @var(D) with
 sem:tables:main:columns:location = @var(L) with
 sem:function = priority.
```

FIGURE 11. Overrides and information saving

(including its semantic information). The adjunct shares the `S_Date` and `Location` information of its head as reflected in the coindexing variables 1 and 2, plus it specifies that its own semantics will take priority.

To deal with the overrides in a specific subdomain, the outages subdomain for instance, we define a lexical type `outage_override` which inherits the information from the `dateloc` lexical type and can then be used when selecting information from the head and overriding the information from the head. This lexical type is shown both as a feature structure and in terms of its HPSG-PL specification in Figure 12.

$$\begin{bmatrix} \text{SEM} & \begin{bmatrix} \text{FUNCTION} & \text{priority} \\ \text{TABLES | MAIN | COLUMNS} & \begin{bmatrix}\text{S\_DATE} & \boxed{1}\\ \text{LOCATION} & \boxed{2} \\ \text{OUTAGE\_CODE} & \boxed{3}\end{bmatrix}\end{bmatrix} \\ \text{SYN | LOC | HEAD | HEADS} & \left\langle \begin{bmatrix} \text{SEM | TABLES | MAIN | COLUMNS} & \begin{bmatrix}\text{S\_DATE} & \boxed{1}\\ \text{LOCATION} & \boxed{2} \\ \text{OUTAGE\_CODE} & \boxed{3}\end{bmatrix}\end{bmatrix}\right\rangle \end{bmatrix}$$

```
outage_override type @dateloc with
syn:loc:head:heads =
            [@semsign with
             sem:tables:main:columns:outage_code = @var(C)] with
sem:tables:main:columns:outage_code = @var(C).
```

FIGURE 12. Subdomain specific overrides.

When the semantic contribution of an adjunct is overridden by that of the head, it is more difficult for the head to access and preserve information associated with the adjunct. As reflected in our grammar rules, there is no direct means for a head to access semantic information (or really any information) associated with its adjunct. In the revised version of HPSG [23], cyclic feature structures are allowed and thus such bidirectional dependencies are allowed. However, there is an indirect means by which the head can access information. The adjunct can place (or instantiate) information into the head, which the head can then examine *knowing that it came from the adjunct*. We frequently use the REFERENT feature of the head to hold this information. Let us look at a specific example to illustrate how this contextual information is passed and used.

In our lexicon, there are three entries for *count* because it has a special meaning in the service calls subdomain, another special meaning in the telephone subdomain, and a third meaning for the four remaining subdomains, including outages. The contextual technique allows those four remaining subdomains to use a single entry. In the service calls subdomain, heads like *count, ratio* and *breakdown* look inside the COUNT, RATIO or BRKDOWN feature of the REFERENT feature structure that has been passed to them by adjuncts like *trouble call* or *service call*, and use the TABLES information as their own. These heads are specified as FUNCTION = **priority** so that their semantics are used instead of that of their adjunct. Furthermore, since these heads might themselves act as adjuncts to some subsequent head, these heads instantiate information within their own HEADS feature which is available for future potential heads.

Such dependencies are illustrated in the feature structures for the phrases *service call* in Figure 13. Notice that the REFERENT feature within the HEADS feature (i.e., the REFERENT of a constituent to be modified) has different TABLES information depend-

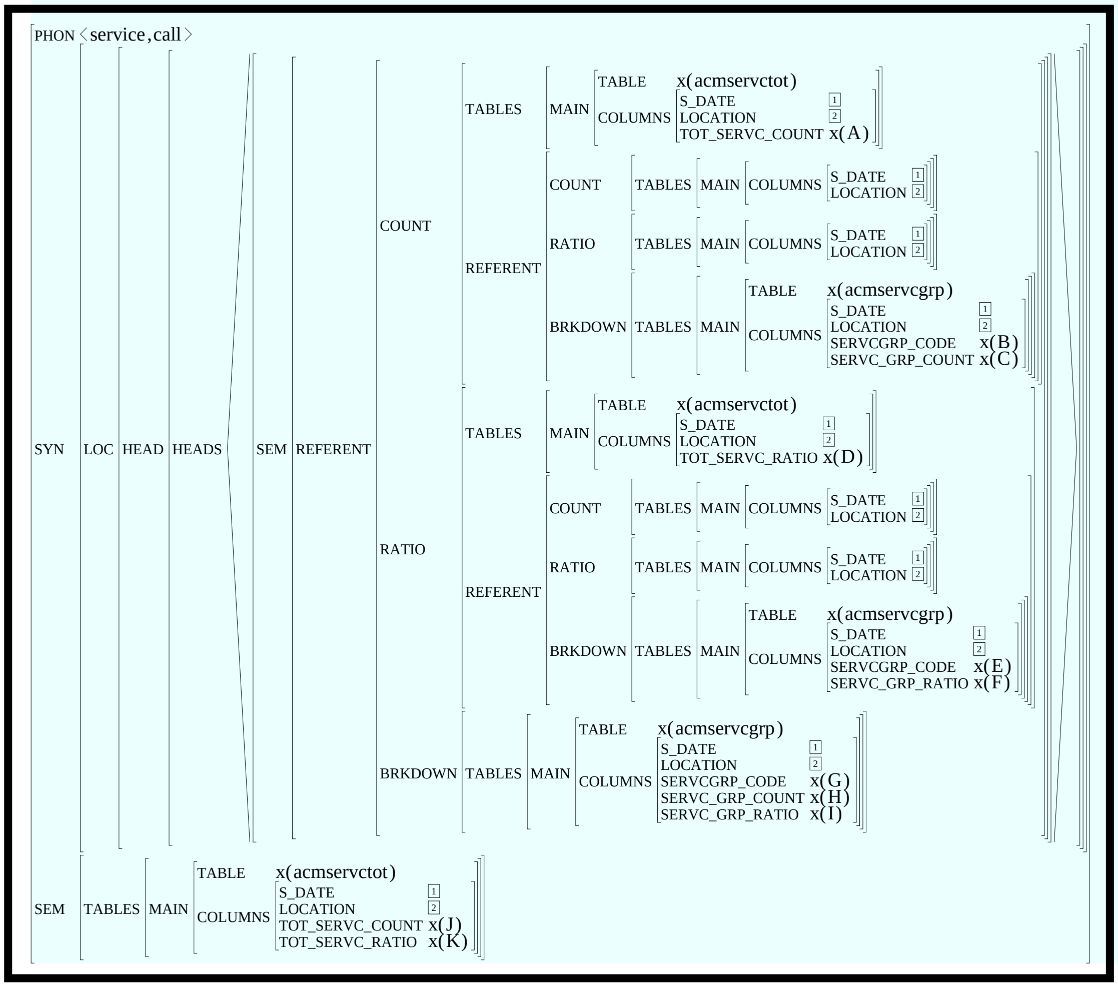


FIGURE 13. Analysis of *service call.*

ing on the attribute selected. For example, the COUNT|TABLES feature contains the TOT_SERVC_COUNT column while the RATIO|TABLES feature contains the TOT_-SERVC_RATIO column. The BRKDOWN|TABLES feature specifies a different table (and thus different columns) than in the other two cases. This reflects the three different meanings associated with the phrases *service call count*, *service call total*, and *service call breakdowns*. Notice also that the meaning of the phrase *service call* by itself is distinct from any of these cases. The REFERENT feature with the HEADS attribute contains additional referent information which would be relevant for subsequent modifiers, thus allowing analyses for phrases like *service call count breakdown.* Due to the way in which our lexical entries are structured, there is some unnecessary information that is carried along in the values of features like REFERENT|COUNT|REFERENT|COUNT. This information is associated with the `dateloc` lexical type introduced earlier in this section.

Notice in the sign contained in the HEADS list of *service call* that there is a selection of different possible semantics for phrases which in *service call* can appear as a modifier. Whatever *service call* modifies will have all this information instantiated within its own feature structure. So if *service call* modifies a noun like *count*, shown in Figure 14, then the value of *count*'s SEM feature will contain all the REFERENT information provided. The lexical entry for *count* is constructed so that it merely looks into the REFERENT fea-

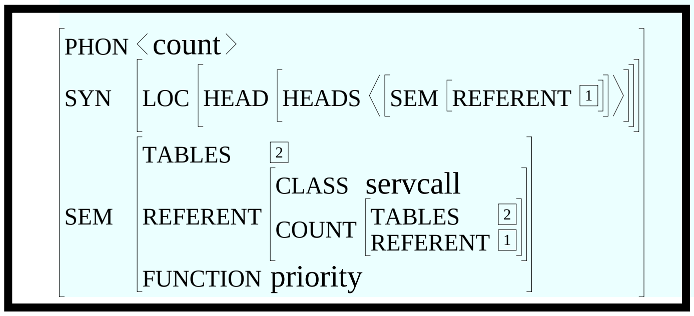


FIGURE 14. Analysis of *count*.

ture and equates its own meaning with that contained in the COUNT feature of its REFERENT. During a derivation, this means that *count* uses as its own meaning what ever meaning is supplied to it by the phrase that is modifying it. So on its own, *count* does not have a meaning (in our domain), it only has a meaning in the context of some modifier. The final semantics of the phrase *service call count* is provided in Figure 15.

Notice how our approach takes care of the compositionality problems discussed in Section 2.2.3. We take care of them by having the semantics of the modifier contain a selection of different meanings. When a modifier combines with a head, the head then simply selects the information supplied to it, and uses it as its own semantics. With the head declaring itself to have **priority**, it means that its own semantics will be used in determining the semantics of the resulting compound.

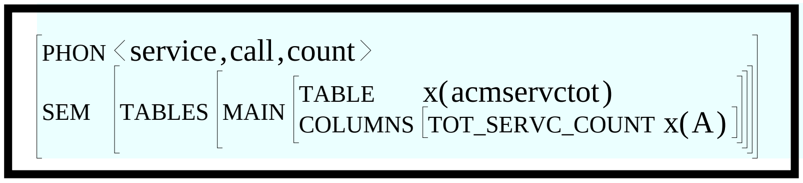

PHON ⟨service, call, count⟩
SEM TABLES MAIN TABLE x(acmservctot)
COLUMNS TOT_SERVC_COUNT x(A)

FIGURE 15. Analysis of *service call count.*

3.2.4 Summary

We have spent a fair amount of space discussing how semantic information from the lexicon can be used during a derivation to obtain the semantics of a complex constituent particularly in those cases where a purely compositional approach to semantics would be troublesome. Our approach assumes a unification based treatment of semantics in which information is encoded in feature structures. Although the simple unification of the feature structures works in some cases, we need the additional flexibility provided by the **default** and **priority** features which gives us a selective form of unification. We have assumed that compounding is treated syntactically as adjunction, and the adjunct can examine or specify all the information associated with its head. Such an assumption is also consistent with categorial grammar approaches and thus the same techniques that we use might be used with formalisms such as Unification Categorial Grammar [4]. We have also assumed that the head can access specific semantic information associated with its adjunct. In some formalisms, it is easy to describe these bidirectional (or even circular) relationships. But in the original formulation of HPSG, this was not allowed. Thus we developed some information passing techniques (using REFERENT information) that permitted the head to play a role in determining the resulting semantics. These information passing techniques are not specific to HPSG, and could be adapted for other formalisms as well.

3.3 Semantic Representations to SQL via Logical Form

HPSG semantic representations are translated first into an internal logical form before being transformed into SQL. The SystemX logical form is a set domain relational calculus. Its inspiration comes from the logical form used in the TQA natural language interface [12], to which it retains many similarities. Since it is a relational calculus it can be translated into any relational query language, thus enabling SystemX to be transported to a wide range of relational database management systems.

During the process of translating a semantic representation of a query into logical form and then into SQL, information is added which enhances the report ultimately displayed. SystemX has a number of strategies for improving the presentation of the tabular reports it produces. These include reducing noise in the data by making assumptions about the data users want to see, by ordering the rows and columns in such a way that the reports are easy to understand, and by using English expressions rather than internal database names and codes whenever possible. The information used by the system to implement these strategies is acquired during the customizing process either from the database itself or from someone who is familiar with the application.

Figure 16 shows a semantic representation for the sentence *Give me the unscheduled outages.*

```
((TABLES
  ((MAIN
    ((COLUMNS
      ((OUTAGES NIL NIL) (OUTAGE_CODE 20 NIL)
                   (LOCATION NIL NIL)
                   (S_DATE NIL NIL))
      NIL)
     (TABLE ACMSREL NIL))
    NIL)
   (SUBTABLES NIL NIL))
  NIL)
 (FUNCTION NIL NIL)
 (REFERENT NIL NIL)
 (TYPES ((FTYPE NIL NIL) (DTYPE NIL NIL))
      NIL))
```

FIGURE 16. Semantic representation for *Give me the unscheduled outages.*

Figure 16 is a representation of the same sentence in logical form.The logical form has been enhanced with some defaults which are discussed in Section 3.3.1.

```
(((SETX X22
    (SETX X23
      (SETX X24
        (AND
          (RELATION ACMSREL
            (S_DATE LOCATION OUTAGE_CODE OUTAGES)
            (X24 X23 '20 X22)
            (= = = =))
          (MAX X24 ((SETX X25 (RELATION ACMSREL (S_DATE) (X25) (=)))))
          (IN* X23 ('2700 '2710 '2720 '2730 '2740 '2800 '6300)))))))
```

FIGURE 17. Logical form for *Give me the unscheduled outages* with defaults added.

Table 1 shows the output produced from the SQL statement of Figure 16.

Table 1: Output generated from SQL.

OUTAGES	LOCATION	S_DATE
40	6300	31-AUG-92
14	2800	31-AUG-92
2	2740	31-AUG-92
4	2730	31-AUG-92
4	2720	31-AUG-92
7	2710	31-AUG-92
9	2700	31-AUG-92

Certain problems exist with the intelligibility of Table 1:

- The company divisions are referred to by their internal database codes, rather than by their English names.
- Column names are names of attributes of the database relation from which the data for the report was extracted. In this case the headings are fairly intuitive, but often attribute names are anything but intuitive to end users.
- The columns are ordered such that the column containing the statistics which measure the sets of outages appear to the left of the columns containing the attributes which specify those sets.
- The rows are not ordered appropriately. The row describing the Western Region as a whole (`location 6300`), appears above the rows describing the divisions within the region rather than below them where summary lines in statistical reports generally are found.

SystemX corrects these problems via enhancements made first to the logical form representation and then to the SQL. The SQL representation ultimately generated has the much more readable output shown in Table 2. In the rest of Section 3.3the enhancement

Table 2: The same output made more readable.

Date	Location	# of Outages
31-AUG-92	VANCOUVER	9
31-AUG-92	VICTORIA	7
31-AUG-92	FRASER	4
31-AUG-92	SURREY	4
31-AUG-92	ABBOTSFORD	2
31-AUG-92	CALGARY	14
31-AUG-92	WESTERN CANADA	40

process is described in detail.

3.3.1 Addition of Defaults

Default selection restrictions are added to queries during the translation of the semantic representation into logical form. For example, in the RCIOPS application, if a query does not mention time and location then specifications for the most recent possible time and an appropriate set of locations will be added. (For our target user, the Vice-President of Customer Service for the Western region, which is the default location set encompassing the divisions under his purview.) Defaults are added to the system when it is being customized to a particular application. Figure 16 shows the logical form for *Give me the unscheduled outages* with time and location defaults added.

3.3.2 Substitution of English Names for DB Identifiers

Another enhancement made to the logical form representation of a query causes the substitution of English *names* for internal database *identifiers* in the reports presented to the user. The information used in the substitution is extracted automatically from *reference relations*, i.e., database tables which relate attributes whose values are names, with attributes whose values are identifiers. For example, the relation `Rflocations` is a reference relation in the Rogers database: it relates the attribute `location`, whose values are codes identifying the divisions and regions within the company (e.g., 2700), with the attribute `location_desc`, whose values are the English names of those divisions and regions (e.g., Vancouver). In order to guarantee uniqueness of identifiers and permit efficient comparison of database values, codes such as the ones constituting the domain of the `location` attribute are commonly used within databases to identify objects in the application domain. However, users of the database may not know the denotation of these codes. The presence of reference relations in the database facilitates the use of English in reports in place of the codes. Figure 18 shows the logical form from Figure 16 after the substitution enhancement has been performed.

The substitution of names for identifiers subsequent to the process of the semantic interpretation is highly desirable for the parser and lexicon developers. This is because the

```
((SETX X22
    (SETX #:NONE-1853
      (SUBX X23
        (SETX X24
          (AND
            (IN* X23 ('2700 '2710 '2720 '2730 '2740 '2800 '6300)))
            (MAX X24 ((SETX X25 (RELATION ACMSREL (S_DATE) (X25)
(=)))))
            (RELATION ACMSREL
              (S_DATE LOCATION OUTAGE_CODE OUTAGES)
              (X24 X23 '20 X22)
              (= = = =))
            (RELATION RFLOCATIONS
              (LOCATION_DESC LOCATION) (#:NONE-1853 X23) (= =))))))))
```

FIGURE 18. Logical form following name substitution.

semantics of the parser is closely tied to the structure and contents of the database, so lexical entries contain tokens with the same names as the identifiers that denote domain entities within the database, e.g., (`Location` 2700) for *Vancouver* and (`Servcgrp_Code` 40) for *terminal calls*. Lexicon developers can thus focus on the relationship of the lexicon to the database without being concerned about the readability of reports.

SystemX acquires knowledge of reference relations via a simple dialogue with the application developer (see Figure 19).

```
Is ACMMAN a reference relation? (Y or N): n

Is RFLOCATIONS a reference relation? (Y or N): y

Which of the following attributes in relation RFLOCATIONS contains
names of entities denoted by values of attribute LOCATION:
LOCATION_DESC CONTAIN_CODE REF_NO?
(Type nil to quit) --> location_desc

Is RFOUTAGES a reference relation? (Y or N): y
```

FIGURE 19. Acquiring identifier/name pairings.

3.3.3 Addition of English Aliases

SystemX enhances the SQL it generates from logical form representations by substituting English *aliases* for database attribute names normally used as column headings in report tables. Database attribute names frequently are not easily understood by users of the database. Figure 20 shows the SQL generated from the logical form in Figure 18 with English aliases added (e.g., "Location" is the alias for `LOCATION_DESC`).

English aliases are acquired during application development via a dialogue with the developer, and stored along with other information about attributes (such as data type and length, etc.) automatically acquired by the system from the data dictionary.

```
   SELECT A.S_DATE “Date”, B.LOCATION_DESC “Location”, A.OUTAGES
“# of Outages”
FROM ACMSREL A, RFLOCATIONS B
WHERE A.OUTAGE_CODE = 20
 AND A.LOCATION = B.LOCATION
 AND A.S_DATE = (SELECT MAX(DISTINCT C.S_DATE) FROM ACMSREL C)
 AND A.LOCATION IN (2700, 2710, 2720, 2730, 2740, 2800, 6300)
ORDER BY A.S_DATE, B.REF_NO
```

FIGURE 20. SQL with English names added.

3.3.4 Ordering of Report Columns

During the process of translating logical form into SQL, SystemX arranges for the rows and columns of the report tables to be ordered in a perspicuous manner.

The basic strategy SystemX follows when ordering report columns is to place the column(s) which denote the type of entity described by the report at the left of the table. We call such columns, *subject columns*. For example, in a table reporting the grades assigned to students in a course, the column containing the students’ names (or numbers) is the subject column and will appear at the left. This strategy allows readers of the report to distinguish quickly the subject of a row from the data associated with that subject. SystemX uses information about primary key attributes acquired automatically from the data dictionary (or from the application developer if it is missing from the dictionary) to decide which reports columns are the subject columns. Essentially, if a report column corresponds to a primary key attribute, or if it corresponds to a name attribute in a reference relation, then it is deemed to be a subject column.

SystemX arranges the subject columns in report tables in order of generality when more than one column identifies the subject. This arrangement allows the user to scan the table from left to right to “pin down” a particular subject and its associated data. For example, consider the semesterly report on grades for an entire university department. The subject of the report is enrollments and is identified by a pair of columns, one containing course names and the other containing student numbers. Within the table there are fewer distinct course names than there are distinct student numbers. Thus, the course names will be placed in the first column and the student numbers in the second. Users can quickly find the particular grade they are interested in by locating the course name (from the relatively small number of distinct course names) at the extreme left, move to the right to find the student number from among the (relatively) small subset of the distinct student numbers associated with the course name, and finally locate the grade by continuing to the right. SystemX dynamically determines the ordering of the subject columns in tabular reports by determining (once per session) the number of distinct values in each associated attribute via a database query. Once the size of the domain is established for a particular attribute it is saved within the ADKB for future use during the current session.

When a particular attribute does not fit into this ordering strategy, SystemX allows developers of application systems to override the strategy by specifying a relative ordering weight for the attribute to be stored in the ADKB. For example, it is frequently the case in applications which contain historical data that the columns containing temporal values should be placed leftmost in a report, regardless of the length of the history stored in the database (i.e., regardless of the number of values contained in the corresponding attributes). The location of the `Date` column in Table 2 was determined by using this override facility.

3.3.5 Ordering of Report Rows

Unless the user has specifically mentioned a preferred ordering in her query, SystemX orders rows in tabular reports by the values in the subject columns in the same order as the columns appear in the report. In the case of a department's semesterly grades report whose subject columns are `course_name` and `student_number` in that order, this row ordering strategy will arrange the rows first into groups of courses (alphabetically ordered on course names), and order the rows within course groups numerically on student numbers.

SystemX provides the facility to specify that row ordering is to take place on attributes other than key attributes or name attributes. For example, the values of the attribute `ref_no` in the relation `Rflocations` impose a certain order onto the various divisions and regions in the company. As Figure 20 and Table 2 illustrate, SystemX automatically makes the appropriate substitution of the `ref_no` attribute for the `location_desc` attribute in the SQL *ORDER BY* clause which specifies the order of the rows in the report.

3.4 Graphical Data

SystemX is able to present data in graphic form. Although users find this type of presentation highly useful, specifying the parameters which define the desired graph tends to be a complex task. Underlying many executive information systems are historical statistical databases that summarize company operations over periods of time. Users of such databases often wish to see line graphs which show the trend in the numbers representing a particular facet of the business. Anomalies and other items of interest are usually readily apparent in these types of graphs. This is the format in which SystemX can represent this type of data.

A complete specification of a graphed trend includes the statistic of interest, the timing of the data to be represented and the business unit(s) to which the statistic applies, each of which will be represented by one line on the graph. In Figure 21 the statistic graphed is the monthly total of unscheduled outages, the time period is the 12 months prior to and including August, 1992, and the business units are the Eastern and Western Regions respectively.

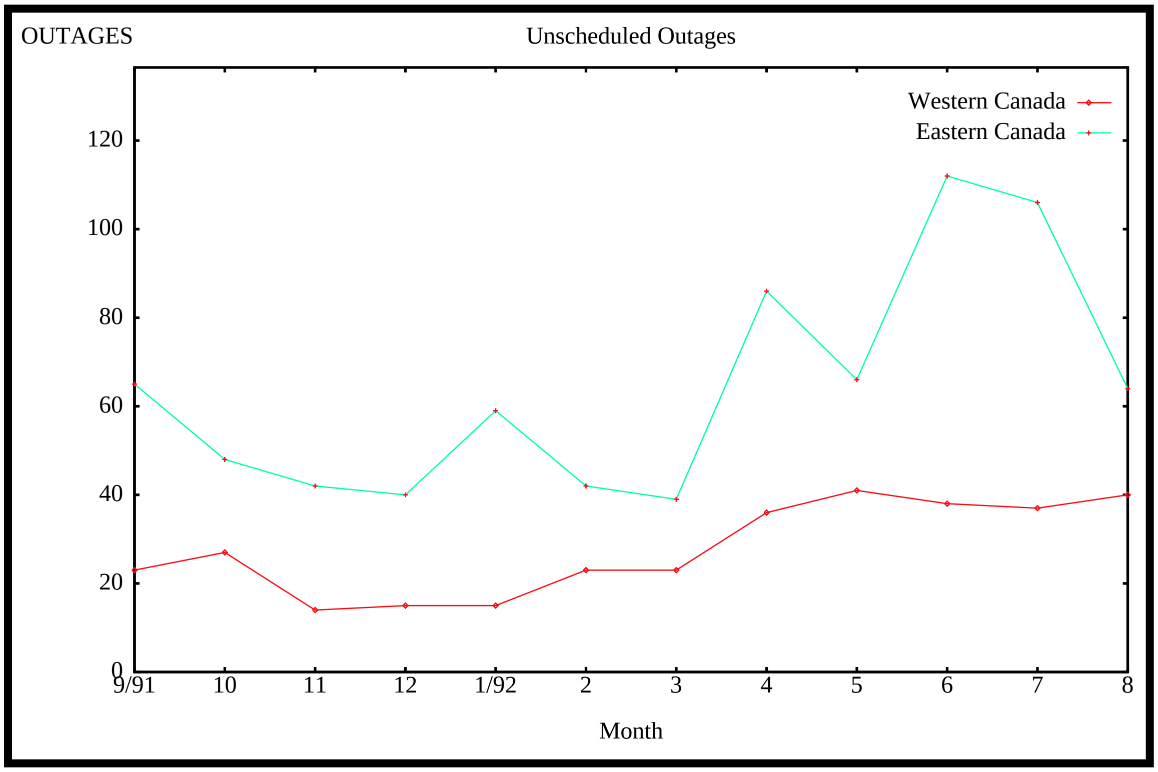


FIGURE 21. A trend graph.

4.0 System Use

The following example session shows what interaction with SystemX is like. User input is in italics; system responses are in the same font as the body of the paper. The first two queries are specified in natural language with tabular presentation of data. The third and fourth requests are for line-graph presentation of data. The third request is for a canned query (system reliability — another name for outages), whereas the subject of the fourth query is specified in English. The other parameter values required for trending are set with the help of a guided dialogue.

SX Main Menu

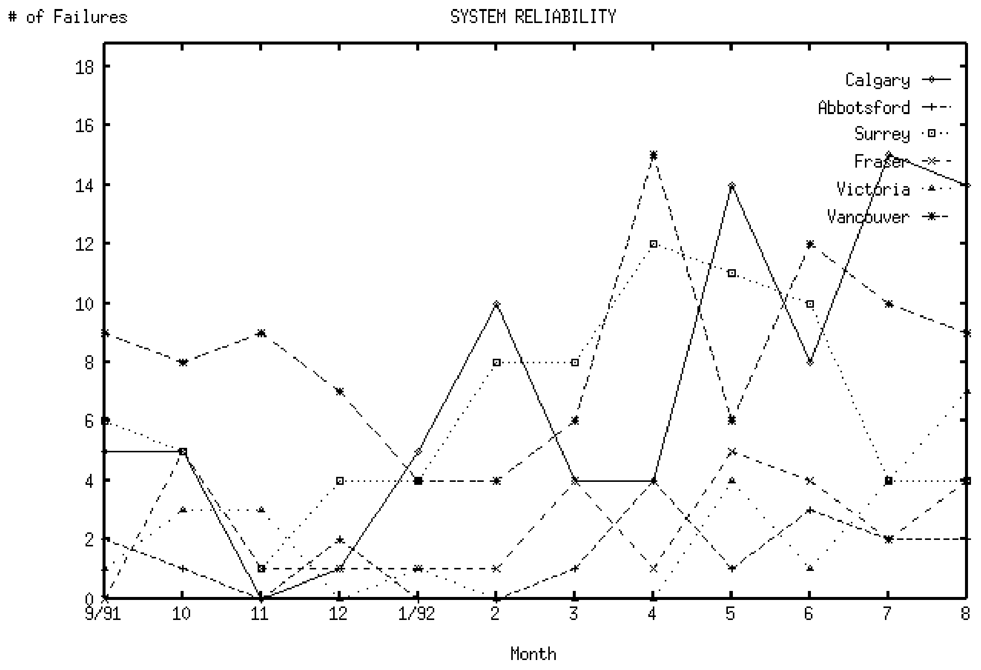
1 New Query
2 Display SQL
3 Save Last Response
4 Print Saved Responses
5 Display a Trend
6 Print Last Trend
7 Make a Comment
8 Stop
WHAT NEXT? > *1*

```
New Query

Enter sentence to be parsed:
```
give me the basic service problems for vancouver in august

```
Parsing Completed.

give me the basic service problems for vancouver in august

Consulting Oracle for your data.

Date           Call Type                        # Serv Calls   KSubs
--------------------------------------------------------------------
29-AUG-92   SUB OWNED EQP FAULTS        482            1.7
29-AUG-92   DROP PROBS. OUTSIDE         210             .7
29-AUG-92   SUB EDUCATION               141             .5
29-AUG-92   DISCONNECT IN ERROR         221             .8
29-AUG-92   NO FAULT FOUND              151             .5
29-AUG-92   SUB NOT HOME                223             .8
29-AUG-92   CANCELLED AT DOOR             3              0
29-AUG-92   REFERRED SYS PBLMS          377            1.3
29-AUG-92   ADDRESSABLES                 23             .1
29-AUG-92   DROP PROBLEMS INSIDE        320            1.1
29-AUG-92   OTHER TRUCK ROLLS            33             .1

11 row(s) selected

1  New Query
2  Display SQL
3  Save Last Response
4  Print Saved Responses
5  Display a Trend
6  Print Last Trend
7  Make a Comment
8  Stop
```
WHAT NEXT? > *calls received*

```
Parsing Completed.

calls received

Consulting Oracle for your data.

Date              Office         Location                  # Calls Recd
--------------------------------------------------------------------------
31-AUG-92         BUSINESS       VANCOUVER                   86518
```

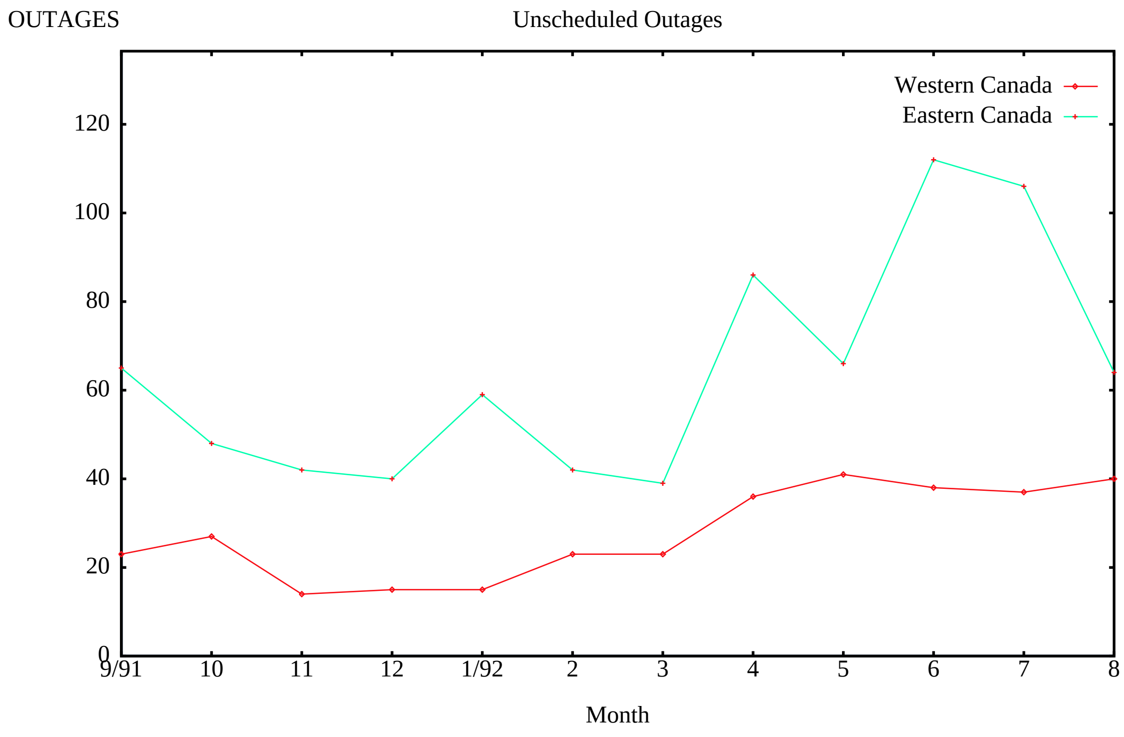
```
31-AUG-92      BUSINESS     VICTORIA             13165
31-AUG-92      BUSINESS     WESTERN CANADA      116105
31-AUG-92      REPAIR       VANCOUVER            42949
31-AUG-92      REPAIR       VICTORIA              3493
31-AUG-92      REPAIR       ABBOTSFORD            1157
31-AUG-92      REPAIR       CALGARY               5617
31-AUG-92      REPAIR       WESTERN CANADA       53216

9 row(s) selected

1  New Query
2  Display SQL
3  Save Last Response
4  Print Saved Responses
5  Display a Trend
6  Print Last Trend
7  Make a Comment
8  Stop
WHAT NEXT? > 5
Display a Trend

Trend Menu

1  # of C/S Representatives
2  Service Call Ratios
3  System Reliability
4  Subscribers per Employee
5  Subscribers per Km of Plant
6  Maintenance Calls
7  Repair Calls per TSR FTE
8  Repair Calls per TSR Hours
9  Specify Ad Hoc Trend
10  Cancel Trend Request
WHICH TREND? > 3
System Reliability

Do you wish the current, 12 month trend for the Western divisions? (Y or N):  y

Consulting Oracle for length of recorded history.

Consulting Oracle for range of valid y-axis values.

Consulting Oracle for data for VANCOUVER.

Consulting Oracle for data for VICTORIA.

Consulting Oracle for data for FRASER.
Consulting Oracle for data for SURREY.

Consulting Oracle for data for ABBOTSFORD.

Consulting Oracle for data for CALGARY.

Building the graph.

Hit return to continue
```



```
1  New Query
2  Display SQL
3  Save Last Response
4  Print Saved Responses
5  Display a Trend
6  Print Last Trend
7  Make a Comment
8  Stop
WHAT NEXT? > 5
Display a Trend

Trend Menu

1  # of C/S Representatives
2  Service Call Ratios
```

```
3  System Reliability
4  Subscribers per Employee
5  Subscribers per Km of Plant
6  Maintenance Calls
7  Repair Calls per TSR FTE
8  Repair Calls per TSR Hours
9  Specify Ad Hoc Trend
10  Cancel Trend Request
WHICH TREND? > 9
Specify an Ad Hoc Trend

Enter phrase describing data to be graphed:
(Return to cancel): the unscheduled outages

Parsing Completed.

Do you wish the current, 12 month trend for the Western divisions? (Y or N): n

Do you wish to specify a historical trend date? (Y or N):  n

Consulting Oracle for length of trend period.

Do you wish to specify the trend duration (default 12 mos.)? (Y or N):  n

Do you wish to specify the trend locations (default: Western divisions)? (Y or N):  y

Retrieving Location table from Oracle.
LOCATION     LOCATION_DESC
 ------------------------------------------
    6100          CANADA
   2100           EASTERN CANADA
   6300           WESTERN CANADA
   2300           TORONTO
   2310           GRAND RIVER
   2320           LONDON
   2330           PEEL
   2340           PINE RIDGE
    2380          CORNWALL
    2390          NEWMARKET
    2700          VANCOUVER
    2710          VICTORIA
    2720          FRASER
    2730          SURREY
    2740          ABBOTSFORD
    2800          CALGARY
    2360          HAM/NIA
    2400          OTTAWA
```

```
18 row(s) selected

Please enter all location codes together (e.g. 2700 2710 2720)  -> 6300 2100

Consulting Oracle for range of valid y-axis values.

Consulting Oracle for data for WESTERN CANADA.

Consulting Oracle for data for EASTERN CANADA.

Building the graph.

Hit return to continue
```



5.0 The Evaluation

Given our view of design as an evolutionary process (see introduction to Section 2.0), testing and evaluating SystemX was important. Although SystemX was used and tested in the laboratory by its developers, this did not give us information about how the system would perform with real users in their own work environment. Our interaction with Rogers Cablesystems provided a test bed for gathering some real-world data on SystemX's usability.

User interface evaluation is currently an active area of research (see [1], [11], [14], and [22], for example). The methods studied — heuristic evaluation, usability testing, guidelines, cognitive walkthroughs — most often occur at the sites in which the systems are developed. When people are brought in to use a system, they are often given tasks to complete rather than allowed free choice. These methods are valuable, but we wanted to look at usability of SystemX in terms of the way people go about getting their own tasks done, not ours.

5.1 Methods

We prepared a field test which employed both formative and summative evaluation methods. The goal of formative evaluation is to provide information about how to refine and improve the design of a system. Summative evaluation provides information to assess impact, usability, and effectiveness of the system. Therefore qualitative (formative) and quantitative (summative) measures were required. Targeted areas of study were retention, task completion, dialogue failure (due to errors, cancelled requests, queries that could not be processed, and inappropriate responses), and user preferences and satisfaction. (Task is used here to mean a process by which the user's goal or purpose in interacting with the system is achieved.) We were also interested in user strategies for solving problems and accomplishing goals.

Minimally intrusive evaluation methods were employed. These included logging, videotaping, on-line commenting, and questionnaire/interviewing. The user's interactions with the system were automatically logged in a file and time-stamped. An on-line commenting facility was provided to accept and store typed, time-stamped comments. The user was videotaped using the system twice during the field test, which lasted a month. A follow-up interview was conducted, guided by a questionnaire.

The videotaped training session took place in our research lab and consisted of a five-minute demonstration followed by a ninety-minute hands-on session, during which the user talked freely about what he was doing and asked questions of the trainers. A few days after the training session, the system was set up in his office and a 46-page user manual provided. After ten days the user was again videotaped interacting with SystemX. He was interviewed by two researchers at the conclusion of the test.

The videotaping was conducted using a two-camera set up. One camera was set to capture the face of the user and the other the screen. A video tuner was used to inset the output from one camera onto the output of the other camera. The final product was a single tape of the interaction at the screen with the user's face inset in the upper right-hand corner. This organization facilitates evaluation of interaction, especially at the level of nonverbal cues from the user.

5.2 Results

We were interested in analysis of the data in terms of system performance and usability. Under what circumstances does dialogue failure occur? Does the interface support work at a task-oriented level? What linguistic elements currently outside the coverage of the system are necessary for a usable interface? What extensions to the system would users like to see? Although many user preferences may not be implementable at this time, we are very interested in determining what those preferences are as they could help in charting future research directions.

VANNA, a tool developed at the University of Toronto by Beverly Harrison and Ron Baecker ([10]), was used to facilitate the laborious task of video tape annotation and analysis. The system allows evaluators to “code” (something like bookmarking and notemaking) a video tape while the tape is playing and also allows “real time” coding while the video is being shot.

Evaluation was done at three levels: (1) patterns of use, (2) feasibility of user's “wish list” for extensions to the system, and (3) implications of patterns of use on extension of linguistic coverage and interface refinement and improvement. Patterns of use and the user’s “wish list” are reported in Section 5.2.1and Section 5.2.2 respectively. Implications of the results in terms of modifications and extensions to SystemX are presented in Section 5.3. At the level of patterns of use, we examined counts of menu selections, interactions not resulting in data, natural language use, patterns in queries, and task modes.

5.2.1 Patterns of Use

Menu Selection Counts. Table 3 shows the number of times each item in the main menu was selected.

Table 3: Count of main menu selections.

Menu Item	# of Selections
1 New Query	66
2 Display SQL	7
3 Save Last Response	1
4 Print Saved Responses	1
5 Display a Trend	229
6 Print Last Trend	102
7 Make a Comment	2
8 Stop	9
Total	417

Over half of the menu selections were for graph presentation of data (229 out of 417), a quarter for printouts of graphs (102 of 417), and a sixth for tabular data (66 of 417). Over three quarters of the requests for data were for line-graph presentation of data (229 of 295).

Table 4 shows the number of times each item in the trend menu was selected.“Service Call Ratios” (item 2) was the most frequently requested canned graph, but the most frequently selected item by far was “Specify Ad Hoc Trend” (item 9). Over 60% of the requests for graphs were ad hoc and partially specified in natural language.

Table 4: Count of trend menu selections.

Menu Item	# of Selections
1 # of C/S Representatives	2
2 Service Call Ratios	34
3 System Reliability	10
4 Subscribers per Employee	2
5 Subscribers per Km of Plant	12
6 Maintenance Calls	11
7 Repair Calls per TSR FTE	13
8 Repair Calls per TSR Hours	4
9 Specify Ad Hoc Trend	135
10 Cancel Trend Request	5
Total	228

Interactions not Resulting in Data. Table 3 shows the number of interactions not resulting in data. It should be noted that these counts include cancelled requests, failures, and stop requests but do not include failures in which the data presented is not the data requested. The grouping of interactions not resulting in data is convenient for doing usability analysis.

The first column of Table 3 contains the menu selections subcategorized for the type of behaviour associated with the non-presentation of data. For example, “New Query” requests which did not result in data were of four types: (1) those which could not be processed, (2) those resulting in Lisp errors, (3) those resulting in SPE errors, and (4) those cancelled with a Ctrl-C interrupt. There are multiple layers supporting SystemX, and errors were categorized according to the layer in which an error occurred. Thus Lisp errors are those occurring in the Lisp layer (SystemX is written in Lisp). SPE (Symbolic Programming Environment) errors are those occurring in the windowing environment layer.

Table 5: Count of interactions not resulting in data.

Behaviour	# of Occurrences
1 New Query	
– Couldn’t process	16
– Lisp error	2
– SPE error	1
– Ctrl-C	1

Table 5: Count of interactions not resulting in data.

Behaviour	# of Occurrences
2 Display SQL	
– No data	1
3 Display a Trend	
– Cancel from trend menu	5
– Cancel while specifying topic	20
– Lisp error	9
– ?	1
4 Print Last Trend	
– Lisp error	6
– Default printer name problem	12
5 Stop	
– Midsession	6
– End of session	3
Total	83

About 25% of the “New Query” requests could not be processed. However, slightly less than half of those which could not be processed were most likely not intended as natural language requests for data. For example, two were empty queries, two consisted of “p,” and one of “v1.” If we omit “unintentional” queries from the count, then about 14% of the requests could not be processed. It is interesting to examine the change over time in the ratio of queries that couldn’t be processed to “New Query” requests. The ratio for the first two sessions of the field test was 1:3, whereas the ratio for the remaining sessions was 1:7.5. This decline suggests that the user learned some of the boundaries of the linguistic coverage fairly quickly.

The SPE error occurred during a Ctrl-C interrupt attempt, but it is unclear from the log and videotape just what went wrong. The user decided he wanted a graph of the data instead of a table and tried to abort the query processing. Perhaps a fatal combination of keys was accidentally hit in the abort attempt. One complicated query was interrupted because the processing was taking too long.

The “Display SQL” request which did not result in data occurred because there had been no previous “New Query.” Perhaps this selection was accidental. It is clear from the videotape that some of the “Display SQL” requests were accidental. The user made the selection from the main menu which he actually wanted from the trend menu.

About 38% of the total interactions not leading to data were the result of cancelled requests and midsession stops. Analysis of these interactions will hopefully indicate some ways to enhance the interface so that the appropriate steps for completing a task are more obvious. However, some of the cancels may have occurred because the user changed his mind. It is also obvious from the videotape that some of the midsession stops were unintentional — the saying of “oops” was a good indication.

The “?” under “Display a Trend” refers to an interaction in which we cannot determine from the log what happened. Unfortunately this interaction occurred in a session which was not videotaped. The user had selected “Display a Trend,” then “Specify Ad Hoc Trend,” but the query was not entered.

The Lisp errors which occurred can be categorized into three types: (1) those associated with natural language input (“New Query” and “Specify Ad Hoc Trend” requests), (2) those associated with graphing, and (3) those associated with printing graphs. We thought we had trapped all Lisp errors, but that proved to be over-optimistic. It is interesting to note that an apostrophe appeared as the last or second to last character in the input of all natural language queries resulting in Lisp errors. The apostrophe is the key just to the left of the <Return>, so it is possible that this key was accidentally hit in reaching for the <Return> key. Keyboard design is an ergonomic problem, but the interface must deal with the consequences. The graphing and printing Lisp errors (and default printer name problems) have all since been analyzed, debugged, and fixed.

Natural Language Use. Sixty-eight percent of the requests for data (“New Query” and “Display a Trend” selections) were specified in natural language (counted as “New Query” from the main menu and “Specify Ad Hoc Trend” selections from the trend menu). We also examined the language used in forming queries with certain questions in mind. What words need to be added to the lexicon? What linguistic elements should be added to extend coverage?

In terms of comparisons of natural language mode to other modes, it is interesting to note that some of the canned trending queries were keyed in from “Specify Ad Hoc Trend” even though they were available directly on the trend menu. The user stated in an interview that he preferred keying to mousing because it is faster. He also stated that he liked natural language for the flexibility it offered, especially for doing analysis of the data. (He had previous experience with Rogers’ in-house executive information system in which queries were restricted to items on a menu.)

Query Patterns. We had originally viewed interaction as independent query-response pairs. This assumption was consistent with the way *we* used the system. However, our first “real” user was motivated by an interest in the data as opposed to an interest in testing the system, and used the system quite differently. The user likes to do what we have called *tweaking parameters*. A graph is requested for a certain statistic, and subsequent queries are for the same statistic but different parameter values. For example, a query for repair calls is followed by queries for repair calls in other locations or time periods.

Task Modes. In observing the way the system was used from a task-oriented point of view, two modes emerge: monitoring and analysis. In monitoring mode daily, weekly, monthly, yearly statistics are tracked. If anything interesting or unexpected is observed, a switch to analysis mode occurs. In analysis mode, queries are generated in an attempt to find explanations for, or test hypotheses about, the cause of the unexpected trend. The user said that this characterization was an accurate reflection of the way he works and used it as a context for talking about enhancements to the system that he would like to see.

Other Strategies. The user developed an interesting strategy for on-line help. When he wanted to review the names of subtypes of a general term, he asked for a tabular presentation of the general term. The system response would be a table of the breakdown of the general term into its subtypes. So, for example, the user requested a tabular presentation of “trouble calls” and was presented with a table of the 12 subtypes of trouble call (drop problems inside, drop problems outside, etc.). He then returned to trending and requested a graph of whatever subtype he was interested in.The user was aware that this information was in the User Manual, but he preferred “on-line help.” Needless to say, we plan to feature on-line help in the next version of SystemX.

The user also had an interesting strategy for debugging. The user made a few attempts to get data over a certain time period. The system responded with data, but the user said it was not the data he wanted. He then requested the SQL code to see how the system had interpreted his query. This was quite a sophisticated end user.

5.2.2 Users List of Extensions

A “wish list” of extensions the user would like to see was compiled in the post-test interview. The list includes (1) exceptions reporting, (2) avoidance of re-keying whenever possible, (3) smoothing of data, (4) extrapolation/forecasting, and (5) overlaying of graphs. The user said that he would like to see the monitoring mode automated. The monitoring subsystem would track data and report exceptions (data above or below certain thresholds). Avoidance of rekeying would be aided by ellipsis and a dialogue box for tweaking parameters. Smoothing of data refers to statistical manipulations to smooth out the “jagged edge” of line graphs. Our user would like to see the system make projections to aid in planning budget, staff requirements, etc. Overlaying of graphs refers to the ability to hold one trend constant and overlay other trends on the same graph. This feature would be useful for doing analysis, e.g., in determining whether the unscheduled outages trend follows the same pattern as the scheduled outages trend. Alternatively, one graph could be positioned above another on the screen. Both graphs would have the same scale for the x-axis, but their y-axes could be different.

5.3 Conclusions Drawn from Evaluation

We feel very fortunate to have found people in the “real world” who are interested in using SystemX. The valuable information we have gained as a result of the field test cannot be obtained in the laboratory. In terms of the methods used, we found the videotapes an important source of data to cross reference with the log files. There were several instances in which the intention of the user could not be determined from the logs, but the videotape provided an explanation. The videotapes were especially helpful in studying strategies because the user talked about what he was doing.

However, care must be taken in interpreting results. Only one executive was involved in the first field test. (Our proposal for two more field tests with a number of company executives has been accepted.) How much of the sublanguage is user-specific, company-specific, or industry-wide? How do other users work?

It should also be kept in mind that the goal of the field test was to obtain contextual information to be used in the refinement of SystemX, not to test hypotheses about usability of interfaces in general. So, for example, the user was asked to talk about what he was doing throughout the videotaped sessions. It could be argued that this changed his behaviour, i.e., he would have worked differently if he had not been encouraged to describe his intentions. However, we would have had to speculate a lot more about his intentions without the videotapes.

Our experience has demonstrated that combining natural language input with other input modalities can result in an effective interface for creating ad hoc graphs to display data dynamically retrieved from databases. The marriage of natural language with other input modalities forces an analysis of how a piece of a query is best specified: by natural language, menu selection or other means.

One metric for determining the appropriate modality is that portions of the query which refer to a small set of possible values are best handled by a similarly constrained input modality such as menu selection. A possible example is the specification of the date. A user interested in data selected over a coarse grained features such a month can select the relevant value for a 12 item menu. This is not to say that input of this piece of the query should be restricted to this modality. Phrases such as “in the last 2 months”, “between April and May” and “after August” show that even when the set of possible values is constrained, members of the set may be combined into structures whose semantic complexity taxes the menu selection modality.

The converse of this metric for determining the appropriate modality is that pieces of the query whose possible values range over a large set are best specified by natural language. Our experience with the customer service NLI indicates that the statistics which summarize performance can only be comfortably specified in an *ad hoc* query by natural language. The number of different statistics which may potentially be graphed can be large. Furthermore, the trend to amassing ever larger amounts of on-line data is widespread. Once the number of choices becomes significantly large, users of menu-based interfaces or GUIs are forced to navigate through a structure of options, which slows down the specification process and introduces complexity into the specification task. In these situations, a natural language input facility offers a realistic alternative. It allows users to say what it is they want using a short, relatively simple phrase rather than search for it in the structure. An NLI capable of interpreting such phrases is likely to be easier to construct and maintain. In addition, its response times will be short so that user frustration will be minimized. Even when the phrase input is ambiguous, it is likely to restrict the possible choices to a number which can be easily displayed so that the user can quickly indicate the desired one.

A second metric is that structurally complex pieces of a query should have natural language support. The examples given previously — e.g., “after August” — can be supported by menu selection only by forcing the user to learn a syntax. The English version is more natural.

The next level of evaluation concerns the implications of the patterns of use for refinement of the system. How can the system be improved at the linguistic and interface levels in light of what we have learned about the way the system is used in the real world? Some patterns point to extensions to linguistic coverage, some to interface solutions, while others are perhaps best handled at both levels.

Linguistic Coverage. At the linguistic level, natural language requests resulting in *couldn't process* messages were examined for words to add the lexicon. Other extensions to the lexicon include abbreviations and user-defined phrases for forming subsets of locations such as *very large systems, large systems, medium systems,* and *small systems*. The extension of coverage to include limited forms of conjunction has been implemented to allow queries for comparisons such as *repair calls for Vancouver and Toronto*. The extension to ellipsis supports interaction as discourse as opposed to a pattern of independent query-response pairs.

Interface Level. Parameter tweaking could be supported at the linguistic level using ellipsis, and at the interface level by providing a dialogue box in which default values for parameters were preset. The user would then have the option to reset any or all of the parameters without engaging in a lengthy dialogue with the system. In task-mode analysis, the same statistics are examined regularly in monitoring mode, suggesting that these queries might best be canned for quick access. Analysis mode is supported by ad hoc querying, dynamic statistical manipulations, graph-overlaying, and the ability to “apply a trend.”

The user said that he used the graph presentation of data almost exclusively, so we are looking at ways to refine that type of interaction. There are many specific recommendations for interface refinements based on our observations and on the user's suggestions. For example, the default size for graphs could be bigger as the user resized most of the graphs. He would also like to be able to loop within trending, i.e., get several graphs in sequence without having to return to the main menu.

We are investigating the use of “general terms,” i.e., terms that refer to multiple tables, multiple statistical attributes, multiple category attributes, multiple category attribute values, or even “multiple queries” as in the case of coordination. For example, “trouble calls” has twelve subtypes (twelve values for the category attribute `servcall_code`) and two statistical attributes: ratios and counts. What does the user want when general terms are used? How should the data be presented within a table or graph? How can the interface be designed so that system responses are more cooperative in these cases?

Modifications to SystemX. Several modifications were made to SystemX based on analysis of field test results and interviews with the user. The user now loops within trending until “Return to Main Menu” is selected from the trend menu. The trend menu item order has changed. “Specify an Ad Hoc Trend” is now first. “Print Last Trend” has been moved from the main menu to the second item in the trend menu so that the user does not have to return to the main menu to print a graph. The canned trends follow and have been reordered according to user preferences. The user can enter natural language specification of a trend by default without first selecting “Specify an Ad Hoc Trend” from the trend menu (as with “New Query” in the main menu).

We have improved the messages presented when disambiguation is required. Questions for parameter values in trending are now consistently of the form, “Do you want the default?” The title appearing on ad hoc graphs is now the natural language used to specify the graph.

Several problems uncovered in the field test have been corrected such as a problem with the graphic display of Abbotsford data, a problem with getting data about outage minutes through trending, and some minor graphing and printing problems. Other errors now caught include requesting for a one-month trend (which makes no sense for data compiled monthly), hitting <Return> without specifying location(s), and entering an unknown printer name when seeking to print a trend.

Linguistic coverage has been extended in a few ways. Additions have been made to the lexicon. An abbreviations recognition module has been created to support the use of certain abbreviations. Limited forms of ellipsis and coordination are now supported.

6.0 Discussion

The techniques for obtaining user feedback and system evaluation — interviews, user manual, field trial and evaluation — have been little described in the NLI literature and have caused us to evolve SystemX in ways we likely wouldn't have done ourselves.

Our management of natural language versus other input modes provides users with flexibility. Users can make whole queries with the system's restricted sublanguage if they so wish. In other queries, users can refer to database concepts in the English they use with colleagues, while menus take care of notions which may have restricted content such as time and location.

The use of statistical databases is common in the corporate world but remains relatively rare among NLIs. HPSG has not been used in many NLP systems. Our context-selection and priority mechanisms and syntactico-semantic categories (FTYPE and DTYPE) for treating complex nominals are well suited to HPSG but may well be transferrable to other language formalisms.

With a rapid increase in the number and size of databases, there is a pressing need for systems that allow easy, direct access to and manipulation of information. NLIs are such systems. The need for such systems clearly extends to people from many walks of life. SystemX is an advanced prototype NLI that provides intuitive and rapid access to data and relatively flexible presentation of output. New features are currently being added to the system which will extend these capabilities even further. SystemX is evolving into a sophisticated distributed information system that combines data access with data visualization, knowledge discovery, data analysis, etc. and is capable of understanding natural language input and producing natural language output.

7.0 Bibliography

[1]Booth, P. (1989). *An Introduction to Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

[2]Bouma, Gosse, and Gertjan van Noord (1993). Head-Driven Parsing for Lexicalist Grammars: Experimental Results. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL-93)*, Utrecht, The Netherlands, pp. 71-80.

[3]Calder, Jo, Ewan Klein, and Henk Zeevat (1988) Unification Categorial Grammar: A Concise Extendable Grammar for Natural Language Processing. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, pp83-86.

[4]Capindale, Ruth A., and Robert G. Crawford (1990). Using a Natural Language Interface with Casual Users. *International Journal of Man-Machine Studies,* 32, (3), pp. 341-361.

[5]Cercone, Nicholas J., and Gordon McCalla (1986). Accessing Knowledge through Natural Language. In Marshall C. Yovits (Ed.) *Advances in Computers, Volume 25.* New York, NY: Academic Press, pp. 1-99.

[6]Cercone, Nicholas J., Paul McFetridge, Gary Hall, and Chris Groeneboer (1989). An

Unnatural Natural Language Interface. In *Proceedings of the Combined International Conference of the 16th Conference of the ALLC & 9th Conference of the ACH*, Toronto, ON.

[7] Cercone, Nicholas J., Jiawei Han, Paul McFetridge, Fred Popowich, Dan C. Fass, Chris Groeneboer, Gary Hall, and Yue Huang (to appear). SystemX and DBLEARN: How to Get More from your Relational Database, Easily. *Integrated Computer-Aided Engineering*.

[8]Dasgupta, S. (1989). Structure of Design Processes. In Marshall C. Yovits (Ed.) *Advances in Computers, Volume 28.* San Diego, CA: Academic Press

[9] Hall, Gary, WoShun Luk, and Nicholas J. Cercone (1987). Disambiguating Queries Using Dependency Graphs. Technical report LCCR TR 87-07, Simon Fraser University, Burnaby, BC, Canada.

[10] Harrison, Beverley, and Ron Baecker (1992). Designing Video Annotation and Analysis Systems. In *Proceedings Graphics Interface '92* (Vancouver, BC, May 11-15). Toronto, ON: CIPS, pp. 157-166.

[11]Jeffries, R., Miller, J., Wharton, C., and K. Uyeda (1991). User Interface Evaluation in the Real World: A Comparison of Four Techniques. In *Proceedings CHI '91* (New Orleans, LA, April 28 - May 2). New York, NY: ACM, pp. 119-124.

[12]Johnson, David E. (1984). Design of a Robust, Portable Natural Language Interface Grammar. IBM Research Report RC10867, IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

[13]Kaplan, S. Jerrold (1984). Designing a Portable Natural Language Database Query System. *ACM Transactions on Database Systems*, 9, (1), pp. 1-19.

[14] Karat, C-M., Campbell, R., and T. Fiegel (1992). Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation. In *Proceedings CHI '92* (Monterey, CA, May 3-7). New York, NY: ACM, pp. 397-404.

[15] Kodric, Sandi, Fred Popowich, and Carl Vogel (1992). The HPSG-PL system: Version 1.2. Technical report CSS-IS TR 92-05, Centre for Systems Science, Simon Fraser University, Burnaby, BC, Canada.

[16]McFetridge, Paul (1991). Processing English Database Queries with Head Driven Phrase Structure Grammar. In *Proceedings of the 2nd Japan-Australia Joint Symposium on Natural Language Processing*, Iizuka City, Japan.

[17] McFetridge, Paul, and Nicholas J. Cercone (1990). The Evolution of a Natural Language Interface: Replacing a Parser. In *Proceedings of Computational Intelligence `90*, Milan, Italy.

[18] McFetridge, Paul, Nicholas J. Cercone, WoShun Luk, and Gary Hall (1988a). System X: A Portable Natural Language Interface. In *Proceedings of the 7th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI-7)*, Edmonton, AB, Canada, pp. 30-38.

[19]McFetridge, Paul, Gary Hall, Nicholas J. Cercone, and WoShun Luk (1988b). Knowl-

edge Acquisition in System X: A Natural Language Interface to Relational Databases. In *Proceedings of International Computer Science Conference '88,* The Computer Society of the IEEE (Hong Kong Chapter), Hong Kong.

[20] McTear, Michael (1987). *The Articulate Computer*. Oxford, England: Basil Blackwell.

[21]Martin, Paul, Doug Appelt, Barbara Grosz, and Fernando Pereira (1985). TEAM: An Experimental Transportable Natural Language Interface. *Database Engineering,* 8, (3), pp. 10-22.

[22] Nielsen, J. (1992). Finding Usability Problems through Heuristic Evaluation. *Proceedings CHI '92* (Monterey, CA, May 3-7). New York, NY: ACM, pp. 373-380.

[23] Pollard, Carl, and Ivan Sag (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. Centre for the Study of Language and Information, Stanford University, CA.

[24] Pollard, Carl, and Ivan Sag (forthcoming). *Head-Driven Phrase Structure Grammar*. Centre for the Study of Language and Information, Stanford University, CA.

[25]Popowich, Fred, and Susan Kindersley (1991). Developing Lexical Unification-Based Grammars. *Actus du colloqueILN'91 Informatique & Langue Naturelle,* LIANA, Université de Nantes, France.

[26] Popowich, Fred, Paul McFetridge, Dan Fass, and Gary Hall. (1992). Processing Complex Noun Phrases in a Natural Language Interface to a Statistical Database. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92),* Nantes, France, pp. 47-52.

[27]Popowich, Fred, and Carl Vogel (1991). A Logic-Based Implementation of Head-Driven Phrase Structure Grammar. In *Proceedings of the Third International Workshop on Natural Language Understanding and Logic Programming,* Swedish Institute of Computer Science, Kista, Sweden, pp. 239-255.

[28] Russell, Graham, John Carroll, and Susan Warwick-Armstrong(1991). Multiple Default Inheritance in a Unification-Based Lexicon. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics,* University of California, Berkeley, CA, pp. 215–221.

[29] Shieber, Stuart (1986). *An Introduction to Unification-Based Approaches to Grammar*. Chicago, IL: The University of Chicago Press.

[30]Vogel, Carl, and Fred Popowich (1990). Head-Driven Phrase Structure Grammar as an Inheritance Hierarchy. In W alter Daelemans and Gerald Gazdar (Eds.) *Inheritance in Natural Language Processing Workshop Proceedings,* Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland, pp. 104-113.

[31] Vogel, Carl, Fred Popowich, and Nicholas J. Cercone (1990). Inheritance Reasoning for Head-Driven Phrase Structure Grammar. Technical report CSS/LCCR TR 91-07, Centre for Systems Science, Simon Fraser University, Burnaby, BC, Canada.

[32]Woods, William A., Ronald M. Kaplan, and Bonnie Nash-Webber (1972). The Lunar Sciences Natural Language System: Final Report. Report No. 2378, Cambridge,

MA: Bolt, Beranek and Newman Inc.

[33]Zoeppritz, Magdalena (1983). Human Factors of a “Natural Language” Enduser System. In A. Blaser and M. Zoeppritz (Eds.) *Enduser Systems and Their Human Factors, Proceedings of the Scientific Symposium conducted on the occasion of the 15th Anniversary of the Science Center Heidelberg of IBM Germany, Heidelberg, March 18 1983*. Berlin, Germany: Springer-Verlag, pp. 62-93.

Appendix A Original Rogers Corpus (90 sentences)

RCIOPS customer service domain (62 sentences)

• Manpower subdomain (1 sentence)

"I'd like to see the graph that shows me the number of customer service employees by division."

• Subscribers subdomain (3 sentences)

"I'd like to see the trend that shows me the number of subscribers per employee for each division."
"Show me the trend of the total base of subscribers for Fraser."
"I'd like to see the trend that shows me the number of customers per km of plant."

• Outages subdomain (18 sentences)

"Give me a list of all of the outages that affected more than 5000 customers and give me that list by division."
"For Vancouver, I want to see the items that affected more than 5000 customers, the time that the outage occurred, duration of the outage, and reason for the outage."
"Give me a list of the number of outages that affected between 1 and 5000 customers by division."
"Give me a list of the outages that affect less than 1000 customers."
"Show me a summary of the number of outages we had by reason."
"Give me the summary that summarizes the number of outages we had by reason."
"Show me the trend for both scheduled outages and unscheduled outages."
"Give me the system reliability performance."
"How many major outages have we had in the last week."
"Give me a list showing by location all of the outages, both scheduled and unscheduled, that have affected more than 5000 subscribers."
"Compare system failures by minutes over the last three years for Vancouver."
"Give me the Western Region Outage Log summary."
"Show me the detail for those."
"How many of the same type of problem have we had in the last year?"
"How many of these have we had in the last year?"
"Give me the detail of those 18 outages."
"Show me the detail for the month of June for the less than 400 customers affected outages."
"Give me a list of outages where it took longer than X hours to rectify the problem."

• Service calls subdomain (16 sentences)

"Give me the service call ratio."
"Show me a list in descending order of drop problems by division."
"Show me the 12 month trend for drop problems by division."
"Compare the basic service problem statistics."
"Compare the basic service problems per 1000 customers."
"Compare the terminal equipment problems."
"Compare the service call activity (by division) (per 1000 customers)."
"Compare the service call activity by location in ascending/descending order (by size

of division).”

“Compare the trouble call activity.”

“Give me the rolling year's summary for the Vancouver division.”

“Over the last year for Vancouver what has been the relative order of problems over the last 12 months.”

“For Vancouver give me the major trouble call activity for each of the last 12 months.”

“For all divisions, show me the cancelled at door.”

“Show me by division the subscriber not home activity.”

“Show me the activity for subscriber not home.”

“For Vancouver show me the 12 month rolling history for subscriber not home.”

- **Telephone subdomain (24 sentences)**

“Give me the repair calls per TSR graph.”

“Give me the TSR costs per call graph (for the period ending October 1991).”

“I would like to see the information for Victoria for the month of October.”

“I would like to see the same thing for Calgary in June.”

“I would like to see costs and call volumes.”

“Give me the graph which compares calls received budget, calls received actual and calls answered actual.”

“I'd like to see the % calls answered by division for the last 12 months.”

“I'd like to see the average service level by division.”

“Give me the telephone performance.”

“Give me a list of all the locations where calls abandoned is greater than 5% (for the month of June).”

“Give me a list of all the locations where all trunks busy is greater than 2 hours (for the month of June).”

“Give me a list of all the locations where average wait time is greater than X seconds (for the month of June).”

“Give me a list of all the locations where average talk time is greater than X seconds (for the month of June).”

“For each location compare the number of calls received versus the number of calls budgeted.”

“What was the highest call type for Vancouver for June.”

“For June for Vancouver I want to see the weekly call volumes.”

“Show me day by day the call volumes per hour for June 1-7.”

“Show me day by day for the week of June 1 calls offered, calls taken and% abandoned.”

“On June 1 8-9 AM show me the number of calls by type.”

“What % of phone calls were answered within 10 seconds.”

“Give me the % of the population who called last month.”

“Give me the Vancouver CSR performance for September.”

“How many full-time CSRs do we have.”

“How many phone calls were answered by one CSR in one hour?”

“Compare calls received against budget for the month of July.”

Other RCIOPS subdomains (28 sentences)

• Accounts receivable subdomain (18 sentences)

"How much has my allowance calculation gone up or down this week?"

"How many people paid on the due date (for our last billing (run))?"

"How many people paid fifteen days later?"

"How many people did we send out a reminder notice?"

"And how much of the reminder notice did they pay?"

"What % of quarterly payers pay when due within 15 days?"

"How many wait till 30 days?"

"How many wait till 60 days?"

"How many people paid through the mail and through the bank (for our last billing)?"

"How many people paid the full bill, how many people paid a partial bill, and how many paid more than their bill?"

"How many people paid the full bill, how many paid partial, and how many paid more?"

"For billing week ending Oct 28/90 how many subscribers paid before due date, how many subscribers paid on due date, how many paid after 30 days or after 60 days?"

"After the second month from the BPX3 campaign, how many subscribers disconnected after 30 days, 60 days, 3 months, 6 months, 8 months?"

"Of our hot reconnects, how many paid within 15 days, 30 days and after?"

"From Sept 1 to Oct 31st how many new connections were completed and payment was made within 10 days, 30 days, 60 days?"

"From Sept 1 to Oct 31st how many cold reconnections were completed and payment was made within 10 days, 30 days, 60 days?"

"Of our contract group how many of each group are monthly, bi-monthly, quarterly, semi-annual, annual, and PAC?"

"From that contract group and pay method, what is their paying habits?"

• Work order subdomain (8 sentences)

"Show me for installation time-frames the divisions that don't meet the standard."

"Show me the same information but quantified so as to show how they fail to meet the standard."

"Give me the summary by location for the month of June of the types and numbers of work orders."

"What is our average cost of reconnection work order for the year to date."

"What is our average cost of reconnection work order in June."

"For Vancouver for the last year, give me the total number of reconnection orders and the average cost per order."

"Compare the number of reconnect orders by status for all divisions."

"For all divisions compare the number of reconnect orders by status."

• Pay method subdomain (2 sentences)

"Show me the pay method for Vancouver for this year."

"For the month of June show me the pay types by division."