DOD GATEWAY INFORMATION SYSTEM (DGIS) 
COMMON COMMAND LANGUAGE: 
THE FIRST PROTOTYPING AND THE DECISION FOR 
ARTIFICIAL INTELLIGENCE

Allan D. Kuhn 
with 
Randy L. Bixby 
Duc Tien Tran (Consultant)

August 1987
**DoD Gateway Information System (DGIS) Common Command Language:**

**The First Prototyping & The Decision for Artificial Intelligence**

**Allan D. Kuhn, Randy L. Bixby, and Duc Tien Tran**

**Abstract:**

The first prototyping experiences in DGIS Common Command Language (CCL) are related. DGIS began its initial prototypes in C language with DIALOG, BRS, NASA/RECON, and DROLS. These prototypes in a third-generation algorithmic language brought to surface a number of problems and questions in dealing with the distinctions of information systems. The issues concern both the user interface and the development design. Experiences, results, and conclusions in working with these systems are brought out. The decision to convert to and continue CCL development with Artificial Intelligence tools is explained. Our effort is a merging of PROLOG and C capabilities, to provide the DGIS user an AI-based searcher assistant interface that makes the human-machine interaction more human-like on DGIS.
DOD GATEWAY INFORMATION SYSTEM (DGIS) COMMON COMMAND LANGUAGE: THE FIRST PROTOTYPING & THE DECISION FOR ARTIFICIAL INTELLIGENCE

Allan D. Kuhn
with
Randy L. Bixby
Duc Tien Tran (Consultant)

August 1987

DTIC AI Foundational Series No. 4
DTIC CCL Report No. 1

Defense Technical Information Center
Office of Information Systems and Technology
Alexandria, VA 22304-6145


No. 3: Toward An Artificial Intelligence Environment for DTIC: Proposed Tasks; Recommended Configurations; Projected Start-up Costs. Defense Technical Information Center, May 87, AD-A181 103.


No. 5: [Pending]
EXECUTIVE SUMMARY

DTIC's Common Command Language (CCL) project for the DGIS precipitated DTIC's entry into the Artificial Intelligence environment. This first report on the CCL activity relates our first prototyping experiences in CCL development. DTIC began its initial prototypes in C language with DIALOG, BRS, NASA/RECON, and DROLS. These prototypes, developed in a third-generation algorithmic language, brought to the surface a number of issues relevant to information systems and the Common Command Language concept.

These issues concern both the user interface and the development design. Experiences, results, and conclusions gained from working with these systems are explored. We have learned that creating a standard command language is only a minor part of the challenge we face. The major challenge involves accommodating the operating characteristics of the individual systems.

The creation of a CCL is only one component of the "CCL-need" issue. A second component is creation of a CCL System that allows a user to search in unfamiliar systems without needing to know a system's operating characteristics. A third component is identifying the critical objectives that a CCL System is to serve. In the case of DGIS, the criteria for CCL objectives are the DGIS information processing operations.

The decision to convert to and continue our CCL development with Artificial Intelligence tools is explained. PROLOG was chosen because it is a simple but powerful relational programming language based on the concept of programming in logic. Our effort merges PROLOG and C capabilities, to provide the DGIS user an AI-based searcher assistant interface. The purpose of this interface is to make the human-machine interaction more human-like on DGIS.

With the transition to an AI-based CCL System, the goals of DGIS CCL were re-constituted to incorporate AI-driven capabilities as follows:

- Standardize a command language to communicate with all bibliographic databases.
- Create a CCL System that assists the user in searching unfamiliar database systems.
- Provide the means for a user-friendly search session.
- Provide the means for an intelligent, user-useful search session.
- Have flexibility to adapt easily to changes and enhancements.
## EXECUTIVE SUMMARY

I. INTRODUCTION

II. BACKGROUND

III. PROJECT CONCEPT AND MODULES

IV. OUR FIRST PROTOTYPING DEVELOPMENTS

V. THE FIRST PROTOTYPES

VI. FIRST PROTOTYPE RESULTS

VII. MAJOR PROBLEM

VIII. THE DECISION FOR ARTIFICIAL INTELLIGENCE

REFERENCES

APPENDIX A - DIALOG C Prototype

APPENDIX B - DROLS C Prototype

APPENDIX C - PROLOG Programming for DROLS: First Page

APPENDIX D - DROLS CCL & PROLOG

APPENDIX E - NISO Common Commands
I. INTRODUCTION

The Defense Technical Information Center (DTIC) of the U.S. Department of Defense (DoD) has sponsored development of a DoD Gateway Information System (DGIS) since 1982. The purpose of DGIS is to provide online, streamlined methods for identifying, accessing, searching and analyzing data from heterogeneous databases of interest to the DoD community [CGA85]. The following figure is the top menu of DGIS, and shows its core operations to achieve this purpose [KADe6):

Present-day access to information resources is constrained, since each database system has its own complex access procedure and command language. Additionally, results from multiple databases cannot be combined or analyzed easily by the user. The DGIS provides DoD researchers and managers access to many different databases using a single, simple access procedure. The system has a library of automated routines by which the user may reformat, analyze, and tailor aggregated information into a form useful to the enduser [CGA86]. The intent of DGIS is to provide the user a useful tool to collect, process and transfer information.
II. BACKGROUND

One of the primary development goals of the DGIS was to design and implement a common command language (CCL). The prototype DGIS, established in January, 1986, provides access to a multitude of information systems. But the user must know the native command language and operating characteristics of each system. The Program Manager for the DoD Gateway Information System (DGIS) assigned a Common Command Language (CCL) design activity in April 1986.

A team of three people began the effort. The team consisted of a project manager, a computer systems analyst/programmer, and a technical information specialist. The technical information specialist was responsible for researching and defining the command language information requirements. This information would be used by the programmer to program the CCL software.

The Project Officer defined the boundaries and objective of the CCL as follows:

1. The mission of DGIS is to provide uniform access to remote, diverse databases, aggregate information from those databases, and provide post-processing routines. The end product of the DGIS will be an information product that is tailored to meet the needs of the user.

2. A widely recognized problem in accessing multiple diverse databases is the absolute requirement to search and retrieve information from them in their idiosyncratic command languages. This requirement forces the searcher to learn the native command languages of different database systems in order to acquire a comprehensive set of information.

3. The Objective of the DGIS Common Command Language project, therefore, is to determine, formulate, and structure a DGIS common access approach that facilitates access to multiple databases in a unified manner.

4. DGIS CCL will be optional, at the discretion of the user; if the user so desires, native command language searching is available.

5. Various CCL modes are to be explored including:

   (1) Formulation of a DGIS CCL command set.

   (2) The ability to search any database system with the native language of a system already familiar to the searcher. In this case, the DGIS CCL serves as a transparent conversion mechanism.

   (3) Query languages, forms and methods provided by the fourth generation languages and artificial intelligence.

5. The DGIS Common Command Language development team will monitor developments of other organizations, especially that of the National Information Standards Organization (NISO) and its Subcommittee G for Common Command Language for Online Interactive Information Retrieval.

III. PROJECT CONCEPT AND MODULES

1. CONCEPT ELEMENTS

As we developed the project concept the complexity of the activity became apparent. We organized the activity in a modular structure for easier management, as follows.

Requirements Development Module
Common Access Development Module
CCL Standards Module

The Requirements and Access development modules evolved into high activity, distinct development environments. The Standards module evolved into a support
function requiring only periodic activity.

The plan of action was formulated to take advantage of all DGIS capabilities. For example, in addition to CCL access to the information systems, such as DIALOG, BRS, and DROLS, we wanted CCL access to those systems for which we had postprocessing routines already established through parallel DGIS projects. For the most part, these were the same systems. The DGIS CCL project activity, therefore, was planned as follows:

THE REQUIREMENTS DEVELOPMENT MODULE

The database sets of commonly used commands, i.e., those most frequently required by the user community, were identified. This established a core set of common function commands. This module then entailed:

a. Mapping the commands/functions of the initial DGIS-targetted databases:
   - Defense RDT&E On-Line System (DROLS) (Dept. of Defense)
   - Work Unit Information System (WUIS) (Dept. of Defense)
   - Manpower and Training Information System (MTRIS) (Dept. of Defense)
   - DOE/RECON (Dept. of Energy)
   - NASA/RECON (National Aeronautics and Space Administration)
   - DIALOG
   - BRS
   - ORBIT

b. Designing a DGIS common command language structure. The design was to incorporate standards. This activity was to define the syntax and semantics of a native command language, and bridge it to the Common Command Language.

c. Designing command language translators. This was to be based on merging the mapping requirements with the DGIS UNIX operating system C language programming capabilities.

d. Designing the DGIS Common Command Language communication software for interacting with the target information systems.

COMMON ACCESS DEVELOPMENT MODULE

In this module we explored the potential applications of fourth generation languages and artificial intelligence. We identified their potentials for creating unique methods of searching in and retrieving from diverse databases. This included screens, utilities, menus, windows, query organizing, natural language, knowledge base systems, et al. These technology applications could provide the capability for true simultaneous searching in multiple databases. We partitioned this development into two phases:

   Phase 1 (Near Term): Establishment of single database searching.
   Phase 2 (Long Term): Establishment of simultaneous database searching.

CCL STANDARDS MODULE

The intent of this module was to track national and international CCL standards for use in our DGIS CCL effort. We opted to follow the common command language standard adopted by the U.S. National Information Standards Organization (NISO). We were to track CCL activities by:

a. Liaison with NISO.
b. Participation in CCL activities at conferences and meetings.
c. Determination of the applicability of CCL standards to DGIS CCL.
d. Liaison with organizations actively involved in CCL developments and applications.

The modules described above provided the building blocks of the project.
2. MODULE DEVELOPMENT

STANDARD CCL: We adopted the NISO Draft Standard to serve as the foundation of the DGIS CCL. The standard was very useful not only because the online searching functions were labeled, but also the 1987 draft version included the Backus-Naur Form (BNF) for the command language syntax [NISO87]. We also provided comments on the 1986 draft to NISO.

MAPPING: We learned very early that we were not dealing with commands, but with functions. Mapping involved identifying the functions, and making the bridge between the database function label and the CCL function label. Intensive reviews of database functions, commands and operating characteristics have been made by the technical information specialist, and continue to be made by her as we progress from one database to the next. This activity, of course, has made this person a valuable and knowledgeable database resources expert.

We partitioned the functions and their commands into three groups. Group assignment was made by criticality and frequency of use [BRL87]. These groups are:

- Group I: Most commonly used, basic functions.
- Group II: Remaining common functions.
- Group III: Database Idiosyncratic functions.

The selection for Group I was based on the following criteria:

- a. Common to all databases.
- b. Essential for a complete search and retrieval session.
- c. Small enough group so that several working prototypes could be programmed within a reasonable length of time, e.g., four to six months.

The NISO commands START, CHOOSE, FIND, DISPLAY, and STOP were selected to form Group I. DGIS already had automatic connects and disconnects established, however, leaving only CHOOSE, FIND, and DISPLAY to analyze. The CCL elements associated with Group I are shown in the table below, from the perspective of the CCL user [KAD87]:

<table>
<thead>
<tr>
<th>DGIS</th>
<th>NISO</th>
<th>Diverse Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Conn</td>
<td>START</td>
<td>[Logon Routines]</td>
</tr>
<tr>
<td>CHOOSE</td>
<td>DIALOG: b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DROLS: $s[database name]$</td>
<td></td>
</tr>
<tr>
<td>FIND</td>
<td>DIALOG: s, ss</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DROLS: $s[database name]$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORBIT: [search term]</td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>DIALOG: t</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DROLS: $dsd$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORBIT: print</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>DIALOG: logoff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DROLS: $term$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORBIT: stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WSA: signoff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRS: ...g</td>
<td></td>
</tr>
</tbody>
</table>

Table: Comparison of CCL and Native Command Language Examples.
The command function set for Group II is lengthy, but the primary objective was first to validate the working prototypes based on the Group I commands, and then add on the remaining functions. Group II, therefore, is comprised of (and informally broken down as):

(a) SCAN  PRINT  EXPLAIN
(b) MORE  REVIEW  HELP
(c) BACK  SORT  SHOW
RELATE  SAVE  SET
DELETE  DEFINE

Group III, idiosyncratic commands, are those peculiar to the individual systems and without a common command standard. Our 'pro tempore' solution for those commands was to allow the user to switch into database native command language mode. This decision is subject to further analysis.

COMMON ACCESS: We made a preliminary review of Fourth Generation Language (4GL) capabilities. An important reason for this review was that the DGIS Directory of Online Resources [JCE86], a component of the DGIS software, is being developed on the INGRES DBMS. Through the Directory, users can search by subject for databases that are relevant to their queries. Eventually DGIS CCL will be integrated so that once the relevant databases are determined by the Directory, a user’s query will be automatically filtered through the CCL capability to search those databases simultaneously.

We looked at INGRES 4GL features, such as query organizing, query-by-form, windowing, database building relative to CCL, forms generating, etc. We concluded that 4GL applications might be useful in advanced CCL versions, but initial emphasis was to develop a prototype CCL as quickly as possible. We decided, therefore, to concentrate on C language prototype programming with the purpose of validating simple prototypes and then pursuing more advanced features. The decision turned out to be fortuitous, for later we made the jump to artificial intelligence applications 'vis-a-vis' 4GL.

IV. OUR FIRST PROTOTYPING DEVELOPMENTS

Our initial effort to implement CCL was involved selecting simple approach, and trying to get several prototypes up quickly. Once in place, we could then experiment with them and learn from our experiences. Programming was done in C, merged with two UNIX utilities that were immediately adaptable to CCL needs. Those two utilities were LEX (generator of lexical analysis programs), and YACC (Yet Another Compiler-Compiler) [UNIXol]. LEX was used for lexical analysis of the CCL prototype C programs, YACC for the syntactical analysis. C was used to implement all remaining semantic processing and miscellaneous tasks. [TDTpip]

Communications was a highly critical element. We needed some type of communications program to talk with databases. DGIS had NAM (Network Access Machine) software agents available in the DGIS software for connecting users to databases for native language searching. A technical review was made of the software, and it was ascertained that it provided the needed capabilities for communicating with remote databases in CCL. NAM provided a utility for:

a. Establishing the connection.
b. Validating user access.
c. Logging on to the target database, including entry of the logon codes.

The NAM agent was adapted to CCL, therefore, for communicating the command and response in searching the remote database system. [TDTpip]
V. THE FIRST PROTOTYPES

Our first prototype, achieved five months after beginning the effort (August 1986), was DGIS CCL for DIALOG. DIALOG was chosen because it was a system with which many users in the DoD community are familiar, and find easy to use.

The DIALOG prototype was followed by BRS, NASA-RECON, and DROLS in fairly rapid succession. BRS was chosen because it was another major vendor system with many databases, NASA-RECON because it was a Federal government database system, and finally DROLS, DTIC's database system.

The following, a BRS session, shows how the CCL BRS prototype works. Please note that at this stage, CCL is only a substitute for BRS native commands. Other than the commands, BRS must be addressed on its terms. The CCL invocations are indicated by the prompt 'CCL >'. The line following this prompt is the BRS command entry, which echoes the CCL entry:

```
* Connect brs.
Attempting telephone connection at 2400 baud to TYPNET.
(...etc.)
Login Completed
ENTER Y OR N FOR BROADCAST MESSAGE:_ n
ENTER DATABASE NAME:_
Starting up CCL filter
*** Welcome to CCL ***
CCL > choose psych
..change/psych
*SIGN ON 11:22:50 10/31/86
PSYC 1967 - NOV 1986
BRS SEARCH MODE - ENTER QUERY
1:
CCL > find sleep
..search
BRS SEARCH MODE - ENTER QUERY
1: sleep
RESULT 6902 DOCUMENTS
2:
CCL > display au,ti 1-3
..print
```

VI. FIRST PROTOTYPE RESULTS

We terminated C-programming with completion of the four prototypes. The experience we gained was immeasurably useful. The following issues and features resulted from this first prototyping:

1. The Adaptation of the NAM Connection Agent: As mentioned above, NAM software for connecting with remote databases was already available. Once the sign-on is completed, the user is connected directly with the database. The
user then invokes the CCL translator.

2. CCL Invocation: Currently, once one has accessed a database system through the HAM connection agent, one may invoke the CCL translator with a special key.

3. CCL Translators: The creation of prototype CCL translators taught us that each information system is individualistic and must be treated as such. The translator programming is totally dependent on the mapping requirements for each system. The programmer must also detect anything "hidden" in the target database system that is needed for a response. The CCL translator is a filter that is toggled on and off by a special key. Once activated, it intercepts all CCL commands from the user, translates the command, sends the translation (i.e., the target database native command) for execution, and brings the results back to the user [TDTPip]. The translator is deactivated by the conventional <CNTL>d (exit from a process).

4. Native Command Language Option: The option to use the native command language was necessary when we were prototyping only the basic commonly used commands (Group I already mentioned). The entry of a native command was made very simple: at the CCL prompt, one precedes the native command with a backward slash (\) to tell the translator that the native command is coming, e.g.,

CCL > \s (for DIALOG 'select')

5. The CCL Prompt: The prompt 'CCL >' was incorporated as a reminder to the user that one has invoked the CCL utility.

6. CCL Command Verification: When the user invokes a common command, the translation of the invocation is echoed in the database system structure, e.g., for DROLS:

CCL > find artificial intelligence and psychology

(echo) str artificial intelligence and psychology

CCL: Searching...

The echo may also be turned off, currently with the command: CCL> noecho

7. Online Documentation: The HELP feature to show the user how to use the CCL. The documentation, in very abbreviated form, covers the CCL commands available. For example (DIALOG):

CCL > help find

CCL format
find <term> ...
DIALOG2 format
select <term> ...
DESCRIPTION
Initiate a search.

8. Shell Spawning while in CCL: We incorporated the capability to exploit a UNIX shell, file, or utility while in the CCL. Use of the capability is at the user's discretion; for example, the user may want to list one's files as a review measure while searching a database. The signal to the CCL translator is an initial bang (!), e.g.,

CCL > !ls (for listing files)
CCL > !w (for seeing who is on the system)

etc.
9. New Commands: In developing the DGIS CCL we found that the NISO standard did not cover several items that we deemed useful. Usefulness was based on the following criteria:

a. Functions, prevalent in systems, that aided the user; an example is successive session cost display.
b. Functions, not prevalent, seen as highly useful; e.g., listing the accession number of finds.
c. Functions that we found were needed for an operative CCL; an example is cancelling the translation echo display at one's discretion.

The non-NISO commands that we incorporated under the first prototyping are:

- COMBINE Do Boolean operations (and, or, not) on previously created sets.
- COST Display session cost thusfar.
- EXECUTE Execute a previously saved search strategy (in target database).
- LIST List accession numbers of search results.
- NOECHO Cancel native command function echo to CCL command function.

10. NISO Standard Common Commands Incorporated in the First Prototyping: As we developed the four prototypes, we added on commands to enhance the prototype capabilities. We used, therefore:

- CHOOSE (Group I)
- HELP (for target database help)
- FIND (Group I)
- DISPLAY (Group I)
- RELATE
- MORE -- in NISO 86, changed to FORWARD in NISO 87.
- BACK

The standard START and STOP (Group I) are taken care of by the DGIS automatic connect and disconnect.

11. CCL System Menu Development: As we progressed through the four prototypes, the systems became more terse. We were exposed to many unique features. The programmer was totally unfamiliar with DROLS which is a terse, no-assist system with access limited to a strictly controlled user community. As a skillful programmer he was, therefore, an ideal person to look at DROLS and determine its appropriate functional CCL requirements.

The very terseness of DROLS (including that lack of a prompt) generated the need to experiment with menu sets to step the unfamiliar user through the database. These menus, basically, provide functional information the expert DROLS searcher knows, but is not on the system. CCL menu examples are:

When invoking CCL CHOOSE in DROLS without designating which database -

```
CCL > choose
Select one of the following files:
1. Current Reports
2. Technical Reports
3. New Accessions
4. Work Units
Please enter your choice (1-4) -> 2
Technical Reports file is selected.
CCL > find ... (etc.)
```

When invoking CCL DISPLAY for search results in DROLS without designating a
VII. MAJOR PROBLEM

Each prototype raised issues and problems which we used to refine the successive prototype. As the prototypes progressed, various problems in working with them lead to solutions such as HELP features and menus as mentioned above.

The major problem, however, surfaced as a result of our cumulative experience. We learned that creating "Common Command Language" was NOT a panacea. Programming a "standard" command language was in actuality only substituting one command language for another.

This was most apparent when the DISPLAY function is employed. Quite factually, if the user does not know the DISPLAY formats of an unfamiliar system, one cannot see results. A command with less knowledge requirements is the FIND function. Using FIND, the user is very likely to be able to enter the query and foment results. But any function involving a display is likely to be dead-ended in no display. Substituting CCL for the native command language simply does not obviate the need for referral to a system's user documentation, which gives instruction in terms of its native command language.

Another example is the CHOOSE function. Some systems identify databases by number, others by acronym. For BRS, one must enter CHOOSE NTIS; in DIALOG, CHOOSE 6. The hydra of options and formats keeps cropping up. Each system must be addressed individually, with the goal of having some central pattern program to draw upon. The crutch we have used for the C language-based CCL prototype is the menu.

The creation of a CCL is only one component of the "CCL-need" issue. A second component is creation of a CCL System that allows a user to search in unfamiliar database systems without needing to know that system's operating characteristics. A third component is identifying the critical purposes that a CCL system is to serve.

In the case of DGIS, the critical CCL purposes are defined by understanding the DGIS information processing operations, particularly in postprocessing downloaded files. A DGIS postprocessing requirement is to have a tagged citation for translation. Downloaded citations must be translated into the
DGIS standard citation format before the automated processing routines can be applied.

This necessity is an example of a criterion for a DGIS CCL system. The CCL system must include function default results for those users unfamiliar with a database, particularly for DISPLAY. The default, on simple invocation of DISPLAY, will provide the fully tagged citation. Additional elements, such as menus and question prompts, e.g., “DISPLAY on last set? y/n,” must also be incorporated.

The case of CHOOSE represents another problem environment. In DGIS the solution is the eventual integration of CCL with the Directory of Online Resources. When this is accomplished, the query will be forwarded automatically to the relevant databases through CCL.

The real demon for CCL has turned out to be the idiosyncratic operating characteristics of each database system.

VIII. THE DECISION FOR ARTIFICIAL INTELLIGENCE

1. THE CAUSE

The rigidity and constraints of a straight algorithmic program-based CCL discovered during the prototypings lead us to exploring the potential of artificial intelligence. The natural language and expert system possibilities of AI were very appealing. The project’s programming technical expert, in having an amount of AI background, reviewed the main AI programming languages. His recommendation was to explore AI applications with PROLOG, a simple but powerful relational programming language based on the idea of programming in logic [BA-86].

The initial technical reasons for selecting PROLOG were [TDtpip]:

a. The Reversibility of PROLOG In determining object relationships, a program can be written establishing those relationships, with the inverse of the relationship inherent in the program.

b. Its Database Capability In that PROLOG has its own internal databases, this feature allows a PROLOG program to manipulate codes as relations that can be asserted or deleted. PROLOG incorporation in CCL includes extending to external databases, e.g., INGRES DBMS in the DGIS software, to achieve the flexibility of storing knowledge in both PROLOG internal databases and traditional external databases. This allows including more powerful database technology in the program system for greater performance and easier use of DGIS by the enduser.

c. The Separation of Logic and Control A PROLOG program amalgamates rules and facts, basically making one also the other. Although they are governed by a default execution control, the control can be easily supplemented or replaced by more powerful meta-rules also coded in PROLOG.

d. Object Inheritance and Message Passing These are two powerful features of object language methodology. Both are easily implemented and embedded in PROLOG. Both features are elemental for the more graceful functioning of CCL.

2. THE CONCEPTUAL RESTRUCTURING OF CCL

Using C language programming, the basic CCL elements consisted of the user, the CCL, the database language processor, and the database information accessed. The jump to PROLOG opened new possibilities in which CCL now could be developed as a knowledge-based system. The CCL conceptual structure now became [TDtpip]:
To the DGIS user making use of CCL, the fact that CCL will be PROLOG-driven is transparent. The PROLOG CCL, however, in serving the user, will draw on the command language knowledge base, and also a CCL-user profile knowledge base (still to be developed). The user's query and profile data will be controlled through the control program blackboard, which will coordinate translation and communications in a continuous real-time system mode [BA-86] through the NAM connection agent. The NAM agent passes the communications to and from the target database system's command language processor for searching on the database information.

With the transition to an AI-based CCL Systems, the goals of the DGIS CCL have been re-constituted to incorporate AI-supported capabilities as follows:

a. One command language to communicate with all bibliographic databases.

b. Creation a CCL System that assists the user in searching unfamiliar database systems.

c. Provision of a user-friendly search session.

d. Provision of an intelligent, user-useful search session.

e. Flexibility to adapt easily to changes and enhancements.

3. OTHER CHANGES IN THE ACTIVITY

Because of the relative ease of learning PROLOG programming, another effect of making the transition to PROLOG was to transfer much of that programming from the technical expert to the requirements expert. This change was made at the suggestion of the technical expert. The requirements expert took the training provided by the PROLOG vendor, and began PROLOG programming under the tutelage of the technical expert. This change gave her fuller control of the command requirements, from command language researching to command language knowledge base building. This also allowed the technical expert to concentrate on the knowledge base - database system connector programs, in itself a programming-intensive activity. The following is a small taste of the CCL knowledge base PROLOG programming, for the CCL DELETE command relative to DROLS:
This PROLOG programming sample is for the CCL DELETE command relative to DROLS in the command language knowledge base. The knowledge base in development as of July 1987 is 10 regular pages long.

4. FUTURE DIRECTIONS

Our next phase in CCL is a melding of PROLOG implementation, expert system building, and C supplementary programming. The PROLOG-based CCL has two parts [TDTpip]. One part is fixed, in compiled PROLOG code; the second is variable, in interpretive PROLOG code. The variable part loads and processes information from the two knowledge bases (KB), the command language KB and the user profile KB. Appropriate tools will be incorporated to maintain the KBs (adding, deleting, modifying information). We are currently (August 1987) procuring an artificial intelligence processor system and expert system building software tool. The processor will be networked to the DGIS computer system, and will serve to both develop and maintain AI applications in CCL and other AI applications on DGIS [KAD87b].

We are investigating several schemes for KB organization. In general, we plan to couple PROLOG with a Relational DBMS (RDBMS) where large KBs (most of which are facts) will reside. The technical issue here is the interface between PROLOG and the RDBMS (likely INGRES). We intend to make this interface through SQL (standard query language) so that it can work with any RDBMS, rather than only with INGRES. [TDTpip]

Other CCL system factors are:

a. CCL Integration with Other DGIS Functions << Other DGIS operations are
potentials for AI applications, with which to link with CCL. One is the DGIS Directory of Online Resources, wherein a user's query resources are identified and communicated with automatically and simultaneously. Another is the DGIS postprocessing routines, with which the multiple resource responses are automatically downloaded, translated, merged, and processed (or tailored), based on a one-pass instruction entry with which the user invokes the whole process.

b. Planning Capability

Includes the preliminary structuring of multiple queries and the combining of target databases' result sets.

c. Learning Capability for CCL

Employing learning solution paths [BA-86] for optimizing the information added to the command language KB and the user profile KB.

d. Migrate to Natural Language

The NISO and appended CCL will be the backbone of the DGIS CCL, but in migrating to Natural Language dialogue, will become transparent in a command language translation supporting role.

e. Provide Simultaneous Database Access Capability

That is, true concurrent connecting with, searching in, and downloading of results from multiple systems.

All this to bring about the resolution of, as the technical information specialist states [BRL87]: "The problem of multiple command languages has plagued online searchers since the creation of the second online database."

The appendices that follow show examples of C and PROLOG prototypes. The last appendix lists the NISO CCL commands.
REFERENCES


[UNIXol] University of California-Berkeley. UNIX Programmer’s Manual. LEX(1); YACC(1); NAM(1T). Online manual, UNIX Operating System BSD 4.2.
dialog.c

Date: Thu Sep 11 09:47:29 1986
From: duc (Duc Tran)
Subject: note on how to run the CCL-DIALOG2

SHORT NOTE ON HOW TO RUN THE CCL-DIALOG2 TRANSLATOR

The CCL-DIALOG2 translator is built as an input filter to NAM.
In order to invoke this filter the user must first invoke the NAM agent that
does the connection to DIALOG2. This is done by typing:

% nam cldialog

After connecting to DIALOG2 the user is, by default, talking to
DIALOG2 in its native command language. The standard DIALOG2 prompt of
question mark "?" appears at each line of input. The CCL translator is
activated only by the two key sequence ESC '^T. A CCL message displays:

Welcome to CCL
Type
  help ccl
for the current status

At this point the user can enter one CCL command per command line.
The CCL prompt is:

CCL >

The first time user should type

CCL > help

to learn how to use the CCL commands.

Remember that at this point all the input from the user will go through CCL.
The valid CCL commands are translated into DIALOG2 and sent to DIALOG2 for
execution.

Native DIALOG2 commands can be accepted if a backslash (\) is detected as the
first input character at the CCL command line.

A shell process can be started if a bang (!) is entered at the CCL prompt as
the first character of the command entry.

CCL exits when a "D is detected; control is given to NAM.
One may then continue with the native DIALOG2 commands, or exit NAM by ESC "D ".

A sample session:

CCL > help ccl
CCL > help
'CCL > help help
CCL > choose 1
CCL > find computer
CCL > relate computer
CCL > more
CCL > back
CCL > list -I
CCL > display #1
CCL > \select education
I have a better 'saveon' CCL-DROLS for you. Note that:
- the 'choose' command without argument or with
  wrong arg. [file] is presented with a menu of
  file names. If user uses the correct file name as
  the argument of 'choose' then the menu is not presented,
  just the confirmation message.
- the 'list' command is added to CCL to map it to
  LSR of DROLS which I think is useful.
- 'more' for next display.

The ccdrols is installed. You can try it by typing

% nam ccdrols

Remember once connection is made use ESC 'Y to invoke CCL.
Thus CCL-DROLS is now demonstrable.

saveon starts here

Saveon Transcript started on Fri Mar 13 08:19:06 1987
1 % nam ccdrols

Attempting telephone connection at 1200 baud to DROLS.
Dialing...
Dialing done.
Connection established to DROLS.
Login complete.

ROTYE ON-LINE SYSTEM MAR 13, 1987

PLEASE ENTER YOUR TERMINAL IDENTIFICATION

WELCOME ON LINE - DATE 031387 TIME 081957

IF YOU DISPLAY ENTRIES OF REPORTS WITH REFERENCES MARKED
EXPORT CONTROL THE FOLLOWING WARNING APPLIES:

WARNING

THIS DOCUMENT CONTAINS TECHNICAL DATA WHICH EXPORT IS
RESTRICTED BY THE ARMS EXPORT CONTROL ACT (TITLE 22, U.S.C.,
SEC. 2751 ET SEQ.) OR EXECUTIVE ORDER 12402. VIOLATIONS OF
THESE EXPORT LAWS ARE SUBJECT TO SEVERE CRIMINAL PENALTIES.
drols.c2

---
ADA103311
ADA103300
ADA100318
ADA048919
ADA036917
ADA036915

CCL > noecho

CCL > display

Select data type to be displayed:
1. Search Results.
2. Qualifed Results.
5. Single Current File Number.
6. Single Work Unit Number.
7. Available Files.
8. Information Log.
10. Inverted File.

Please enter your choice (1-10) --> 1
Please enter a field no [0 for end of field list] --> 3
Please enter a field no [0 for end of field list] --> 21
Please enter a field no [0 for end of field list] --> 23
Please enter a field no [0 for end of field list] --> 0

CCL > (Sub command for display mode)

Select a display mode:
1. Item by item display.
2. Continuous display.
Please enter your choice (1-2) --> 1

---
1 OF 20
--- AD NUMBER: P003939
--- ENTRY CLASSIFICATION: UNCLASSIFIED
--- DESCRIPTORS: "ARTIFICIAL INTELLIGENCE, "ELECTRONIC EQUIPMENT, "PERCEPTION (PSYCHOLOGY), PSYCHOLOGY, PROBLEM SOLVING, MAINTENANCE,
--- TRAINING

CCL > more

--- 3 OF 20
--- AD NUMBER: B108147
--- ENTRY CLASSIFICATION: UNCLASSIFIED
--- DESCRIPTORS: "PSYCHOLOGY, "ARTIFICIAL INTELLIGENCE, COMPUTER PROGRAMS, PROGRAMMING LANGUAGES, HUMAN FACTORS ENGINEERING,
--- HARDWARE, PHYSICS, MANAGEMENT

CCL > 1
1 % who
batesple ttyhl Mar 13 08:09
gillean ttyhb Mar 13 08:20
pdesmo ttyhc Mar 13 08:17
utys ttyhf Mar 13 08:20
duc ttyio Mar 13 08:11
njones ttyjg Mar 13 08:03
cacheko ttyjc Mar 13 08:20
2 % exit
3 %
CCL > "D
"** Goodbye **

You are now back to NAM.

Logging off DROLS.
drols.pl

:- module(drols, [forward, back, find, cpl, display, scan, combine, sort, delete, explain, print, cost, define, whatis, execute, relate, save, set, show, stop, keep, error], [nonexecutable]).

% Top level CCL

% CCL to DRLS translation rules

% CCL to CRLS rule for forward
forward_cmd --> forward_cmd, [dbname, dbname].

% CCL to CRLS rule for back
back_cmd --> back_cmd, [dbname, dbname].

% CCL to CRLS rule for find
find_cmd --> find_cmd, [dbname, dbname, dbname].

% CCL to CRLS rule for cpl
cc1_cmd --> cpl_cmd, [dbname, dbname].

% CCL to CRLS rule for display
display_cmd --> display_cmd, [dbname, dbname].

% CCL to CRLS rule for scan
scan_cmd --> scan_cmd, [dbname, dbname].

% CCL to CRLS rule for combine
combine_cmd --> combine_cmd, [dbname, dbname].

% CCL to CRLS rule for sort
sort_cmd --> sort_cmd, [dbname, dbname].

% CCL to CRLS rule for delete
delete_cmd --> delete_cmd, [dbname, dbname].

% CCL to CRLS rule for explain
explain_cmd --> explain_cmd, [dbname, dbname].

% CCL to CRLS rule for print
print_cmd --> print_cmd, [dbname, dbname].

% CCL to CRLS rule for cost
cost_cmd --> cost_cmd, [dbname, dbname].

% CCL to CRLS rule for define
define_cmd --> define_cmd, [dbname, dbname].

% CCL to CRLS rule for whatis
whatis_cmd --> whatis_cmd, [dbname, dbname].

% CCL to CRLS rule for execute
execute_cmd --> execute_cmd, [dbname, dbname].

% CCL to CRLS rule for relate
relate_cmd --> relate_cmd, [dbname, dbname].

% CCL to CRLS rule for save
save_cmd --> save_cmd, [dbname, dbname].

% CCL to CRLS rule for set
set_cmd --> set_cmd, [dbname, dbname].

% CCL to CRLS rule for show
show_cmd --> show_cmd, [dbname, dbname].

% CCL to CRLS rule for stop
stop_cmd --> stop_cmd, [dbname, dbname].

% CCL to CRLS rule for keep
keep_cmd --> keep_cmd, [dbname, dbname].

% CCL to CRLS rule for error
error_cmd --> error_cmd, [dbname, dbname].
drols.pl

Date: Thu May 7 10:40:07 1987
From: duc (Duc Tran)
Subject: CCL/prolog translator and DROLS

Enclosed is a sample session of the CCL translator for DROLS. Notice the improvement in the way DROLS KB is loaded. You now are put in CCL right away, and DROLS Knowledge Base is loaded by typing 'start drols'.

Allen's remark about "CCL problem" that the users do have to know something about the database one is working on is quite accurate with our current experiences. I hope that after this CCL/prolog prototype phase is over with five databases we will have time to reassess and perhaps make the "CCL problem" less obvious.

-duc-

Sample transcript started on Thu May 7 10:26:10 1987
duc[1] % trans

(Enter

CCL> start database
to load a knowledge base).

*** Welcome to CCL ******

CCL > start drols
{/al/prolog/translator/drols.pl consulted (6.650 sec 11824 bytes)}

CCL > choose

(1) Current Reports
(2) Technical Reports
(3) New Accessions Reports
(4) Work Units

Selection --> 1
CCL > sort

Please enter sort field (separated by space)
1 2 3

Please enter sort sequence(A or D)
a
$SOQ#d
1
2
3
4
5
a
END

CCL > choose

(1) Current Reports
(2) Technical Reports
(3) New Accessions Reports
(4) Work Units

Selection --> 2
APPENDIX E

NISO COMMON COMMANDS

National Information Standards Organization. Z39, Committee G

Z39.58

Common Command Language for Online Interactive Information Retrieval

Table 1

Primary Command Names and Abbreviations

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACK</td>
<td>BAC</td>
<td>To view data preceding displayed data or items on a list.</td>
</tr>
<tr>
<td>CHOOSE</td>
<td>CHO</td>
<td>To select file(s) or database(s) to be searched.</td>
</tr>
<tr>
<td>DEFINE</td>
<td>DEF</td>
<td>To create user macros, to rename a command name, or to name a command expression with a word.</td>
</tr>
<tr>
<td>DELETE</td>
<td>DEL</td>
<td>To delete search strategies, search sets, search results, PRINT requests, or DEFINEd commands.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>DIS</td>
<td>To view online the results of searches of the database(s).</td>
</tr>
<tr>
<td>EXPLAIN</td>
<td>EXP</td>
<td>To obtain information about the system, its use, and its database(s).</td>
</tr>
<tr>
<td>FIND</td>
<td>FIN</td>
<td>To enter a search statement.</td>
</tr>
<tr>
<td>HELP</td>
<td>HEL</td>
<td>To directly obtain online assistance or instruction specific to the context of the interaction.</td>
</tr>
<tr>
<td>KEEP</td>
<td>KEE</td>
<td>To identify and save specific result sets and/or particular records from previous search results.</td>
</tr>
<tr>
<td>FORWARD</td>
<td>FOR</td>
<td>To view continuing data, or data following displayed data or items on a list.</td>
</tr>
<tr>
<td>PRINT</td>
<td>PRI</td>
<td>To request offline printing.</td>
</tr>
<tr>
<td>RELATE</td>
<td>REL</td>
<td>To view terms logically related to search term.</td>
</tr>
<tr>
<td>REVIEW</td>
<td>REV</td>
<td>To view search history, i.e., search statements.</td>
</tr>
<tr>
<td>SAVE</td>
<td>SAV</td>
<td>To save search strategies for subsequent use.</td>
</tr>
<tr>
<td>SCAN</td>
<td>SCA</td>
<td>To view an ordered list of index terms.</td>
</tr>
<tr>
<td>SET</td>
<td>SET</td>
<td>To set or override default parameter values.</td>
</tr>
<tr>
<td>SHOW</td>
<td>SHO</td>
<td>To view session parameter default values and non-instructional system or session information.</td>
</tr>
<tr>
<td>SORT</td>
<td>SOR</td>
<td>To arrange search results by specified field(s).</td>
</tr>
<tr>
<td>START</td>
<td>STA</td>
<td>To initiate a session.</td>
</tr>
<tr>
<td>STOP</td>
<td>STO</td>
<td>To terminate a session.</td>
</tr>
</tbody>
</table>

Source: NISO87, p.27