NEW PROPOSAL FOR A COMMON COMMAND LANGUAGE IN INFORMATION SYSTEMS

by

Yves VANDER AUWERA

Luc BERNARD

Technical Report n° 118

ABSTRACT

A new command language based on the proposal by A.E. Negus is introduced for bibliographic data base searches. It is adapted to any bibliographic database (universality), provides large and powerfull facilities. Finally it is simple to use and to analyse.

KEYWORDS

Information systems

Command Language

Search statement

Many command languages have yet been developped for information systems.
Users have to access to several systems usually with different command languages.
Therefore those languages should tend to some standardisation.
A first attempt was made by A.E. NEGUS who developped in 1977 a common command language (abbreviated as CCL) for EURONET-DIANE ([2]).
The grammar he introduced was very simplified and therefore allows the introduction of commands syntactically correct but meaningless as in the following example : FIND CC = 5A . . . . . 1D $ $ 3B ... u$$9L./CC<CC
Moreover the grammar contains some ambiguities and needs a complicated analyser (LALR (1) techniques which are common for complex high level programming languages are not sufficient).
Some relevant remarks have already been made ([1]) concerning design errors (e.g. a string cannot contain letters only, which is in contradiction with some examples given in the EURONET guideline [2]).
The object of the present paper is to define a language syntax which although it is a bit more complicated (311 rules to describe the grammar versus 300 for the EURONET guideline) simultaneously is easier to analyse and more realistic.
Many meaningless commands are syntactically avoided and LALR (1) techniques are sufficient to parse it.
To describe the syntax the following formalism will be used :

[entity] means that the entity is optional

$\left\{\begin{array}{l}\text{alternative}_1\\ \text{alternative}_2\\ \vdots\end{array}\right\}$ means that one of the alternatives must be choosen

$\{\text{entity}\}^*$ the entity is repeated any number of times,

$\{\text{entity}\}^+$ but at least once for +

underlined words are reserved words

<entities> are syntactical entities.

The grammar is given in Backus Naur Form [3](BNF) in Appendix 3 .

1. INITIALISATIONS AND TERMINATIONS

1.1 The BASE command is used to open a session on a data base. It means that the commands which follow, will be related to the data base invoked, until another base command arises.

The command syntax is :

<base-command> ::= { base / bs } [<string>{;<base-par>}*]

<base-par> ::= { pw = <string> / ed = <date>to<date> / qn = <unsigned-number> / tl = <language-code> }

If a string is specified after the keyword base or bs, it designates the invoked data base. If no data base is specified, the system will prompt the user to select a data base. As can be seen the command may accept 4 differents parameters :

- pw to specify the data base password
- ed to restrict the field of information the user will access
- tl to specify the thesaurus language

<language-code> ::= { alba / arab / : / yugo }

- qn to initialize the query number to a specified value.

  Each search is characterized by a query number (QN), which is very usefull to define SDI profile. Normally, they should be automatically provided by the system.

Ex.: base infobank; pw = lock; ed = 77.03 to 80.11.30

It realises the connection to the data base "infobank" with the password "lock" and limits the accessible informations to those recorded between 77.03 and 80.11.30.

1.2 The STOP command is used to stop the current search or to end the dialog.

<stop-command> ::= { stop / st } [qn = { new / <unsigned-number> }]

If qn is specified a new search is initiated with an automatically assigned (new) or a user-defined query number.

Ex.: stop qn = new

2. UTILITIES COMMANDS

They provide some commodities to the user.

2.1 The NEWS command gives latest information on the system

<news> ::= { nw / news }

2.2 The INFO command gives general information concerning Euronet, the cost of a search, the schedule time, the users, the current search status, etc.

<info-command> ::= { info / in } { euronet / cost / schedule / users / status }

Ex.: info schedule

The class of information may be extended, aswell as the meaning of each parameter.

2.3 The OWN command allows the use of user-defined commands.

<own-command> ::= { own / ow } ({ proc / com } ,<string>{,<parameter>}*)

<parameter> ::= { <mix-letter-digit> / <mix-letter-digit>=<string> }

A call to a user-procedure can be achieved using the proc option.
In this case, the string which follows specifies the name of the procedure called. Some parameters may be passed to this procedure either by position or by key.

Ex.: own (proc, statistic, result-file = sysprint, data-file = 16)

It generates a call to the procedure STATISTIC which needs a DATA- and a RESULT-FILE.

If the com option is used, the string specifies the name of a user-defined command. From now this command may be used like the other CCL commands. Like previously some parameters may be passed to achieve the insertion of the new command.

Ex.: own (com, find 2, dbn = test)

From now the command find 2 is available. The dbn parameter is supplied to generate the runtime routines for the specified data-base.

Note that in this case a library of syntactic and semantic description of the user commands must be provided to ensure a correct analysis and execution of them.

2.4 The HELP command helps the user to learn and understand the commands.

<help-command> ::= { help / h } [[{ base / bs / stop / st / own / ow / ⋮ }]]

Ex.: h

Lists the available commands.

Note that, unlike the other commands, the abbreviation for HELP is only a one character string. The reason for is that normally the HELP command is the only command a user should know when he enters a system for the first time. From then the abbreviation should be as easy to use and to remember as possible. Since most computer-based existing systems use h as a valid abbreviation for HELP we also made this choice.

2.5 The DEFINE command allows the user to modify default values for some parameters. It may also be used to define output specifications which can from then on be used in show and print command (see 4.1 and 4.4) The syntax is :

<define-command> ::= { define / df } <def-element>{;<def-element>}*

<def-element> ::= { f <unsigned-number>=<element>{,<element>}* / tl = <code-langue> / ac = <adt>{,<adt>}* / os = (<pos>) / thes = <string> }

$$\left[\begin{array}{l} \underline{\text{prompts}} = \left\{\begin{array}{l}\underline{\text{long}}\\ \underline{\text{short}}\end{array}\right\} \\ \underline{\text{hist}} = \left\{\begin{array}{l}\underline{\text{yes}}\\ \underline{\text{no}}\end{array}\right\} \\ \text{<def-prt-sp>}\end{array}\right]$$

<element> ::= <gen-field-label> $\left[\left\{\begin{array}{l}\text{/ string>[/<pos>]}\\ \text{// <pos>}\end{array}\right\}\right]$

<pos> ::= <unsigned-number>[:<unsigned-number>]

<adt> ::= $\left\{\begin{array}{l}\underline{\text{ca}}\\ \underline{\text{typ}}\\ \underline{\text{ann}}\\ \underline{\text{hist}}\end{array}\right\}$

<def-prt-sp> ::= $\left\{\begin{array}{l}\underline{\text{sort}} = \text{<gen-field-label>}\{,\text{<gen-field-label>}\}^{*}\\ \underline{\text{p}} = \text{<pr-dest>}\\ \underline{\text{l}} = \text{<string>}\end{array}\right\}$

<pr-dest> ::= $\left\{\begin{array}{l}\underline{\text{offline}}\\ \underline{\text{remote}}\\ \text{<string>}\end{array}\right\}$

Def-element of type "f" must be used to define or redefine an output format.

Ex.: df f 12 = ti/title/5, au // 10 : 2

Output resulting from using this format will be as follow :

```
      |  1  2  3  4  5  6  7  8  9  10
------+-------------------------------------------
1     |            TITLE :  DATA BASE MANAGEMENT
2     |
3     |                                LINCOLN HENRI
      |
```

The string specifies the title of the output field. The position of the output field is specified in term of column number or column number and line displacement.

The tl parameter specifies the thesaurus language.

The os parameter may be used to constraint the size of the output medium in term of line size or line size and number of lines printed at a time.

For video terminal the following command will be usefull :

define os = (80 : 20)

The thes parameter, if implemented, defines the thesaurus which must be used for the search data base.

The ai parameter is used to get additionnal information on the thesaurus used : ca = category number

typ = descripter type

ann = annotation

hist = history

prompts specifies the detail level. Short option should be used for experimented user; long is recommended for the other.

l parameter specifies the header which must be set on a printed output.

A printed output can be sent to any support by specifying the p parameter.

sort is provided to enable the specification of default fields for sorting the output records.

3. PROFILE FACILITIES

They allows to define profiles.

3.1 The SAVE command is used to save one or more commands of the current search.

<save-command> ::= { save / sv } [{ all / <unsigned-number>[to unsigned-number>] }]

Since a number is automatically associated with each command within a search, it is possible to define a profile by saving all or part of the current search.

Ex.: save 18 to 31

The commands 18 to 31 of the current search are saved.

If no parameter is specified, the last command will be considered.

Note : the defined profile is identified by the query number given by a base or stop command.

3.2 The DELETE command is similar to the SAVE command except that the meaning of all is to remove the entire profile.

<delete-command> ::= { delete / dl } [{ all / <unsigned-number>[to<unsigned-number>] }]

Ex.: delete

Will delete the last command introduced.

4.1 The SHOW command causes on-line output at the terminal during a search.

The syntax is :

<show-command>::= { show / sw } [<show-operand-list>]

<show-operand-list> ::= <show-operand>{;<show-operand>}*

<show-operand> ::= { s <unsigned-number> / r <unsigned-number>[to unsigned-number>] / f { <unsigned-number> / = <format-list> } }

<format-list> ::= <gen-field-label>{,<gen-field-label>}*

<gen-field-label> ::= { <field-label> / <ti-so-jt> / au / ab }

<field-label> ::= { ct / cc / pt / st / ut / la / nd / . . . }

<ti-so-jt> ::= { ti / so / jt }

If the s parameter is specified, the unsigned number defines the command concerned (since all commands are numbered, see save command). The r parameter specifies the set of hits that will be displayed.

The f parameter specifies a format defined indirectly by a format number or directly by a format list. The format list specifies the fields which must be displayed for each record.

A proposed list of general field labels is given in Appendix [1] . When using a format number, it specifies a user defined (see "define-command") or a system defined format. Five widely used predefined formats are proposed in Appendix [2].

Ex.: sw s = 01; r = 1 TØ 3; f 5

The 3 first records of search number 1 will be shaved using format 5.

show f = ti, au, ab, nd

Show will cause the display of records satisfying the last question (s and r parameters are not specified) restricted to the fields ti, au, ab, nd.

For practical reasons, it is recommended to display only a limited set of record at a time (for instance, not more than one screen display).

The more and back commands will be used to see forward or backward records.

4.2 The MORE command causes the display (from the current record) of further records belonging to the set of the last show.

<more-command> ::= { more / mr }

4.3 The BACK command causes the display (from the current record) of precedent records belonging to the set of the last show.

<back-command> ::= { back / bk }

4.4 The PRINT command serves for the routing of the output records to any medium other than the user terminal.

<print-command> ::= { print / pr } [<print-op>{;<print-op>}*]

<print-op> ::= { <show-operand> / <def-prt-sp> / hist = { yes / no } / ed = <date>to<date> }

The meaning of the parameter is the same as in the define and show command (the hist parameter specifies if the historic of the search which produces the output is to be printed or not)

Ex.: print S3; f 18; hist = yes

This causes the output of the records generated by search 3, using format 18, with the trace back of the search.

4.5 The DISPLAY command is used to obtain listings of alphabetically adjacent or logically related search terms.

<display-command> ::= { display / ds } { <field-label>=<search-term> / <ti-so-jt>=<search-term> / au = <au-string> } [;<addition-list>

<search-term >::= <$string> [/<code-list-string>{,<code-list-string>}*]

<code-list-string> ::= { <code-langue> / nt / bt / st / rt / ua / la }

<addition-list> ::= <adt>{,<adt>}*

Ex.: display ct = ELECTRIC $$$ / ENGL, BT; ann, his

The result may be

Effluent treatment by ion exchange E.B.C. Thesaurus 1976

BT : Chemical and physical effluent treatment ion exchange

Elastomer E.B.C. Thesaurus 1976

BT : Synthetic Resin

** Electrical Equipment E.B.C. Thesaurus 1976

BT : Electricity

Electrical Insulation E.B.C. Thesaurus 1976

BT : Insulation

Electrical Laboratory Equipment E.B.C. Thesaurus 1976

BT : Electrical Equipment

The term marked with " ** " is the first one satisfying or following the control term "ELECTRIC $$$" (where each $ replace one character). The thesaurus language concerned is english. For each term its broader terms are also displayed. The annotation "E.B.C Thesaurus" corresponds to the ANN parameter and the year 1976 to the HIS one.

5. QUERY FORMULATION (Find-command)

The general formulation of this command is

$$\left\{\begin{array}{l}\underline{\text{find}}\\ \underline{\text{fd}}\end{array}\right\}\left\{\begin{array}{l}\text{<boolean-expression>}\\ \underline{\text{qn}} = \text{<unsigned-number>}\end{array}\right\};\left[\left\{\begin{array}{l}\underline{\text{v}}\\ \underline{\text{nofind}}\\ \underline{\text{v}}\ ;\ \underline{\text{nofind}}\\ \underline{\text{nofind}}\ ;\ \underline{\text{v}}\end{array}\right\}\right]$$

A query is formulated by building a boolean expression based on search terms. It may also be formulated by specifying a query identifier (see Save command) The parameter nofind is used when one only wants to have the syntax checked, but no search executed.

The parameter v causes the query to be echoed and to be confirmed before the command is interpreted.

Ex_1: Find qn = 125; v

The system reaction should be something like :

"Your command was : Find qn = 125; OK ? (Y/N)"

If confirmed the command will be interpreted and executed.

Ex_2: Find qn = 125; nofind

Only the syntax of the query "125" is checked.

Note that when used with a boolean expression, nofind causes the syntax of this expression to be checked.

The detailed syntax for boolean expression is given in appendix 4. We describe it here in general terms only. A boolean expression is build using factor connected by logical operator (and, or, not). Parenthesis may be used to override the priority rules.

Ex_1: Find ed ⩾ 78.12 or ed = 76;

Find pd = (72, 73, 76) or pd = (78 to 80);

In thesecases the factors are date constraints (date, set of dates, range of dates).

Ex_2: Find ti] (EURONET-(COMMAND or LANGUAGE));

This time the factor is a text constraint : title must contain the word "EURONET" followed at any distance (-) by the words "COMMAND" or "LANGUAGE". It must be pointed out that this inclusion operator (]) is something new in information retrevial systems. It is introduced for semantical reasons because it is meaningless to allow text search commands on fields which are not text fields.

For semantical reason the search on abstracts is limited to the inclusion operator :

Text search may also use the following adjency operators

- A sequence of periods ($string_1$...$string_2$) means that $string_1$ must be followed by $string_2$ with a maximum distance of 3 words (in the specified field).
- A sequence of plus characters ($string_1$ ++ $string_2$) means that $string_1$ must be followed by $string_2$ with excactly 2 words between them (in the specified field).
- A sequence of dollars ($string_1$ $$$$ $string_2$) means that $string_1$ must be followed by $string_2$ with excactly 4 characters between them (in the specified field).

Ex_3: fd 1 and ct = (MANAGMENT and (LIBRARY or "LIBRARY Δ INSTITUTION"));

As shown by this example factors may also be an unsigned number associated to a precedent find command within the current search or a constraint on other fields (controlled terms, classification codes, languages ...).

Ex_4: fd au = (SMITH/STAN or LAVER);

The factor concerns here author fields (firstname or firstname and lastname may be specified).

Space, truncation and masking

The spaces introduced in a command are not significant. They are considered as delimiter when requested. However it is possible to introduce significant spaces in a string by enclosing it between quotes.

Truncation can be used anywhere except in text search. The truncation character is " $" and is considered as replacing one character. Left and right truncation are allowed.

Masking is also achieved using dollar characters. Note that the meaning of a range like $$ to $$$ is to consider the set of values between the first value of 2 characters to the last value of 3 characters.

CONCLUSION

The language proposed allows very large facilities and can be used mainly for any bibliographic data base. However unlike the proposal made by A.E. Negus syntactical restrictions are introduced to help to avoid meaningless commands and to achieve a reasonnable syntactic analyser (LALR (1)). It is currently beeing implemented at the Computer Science Laboratory of the Brussels Free University.

Appendix 1 : list of field labels

AB	abstract
AF	author affiliation
AU	author name
CC	classification code
CO	coden
CP	patent country
CR	chemical abstracts registry number
CS	corporate source
CT	controlled term (i.e. thesaurus term)(either PT or ST)
CY	publication country
DT	document type
ED	computer entry date
FT	free text term (i.e. candidate descriptor)
JA	journal abbreviation
JT	journal tittle
LA	language
NA	abstract number
NC	contract number
ND	document number
NP	patent number
NR	report number
PD	publication date
PT	controlled principal descriptor
PY	publication year
RA	referenced author
RF	reference
RI	referenced inventor
RJ	referenced journal
RP	referenced patent
SB	ISBN
SO	source
SS	ISSN
ST	controlled secondary descriptor
TI	tittle
WL	wiswener line notation

Appendix 2 : predefined formats

Field names	abbre-viations	0=default format	Display formats 1	2	3	4	5
abstract	AB		x				
author name	AU	x	x	x		x	
classification code	CC			x	x		
controlled principal descriptors	PT	x		x	x		
controlled secondary descriptors	ST	x		x	x		
document number	ND	x	x	x	x	x	x
document source	SO		x	x		x	
document title	TI	x	x	x	x	x	x
free terms (candidate descriptors)	FT				x		

APPENDIX 3

```
<master> ::= <stop-command> | <base-command> | <more-command> | <back-command> |
             <show-command> | <print-command> | <delete-command> | <save-command> |
             <help-command> | <info-command> | <news> | <define-command> |
             <display-command> | <own-command>

<info-command> ::=  <info><T-info>

<info> ::=   info | in

<T-info> ::=   euronet  | cost | schedule | users | status

<news> ::=   news | nw

<help-command> ::=  <help> | <help><command>

<help> ::=   help | h

<command> ::=  <find> | <base> | <stop> | <display> | <sav> | <show> | <print> | <def> |
               <delt> | <more-command> | <back-command> | <info> | <news> | <own>

<save-command> ::=  <sav>  | <sav><del-op>

<sav> ::=   save | sv

<delete-command> ::=  <delt> | <delt><del-op>

<delt> ::=   delete  | dl

<del-op> ::=   all | <unsigned-number> | <unsigned-number> to <unsigned-number>

<print-command> ::=  <print> | <print><print-op-list>

<print> ::=   print |pr

<print-op-list> ::=  <print-op> | <print-op> ; <print-op-list>

<print-op> ::=  <show-operand> | <def-sp> | <search-hist> | <date-s>

<label> ::=   l =<string>

<pr-dest> ::=  offline | remote | <string>

<sort-keys> ::=   sort =<sort-key-list>

<sort-key-list> ::=  <gen-field-label> | <gen-field-label> , <sort-key-list>

<search-hist> ::=   hist = yes | hist = no
```

```
<show-command> ::=   <show> | <show><show-operand-list>

<show> ::=   show | sw

<show-operand-list> ::=   <show-operand> | <show-operand> ; <show-operand-list>

<show-operand> ::=   <s> | <rec> | <format>

<s> ::=   s  <unsigned-number>

<Rec> ::=    r <unsigned-number> | r <unsigned-number> to <unsigned-number>

<format> ::= f <unsigned-number> |  f = <format-list>

<format-list> ::=  <gen-field-label> | <gen-field-label> , <format-list>

<gen-field-label> ::=  <field-label> |  <tisojt> | ab | au

<more-command> ::=   more | mr

<back-command> ::=   back | bk

<base-command> ::=  <base> | <base><string> | <base><string> ; <base-par-list>

<base>  ::=   base | bs

<base-par-list> ::=  <base-par> | <base-par> ; <base-par-list>

<base-par> ::=  <pw> | qn = <unsigned-number>| tl = <language-code> | <date-s>

<pw> ::=   pw  =  <string>

<date-s> ::=   ed = <date> to <date>

<date> ::=  <period> . <period> .  <period> | <period> . <period>

<period> ::=   <digit><digit>

<stop-command> ::=   <stop> | <stop>  qn = <val>

<stop> ::=   stop | st

<val> ::=   new  | <unsigned-number>

<find-command> ::=  <find><boolean-exp> ; | <find><boolean-exp> ; <postfixe> |
                    <find>  qn = <unsigned-number> ;| <find>  qn = <unsigned-number> ;
                    <postfixe>

<postfixe> ::=   v | nofind | v ; nofind | nofind ; v

<find> ::=    find | fd
```

```
<unsigned-number> ::=  <digit> | <digit><unsigned-number>

<own-command> ::=   <own> ( <parameter-code> , <string> , <parameter-list> ) |
                    <own> ( <parameter-code> , <string> )

<parameter-code> ::=    proc  | com

<own> ::=    own | ow

<parameter-list> ::=  <parameter> | <parameter> , <parameter-list>

<parameter> ::=  <mix-letter-digit> = <string>  | <mix-letter-digit>

<boolean-exp> ::=   <term>  | <term> or <boolean-exp>

<term> ::=   <factor> | <factor><binop><term>

<binop> ::=   and | not

<factor> ::= <prefix><simple-operand> | ( <boolean-exp> ) | <unsigned-number> |
            ab ] <ext-string-op> | <tisojt> =  <list-op> | <tisojt> ] <ext-string-op>|
            au = <au-simple-operand> | <date-label><relational-connector><date> |
           <date-label> = ( <date-choice> )

<prefix> ::=  <field-label><relational-connector>

<field-label> ::=  cc | ct | pt | st | ut | la | nd

<relational-connector> ::=    = | > | >= | < | <=

<date-label> ::=  ed | pd

<date-choice> ::=  <date> to <date> | <date> , <date-list>

<date-list> ::=  <date> | <date> , <date-list>

<simple-operand> ::= <genstring> | ( <string-list> )

<string-list> ::=  <genstring> , <list-only> | <string-exp>

<list-only> ::=  <genstring> | <genstring> , <list-only>

<genstring> ::=  <$string> | <$string> to <$string>

<$string> ::= <quoted-string> | <mix-letter-digit-$> |  <quoted-string><$string>|
              <mix-letter-digit-$><quoted-string> | <mix-letter-digit-$>
              <quoted-string><$string>

<mix-letter-digit-$>  ::=  <car-$> | <car-$> <mix-letter-digit-$>

<car-$> ::=   <letter-digit> |  $
```

<letter-digit> ::= <letter> | <digit>

<string-exp> ::= <stringterm> | <stringterm> or <string-exp>

<stringterm> ::= <stringfactor> | <stringfactor><binop><stringterm>

<stringfactor> ::= <$string>| (<string-exp>)

<tisojt> ::= ti | so | jt

<list-op> ::= <genstring> | (<genstring> , <list-only>)

<string> ::= <quoted-string> | <mix-letter-digit> | <quoted-string><string> |
<mix-letter-digit><quoted-string><string> | <mix-letter-digit>
<quoted-string>

<quoted-string> ::= " <mix-letter-digit-space> "

<mix-letter-digit-space> ::= <car> | <car><mix-letter-digit-space>

<car> ::= <letter-digit> | space

<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q |
r | s | t | u | v | w | x | y | z

<mix-letter-digit> ::= <letter-digit> | <letter-digit><mix-letter-digit>

<au-simple-operand> ::= <gen-au-string> | (<au-string-list>)

<au-string-list> ::= <gen-au-string> , <au-list-only> | <au-string-exp>

<au-list-only> ::= <gen-au-string> | <gen-au-string> , <au-list-only>

<gen-au-string> ::= <au-string> | <au-string> to <au-string>

<au-string-exp> ::= <au-string-term> | <au-string-term> or <au-string-exp>

<au-string-term> ::= <au-string-factor> | <au-string-factor><binop><au-string-te

<au-string-factor> ::= <au-string> | (<au-string-exp>)

<au-string> ::= <$string> | <$string> / <$string>

<ext-string-op> ::= (<ext-string-list>)

<ext-string-list> ::= <ext-string-exp> | <ext-string-exp> , <ext-string-list>

```
<ext-string-exp> ::=  <ext-st-term> | <ext-st-term> or <ext-string-exp>

<ext-st-term> ::=  <ext-st-factor> | <ext-st-factor><binop><ext-st-term>

<ext-st-factor> ::=   <ext-string> | <ext-string><adj-seq><ext-st-factor>

<ext-string> ::=  <string> | (<ext-string-exp>)

<adj-seq> ::=   -  | <point-seq> | <dol-seq> | <pl-seq>

<point-seq> ::=    . | .  <point-seq>

<dol-seq> ::=   $ | $  <dol-seq>

<pl-seq> ::=   +  |  +  <pl-seq>

<display-command> ::=  <display><d-prefix><search-term> | <display><d-prefix>
                       <search-term> ; <addition-list> | <display> au = <au-string> |
                       <display> au = <au-string> ; <addition-list>

<display> ::=   display | ds

<d-prefix> ::=  <field-label> = | <tisojt> =

<search-term> ::= <$string> | <$string> / <code-list>

<code-list> ::=  <code-l-st> | <code-l-st> , <code-list>

<code-l-st> ::=  <code-langue>  | nt | bt | st | rt | ua | la

<code-langue> ::=  alba | arab | bulg | chin | czec | danh | duth | engl | finn |
                   flem | fren | germ | grek | hebr | hung | ital | japn | lith |
                   norg | pers | polh | port | romn | russ | sloe | span | srcr |
                   swed | turk | ukrn | yugo

<addition-list> ::=  <addition> | <addition> , <addition-list>

<addition> ::=  ca | typ | ann | hist

<define-command> ::=  <def><def-list>

<def> ::=   define | df

<def-list> ::=  <def-elem> |  <def-elem> ; <def-list>

<def-elem> ::= <format-def> | <thesaurus-def> | ai = <addition-list> | <def-sp> |
               os = ( <pos> ) | <def-rl>  |thes = <string> | <search-hist>

<format-def> ::=    f  <unsigned-number> = <format-def-list>
```

<format-def-list> ::= <elem> | <elem> , <format-def-list>

<elem> ::= <gen-field-label> | <gen-field-label> / <string> | <gen-field-label>
<string> / <pos> | <gen-field-label> // <pos>

<pos> ::= <unsigned-number> : <unsigned-number> | <unsigned-number>

<thesaurus-def> ::= tl = <code-langue>

<def-sp> ::= <sort-keys> | p = <pr-dest> | <label>

<def-rl> ::= prompts = short | prompts = long

APPENDIX 4

<boolean-expression> ::= <term> { or <term> }*

<term> ::= <factor> { { and / not } <factor> }*

<factor> ::= { <field-label><relational-connector><simple-operand> / (<boolean-expression>) / <unsigned-number> / [{ <ti-so-jt.> / ab }] (<ext-string-list>) / <ti-so-jt> = <operand-list> / au = <au-simple-operand> / { ed / pd } <relational-connector><date> = ({<date-choice>}) }

<relational-connector> ::= { > / < / = / <= / >= }

<simple-operand> ::= { <general-string> / ({ <general-string>{,<general-string>}+ / <string-expression> }) }

<general-string> ::= <$string>[to <$string>]

<string-expression> ::= <string-term>{or <string-term>}*

<string-term> ::= <string-factor>{ { and / not } <string-factor> }*

<string-factor> ::= { <$string> / (<string-expression>) }

<ext-string-list> ::= <ext-string-expression>{,<ext-string-expression>}*

<ext-string-expression> ::= <ext-string-term>{ or<ext-string-term>}*

<ext-string-term> ::= <ext-string-factor> [[and / not] <ext-string-factor>]*

<ext-string-factor> ::= <ext-string>{<adj-sequence><ext-string>}*

<ext-string> ::= [<string> / (<ext-string-expression>)]

<adj-sequence> ::= [- / {.}+ / {$}+ / {+}+]

<operand-list> ::= [<general-string> / (<general-string>{,<general-string>}+)]

<date-choice> ::= <date> [to <date> / {,<date>}+]

<au-simple-operand> ::= [<general-au-string> / (<au-string-list>)]

<au-string-list> ::= [<general-au-string>{,<general-au-string>}+ / <au-string-expression>]

<general-au-string> ::= <au-string>[to<au-string>]

<au-string-expression> ::= <au-string-term>{or au-string-term>}*

<au-string-term> ::= <au-string-factor> [[and / not] <au-string-factor>]*

<au-string-factor> ::= [<au-string> / (<au-string-expression>)]

REFERENCES

[1] A Mini-Euronet standard command set at Université Libre de Bruxelles (U.L.B.). Georges Van Slype, Eric Laudet, Claude Machgeels, Yves Vander Auwera.

[2] Euronet Guide Line : Standard commands for retrieval systems. A.E. Negus INSPEC The Institution of Electrical Engineers. Savoy Place, London WC 2R OBL, England.

[3] Introduction to Automata Theory, Languages and Computation. J.E. Hopcroft, J.D. Ullman. Addison-Wesley.

[4] Syntax of NCCL
Luc Bernard, Yves Vander Auwera; technical report n°119, 1981, pp 11., Laboratoire d'Informatique Théorique, Université Libre de Bruxelles.