Subsets and modular features of standard APT

by CLARENCE G. FELDMANN

Massachusetts Institute of Technology*
Cambridge, Massachusetts

INTRODUCTION

The APT (Automatically Programmed Tool) N/C language was developed in 1956 at M.I.T.'s Servomechanisms Laboratory by D. T. Ross. Since that time, the language has been widely used at N/C installations in the United States for all categories of N/C programming. Since 1961, further APT System development has been directed by the Illinois Institute of Technology Research Institute under Dr. S. Hori. Use of the APT Language became so universally accepted that in 1963, the American Standards Association (now the United States of America Standards Institute) initiated an activity to generate a United States standard for the APT Language.

Since the original organizational meeting in 1963, the USASI X3.4.7 Subcommittee has been preparing a Standard for the APT N/C language. This activity early encountered a unique problem in that the APT language had been, since its first industrial use in about 1958, an evolutionary computer system, designed for any user who had the computing facilities to add useful modules onto the system. Although the APT language began as a problem-oriented language for three-dimensional milling, through the process of evolution, it soon added language for drafting, lofting analysis, turning, boring, flame-cutting, lathe control, and whatever other user's peripheral areas of interest came along.

One of the basic criteria agreed upon by the members of USASI X3.4.7 is that a standard language would be of little use if no user or computing equipment manufacturer could reasonably implement a processor for the language. On the other hand, if only a small “core” subset of APT were standardized, this would not be very useful to industry either.

APT subsets and modules

To resolve this problem, it was decided to standardize the entire language, and to also designate certain classes of language syntax rules in APT as belonging to a particular “subset” or “modular feature” of the language.

The terms “APT Subset” and “APT Modular Feature” take on very specific connotations, and must be carefully defined before proceeding. In general, the total APT Language is divided into five basic subsets, where each subset is designed for a specific level of N/C programming and hardware. The subsets are “nested” in that each subset wholly contains all statements in the next smaller subset. The five APT Subsets are pictured below.

FIGURE I—APT subsets

*The work reported in this document has been made possible through support and sponsorship extended to the M.I.T. Electronic Systems Laboratory by the Manufacturing Technology Laboratory, RTD, Wright-Patterson Air Force Base under Contract F33 615–67–C–1530, Project DSR 70429.

The work reported herein was supported (in part) by Project MAC, and M.I.T. Research Program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract N00r-4102(01).
The APT Subsets are designed as "core" languages, i.e., they contain just enough basic facilities to provide a usable language for the indicated level of N/C programming. The largest subset (Multi-Axis Contouring) therefore does not contain all APT statements. It was the vote of the USASI X3.4.7 subcommittee that it would be undesirable to specify large, unwieldy language subsets in which many of the statements are not used at most APT installations. Instead, the "core" subset philosophy was adopted, and the less-essential, special-purpose language "packages" which have been developed over the years by APT users were categorized into a set of "modular features" which can be selected by anyone having a particular need for that feature and "plugged in" to a core subset. There are at present 25 such modular features being considered by X3.4.7.

Another aspect of the Modular Feature approach is that each module has a "natural" subset below which its use is not feasible. For example, it does not make sense to plug in the "Fitted Surfaces" modular feature to a point-to-point subset. Each APT modular feature therefore has a "minimum subset" associated with it.

Relation to USASI X3.4 criteria

The USASI Subcommittee for "Common Programming Languages" (X3.4) is currently working on a document entitled "Procedures for the Standardization Process" which, among other things, contains a section on "Determination of Language Subsets." At this point, it is of interest to discuss their work in this area and show the relationship of the X3.4.7 language breakdown.

The X3.4 document discusses language subsets, but does not describe a "modular feature" setup as adopted by X3.4.7. In defining subsets, they state:

"It is assumed to be a basic requirement of any establishment of subsets that there be two clearly recognizable extreme cases: the maximal language, \( L \), and the minimum feasible subset, \( N \) (for nucleus). The nucleus language, \( N \), is the agreed upon minimum version of the language. At least \( N \) must be completely specified as a single standard language.

The maximal language, \( L \), includes all facilities incorporated as features in a single language. \( L \) need not be a standard language.

Whether or not \( L \) is a standard language, it is desirable that there be a complete specification for \( L \).

If there are, as well, additional subsets \( I_1, I_2, \ldots, I_m \) then it is required that (1) each \( I_j \) be a proper subset of \( L \) and (2) \( N \) be a proper subset of each \( I_j \).

In set-theoretic notation,

\[
L \supseteq I_1 \cup I_2 \cup \ldots \cup I_m
\]

(1)

\[
N \subseteq I_1 \cap I_2 \cap \ldots \cap I_m \cap L
\]

(2)

where " \( \cup \) " denotes the union of features and " \( \cap \) " denotes the intersection of features (selection of only those common to both the two operands)."

The APT subsets comply with these criteria, where \( N \) corresponds to the APT "Minimum System" subset and \( L \) corresponds to all statements in all subsets and modular features taken together. In fact, we have seen that the APT subsets satisfy the further criterion (not required by X3.4) of being nested. Stating this in an additional formula,

\[
I_1 \supseteq I_2 \supseteq I_3 \supseteq \cdots \supseteq I_m
\]

(3)

The USASI X3.4 document describes a module approach as an equivalent way of defining language subsets. However, this description does not fit the APT modular feature approach and the two should not be confused.

In an attempt to put the modular feature approach into set-theoretic notation, consider the modular feature element

\[
MF_{sm}
\]

where \( s \) is a minimum recommended subset number and \( m \) is a modular feature number. An APT Language \( A_{\ell} \) may thus be built from any combination of the form

\[
A_{\ell} = I_j
\]

(4)

or

\[
A_{\ell} = I_j \cup MF_{bm}
\]

(5)

or

\[
A_{\ell} = I_j \cup MF_{sm} \cup MF_{bm}
\]

(6)

etc.

for any subset number \( k \leq j \) and \( p \leq j \), where \( I_j \) represents APT Subset \( j \).

To complete the definition of APT modular features, the following additional relationship should be stated:

\[
I_j \cap MF_{bm} = \varnothing \text{ (the null set)}
\]

(7)

for all \( j, s, \) and \( m \).

In other words, no modular features have any statements in common with any of the subsets.
Subsets and Modular Features of Standard APT

With the above definitions in mind, we shall now describe the various subsets and modules of the APT Language breakdown. You will recall that there are five basic subsets:

1. Minimum
2. Advanced Point-to-Point
3. Minimum Contouring
4. Three-Dimensional Contouring
5. Multi-Axis Contouring

The modular features associated with each of these subsets is as follows:

Subset 1: (Minimum System)
1. Feedrate attached to motion commands.
2. Copy statements to duplicate sequences, with or without translation and/or rotation.
3. Synonyms for APT vocabulary words.

Subset 2: (Advanced Point-to-Point)
1. Matrix transformations and rotations
2. Macro facilities
3. Nested definition capability
4. Special printing, punching, and reading (I/O language).
5. Point patterns
6. Control of multiple postprocessing.
7. Program looping and computing (algebraic calculations, functions for square roots, etc.)
8. Advanced point-line-circle definitions.

Subset 3: (Minimum Contouring)
1. Conic section contouring, including “LOFT conics”
2. Surface “thickness” capability
3. Conditional cut termination program control (what to do next depending upon how the last cut sequence terminated)
4. Deletion of specified cutting sequences (turning off output for tool positioning).
5. Pocket cleanup language
6. Assignment and manipulation of symbolic names of surface variables.
7. Tabulated Cylinder Surfaces
8. Special “offset” startup language

Subset 4: (Three-Dimensional Contouring)
1. Generalized cutter capability
2. Additional three-dimensional surfaces
3. Additional vector definitions
4. Additional cutter-to-part control language
5. Fitted Surfaces
6. Individual surface tolerance on all surfaces.

There are no modular features having subset 5 as a recommended minimum set.

In addition, a set of post processor language modular features are being defined, but are not sufficiently finalized to be presented at this time. Also, this rather large number of modules is not necessarily the final breakdown in the standard, but is the status at the writing of this paper.

We will next go into more detail on each subset of the language. Finally, a unique computer system for defining and maintaining the standard will be described.

Minimum Subset

The basic criterion which was used when deciding how small to make the minimum subset was that the subset be the bare minimum which could still be considered usable. Quoting from a draft of the X3.4.7 working paper on subsets, “The minimum numerical control system contains no geometric definition capability. The motion commands are limited to those using position coordinates or delta motions.”

Specifically, the minimum subset contains the following APT Statements:

1. Cutter orientation statement:
   FROM /x coord, y coord, z coord
2. Motion command:
   GOTO / x coord, y coord, z coord
   GODLTA / delta x, delta y, delta z
3. Program bounds:
   PARTNO (identifying string of characters)
   FINI

In addition, the minimum system postprocessor commands are included.

Clearly, the minimum language as presented above is so minimal that, except for the postprocessing control functions, it represents little more than a technique for converting the tool positions into the input format required by a machine tool. It is debatable whether some additional features such as symbolic names on points should also be included, but it was decided that a user might very well wish to use this simple language, especially if he has only a very small computer at his disposal, or if he is supplied cutter orientation data from some non-APT computer analysis routine. The latter case occurs frequently with sub-contracted jobs to small shops.

Modular features which make useful additions to the minimum subset include the ability to add feedrate values to motion commands, to define synonyms for the APT basic vocabulary words, and to COPY a point a specified number of times with or without rotation and translation.
Advanced point-to-point subset

This subset contains the minimum system syntax plus the ability to process geometric definitions and assign symbolic names to definitions of lines, circles, and points. Since the subset is still a point-to-point language, the motion commands are limited to positioning commands. A limited “ZSURF” capability is included to permit setting the Z coordinate of intersection point definitions of point definitions containing only X and Y coordinates.

One of the outstanding features of the minimum subset is the extremely limited geometric capability which includes only points specified by coordinates and delta motions. The advanced point-to-point subset adds six line definitions, four circle definitions, and six point definitions, listed below:

Line definition

(1) Through two points described by coordinates.
(2) Through two points described by symblos.
(3) Through a point and tangent to a circle.
(4) Tangent to two circles.
(5) Through a point and making a specified angle with the positive X axis.
(6) Through a point and parallel to another line.

Circle Definitions

(1) By coordinates of center, and radius value
(2) By center point and radius value.
(3) By center point and line to which it is tangent.
(4) By a center point and circle to which it is tangent.

Point Definitions

(1) By x, y, and z coordinates.
(2) By x and y coordinates.
(3) By the intersection of two lines.
(4) By the intersection of a line and a circle.
(5) By the intersection of two circles.
(6) On the circumference of a circle, located by angular measurement from the positive X axis.

From examining the above set of definitions, we see that they afford a flexible basic set of definition formats for ruler-and-compass type constructions for point definitions. Clearly, many more APT definitions exist in the library and could have been included, but this set was chosen (based on part programming experience) as the most basic.

Modular features which could reasonably be added include matrix operations, a macro facility, nested definition formats, special input-output statements for printing and punching of both cutter paths and canonical geometric definition data, point pattern definition and motions, algebraic and geometric operations and functions (+, −, *, /, SQRT, SINF, etc.), and a part program looping facility. The extended set of geometric point, line, and circle definitions is also a natural module to add to this subset.

Minimum contouring subset

The Minimum Contouring Subset incorporates all features of smaller subsets and adds the ability to calculate cutter offset paths for line and circle, two-dimensional cuts. Since most common line and circle definitions were included in the previous, point-to-point subset, the most significant additional language added by this subset consists of the cutter definition statement and contour motion statements, along with a series of additional plane definitions.

Specifically, the following features are added:

(1) Establish part surface
(2) Cutter specification by diameter and end radius
(3) Specify cutter path inside and outside tolerance.
(4) Establish a sense of direction for future motions
(5) Motion commands to go right, left, forward, and back
(6) Startup procedure to place the tool relative to a new curve to be machined.
(7) Cutter position commands to orient the cutter to the left, right, or on the curve.
(8) Additional and plane definitions:
   (a) By coefficients of a plane equation.
   (b) By three points not in a straight line.
   (c) Through a point and parallel to a given plane.
   (d) Parallel to and offset from a given plane.

Reasonable modular features which could be added to the minimum contouring subset include definitions for the conic sections, and tabulated cylinders, surface “thickness” capability, conditional check surface control (motion depending upon which of several check surfaces are reached first), an automatic pocket cleanout feature, more generalized startup facilities, a “DNTCUT-CUT” feature for turning off cutter motion output during cutter positioning, and additional flexibility in obtaining and naming values in canonical geometric definitions.

Three-dimensional contouring subset

The Three-Dimensional Contouring Subset adds the vector definition language, three-dimensional cone and cylinder definition language, and a three-dimensional startup procedure. The motion commands re-
main the same except for the addition of GOUP and GODOWN.
Specifically, the following features are added:

(1) Motion commands GOUP and GODOWN

(2) Vector Definitions:
   (a) By x, y, and z components
   (b) By two points

(3) Cone Definition:
   (a) By coordinates of vertex, axis, vector components, and cosine of half angle.
   (b) By vertex point, axis vector, and cosine of half angle.

(4) Cylinder Definition:
   (a) By coordinates of point on axis, axis vector components, and radius.
   (b) By point on axis, axis vector, and radius.

(5) Three-Dimensional Startup Procedure GO/... 

   \{ TO  
   \{ ON \, S1, S2, ...
   \{ PAST

The modular features which are natural additions include a more general cutter definition format, additional vector and three-dimensional geometric definitions, independent tolerance controls on all surfaces, and additional control of the cutter-to-part-surface relationship.

**Multi-axis contouring subset**

The Multi-Axis Contouring Subset adds the ability to control the cutter axis. This may be done by explicitly specifying the axis vector values or by specifying that the axis is to remain normal to the Part Surface or Drive Surface.

**Summary**

In summary, we see that each subset represents a basic language for one level of numerical control programming. The smaller subsets are clearly processable on small computers, and additional modular features could easily be made available for special applications or to make a more flexible language. We have also seen that each modular feature has a "natural" subset level at which it may be applied. Clearly, the same features could also be applied to larger subsets, but application at a lower than the recommended level either makes no sense or destroys the natural subset hierarchy logic.

Lastly, it should be pointed out that a complete subset and modular feature breakdown has also been prepared for the Postprocessor language area. This breakdown adds as much again as what has been discussed above, and is not presented in this paper.

**Computer implementation**

All of the above discussion has been a quick, high-level look at the APT Standard subsets and modules to give the reader a feeling for the ground rules used in breaking down the language. We now shift to the problem of rigorously defining and maintaining the actual standard.

One commonly-used technique is to produce an elaborate document, carefully defining each syntax rule by means of an appropriate "meta-language" and then further elaborating meaning through use of semantics and examples. The document, therefore, is the physical representation of the standard.

The APT Standard is likewise defined via a list of syntax rules, semantics statements, and examples. However, the physical standard exists as a computer data deck rather than a printed document. There also exists a computer program to read this data deck and produce a syntax and semantics document which is very similar to other language standard documents.

The use of a data deck as the physical standard represents a significant step in rigorously specifying and updating the standard. Computer checking programs can be written to check syntax rules for completeness and for ambiguities. New meta-language prints can be produced automatically by the computer whenever changes or additions to the data deck are made.

More significant for the subject at hand, by including subset and modular feature data with each rule in the data deck, the complex job of producing and maintaining the subsets and modular features is greatly simplified. For example, a computer run might change a series of data cards and then produce a new print of selected subsets and modular features, as dictated by control options of the program. The resulting print represents the very latest version of the standard and shows any inconsistencies which might have been introduced by the changes.

**Subset and modular feature data**

The subset and modular feature data are punched into specific card columns reserved for this purpose. The data are associated only with the syntax rule data, and not with the data cards which define the vocabulary words or semantic statements. In this way, the subset and modular feature data need be punched only once, and the related basic word and semantic data are brought in through chaining of references, starting with the syntax rules. We shall see exactly how this
Data deck example

The APT Standard data deck is divided into decks for the basic words of the language, the semantics, and the syntax. Each deck is identified by a deck number punched in a column reserved for this purpose. The syntax deck contains a rule and subrule number for the rule being defined, and a series of positive integers, where integers less than a certain size represent syntax rule referents, and larger integers refer to basic words of the APT language.

For example, let us examine the syntax rule for a “point specification” as used in

\[ \text{GOTO} / \text{(point)} \]

or

\[ L = \text{LINE} / \text{(point)}, \text{(point)} \]

where the first statement causes the cutter to move to the point specified, and the second statement defines a line “L” which passes through the two points specified.

The syntax rule which defines a point specification (“(point)”, above) is given in the syntax data deck as

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Subrule</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>2</td>
<td>0202</td>
<td>2334</td>
</tr>
<tr>
<td>3</td>
<td>0204</td>
<td>2047</td>
</tr>
<tr>
<td></td>
<td>452</td>
<td>2041</td>
</tr>
</tbody>
</table>

By substituting the entry in the semantics deck for rule 52, replacing the large (2000 series) numbers by the actual basic words, and printing in a “prettier” form, the rule becomes

\[
\text{IS}(1) 42 \\
\text{OR}(2) (\text{POINT} / 452) \\
\text{OR}(3) (42 = \text{POINT} / 452)
\]

This is known as the “short form” produced by one of the available computer print routines.

By substituting the names of the rule referents 42 and 452 and using the accepted Backus-Normal form, the print becomes

\[
\langle \text{POINT} \rangle : := \\
\langle \text{IDENTIFIER} \rangle .
\]

Subset and module numbers

The data giving the subset and module numbers for each rule and subrule in the syntax deck are maintained as a separate card deck. Columns are reserved in this deck for the subset or modular feature number for each rule, as well as columns for the recommended subset, if the rule falls into the modular feature category. For example,

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Subrule</th>
<th>Modular Feature</th>
<th>Recommended Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>01000</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>01000</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1</td>
<td>01000</td>
</tr>
<tr>
<td>142</td>
<td>1</td>
<td>1</td>
<td>00111</td>
</tr>
</tbody>
</table>

where each number shown is punched in a specific card column. These selected rules shown above give a representative sampling of the subset and module deck. We see that rule 16(1) is part of a modular feature which has subset 2(01000) as a minimum recommended subset. The modular feature number is given by the column in which the “1” appears. Rule 18(1) is a member of Subset 2. The nested subset characteristic is evident from the fact that any member of a subset also has a “1” in every higher subset card column (e.g., 01111 for rule 18(1)).

CONCLUSION

In conclusion, it should be noted that the information contained in this paper represents a particular level of the work of USASI X3.4.7. By the time the reader sees this paper, it is likely that changes will have occurred, and the latest information may be obtained by contacting the committee.

It should also be made clear that all of the work on the APT Standard has been performed by representatives of the Numerical Control community, including the Aerospace Industry, computer manufacturers, machine tool manufacturers, and universities. This background has made for an APT Standard which is generally agreed upon by a representative group of users, and should therefore be readily acceptable.

REFERENCES

1. Automatic programming of numerically controlled machine tools
   A series of eleven Interim Progress Reports on the M.I.T. APT Project June 1 1956 through November 30 1959 totalling 695 pp

From the collection of the Computer History Museum (www.computerhistory.org)
Subsets and Modular Features of Standard APT

2 S H ORI
"Future of numerical control in industry"
IEEE Transactions on Industrial Electronics May 1963 pp 15-21

3 S A BROWN C E DRAYTON B MITTMAN
"A description of the APT language"
Communications of the ACM Vol 6 No 11 November 1963 pp 649-658

4 C E DRAYTON
"APT for continuous path N/C programming"
Technical Paper No 576 American Society of Tool and Manufacturing Engineers 1964

5 P A R AND A S H ORI R N L ITTLE
"APT—Present and future"

6 R CHANDLER
"Design for numerical control machinery"
Machine Design Magazine February 15 1968

7 Procedures for the standardization process
USASI X3.4 Subcommittee Common Programming Languages Document No X3.4/68-1 June 25 1968

8 T C HARRIS J ALBERT C G FELDMANN et al
"Report of the subsets and modular features subgroup"
USASI X3.4.7 Working Paper No 102 June 17 1967

9 C W W ILSON III
"Computer routines for editing numerical control meta-language"
June 12 1968 USASI X3.4.7. Working Paper No CWW/11