This document was produced by SDC in performance of contract AF 19(628)-5166 with the Electronic Systems Division, Air Force Systems Command, in performance of ARPA Order 773 for the Advanced Research Projects Agency Information Processing Techniques Office.

TECH MEMO a working paper

System Development Corporation/2500 Colorado Ave./Santa Monica, California 90406

TM 2710/260/00

AUTHOR J. A. Barnett
J. A. Barnett

TECHNICAL C. Weissman / J.I.S.
C. Weissman

RELEASE C. Weissman / J.I.S.
C. Weissman

for

J. I. Schwartz

DATE 6/29/66 PAGE 1 OF 6 PAGES

SIM, An *S-Expression* Pattern-Matching *Function*

ABSTRACT

This document describes SIM, a LISP 2 pattern-matching *function* that is used for analyzing *S-expressions*. A basic knowledge of LISP 2 Intermediate Language is assumed.

1. PURPOSE

SIM is a LISP 2 pattern-matching *function*[1] that is used for analyzing S-expressions.

2. DESCRIPTION

SIM has two arguments: a pattern written in a Backus Normal Form-like language for *S-expressions* and a sample. The sample is matched to the pattern using the rules given below. The value of SIM is *Boolean*. If the sample and the pattern match, the value is true; otherwise the value is false. No reconstruction capability is included as with other similar pattern matchers (e.g., CONVERT, FLIP, etc.).

3. USAGE

SIM was originally designed as an aid for implementing the LISP 2 compiler; however, uses have been found in several other areas of LISP programming. Therefore, SIM is included as a primitive in the LISP 2 system.

3.1 FUNCTION FORMAT

The SIM function declaration has the form[2]:

```
(FUNCTION((SIM . LISP)BOOLEAN)((PATTERN SYMBOL) (SAMPLE SYMBOL)))
```

where PATTERN is any legal *pat* (see Section 3.2) and SAMPLE is any *S-expression*.

---

[1] Nonprimitive syntactic entities are italicized. If compound words are used to denote the entities, the words are, in most cases, joined by italicized colons.

[2] This and all succeeding examples are written in LISP 2 Intermediate Language.

3.2 LEGAL PATTERNS

A legal pattern is either a simple pattern or is defined in terms of simple patterns. In the equation below, a legal pattern, *pat*, is defined in terms of simple patterns, *sp*, and other legal patterns. The following BNF-like equation defines the legal construction rules for patterns:

$$pat \equiv sp\,|\,\mathrm{CONS}(pat,pat)\,|\,\mathrm{APPEND}(pat,pat)$$

The following are legal simple patterns (*sp*):

A.	This *sp* matches any *atom*.
N.	This *sp* matches any *number*.
S.	This *sp* matches any *S-expression*.
L.	This *sp* matches any nonatomic *S-expression*.
ID.	This *sp* matches any *identifier*.
V.	This *sp* matches any *variable*, i.e., an ID., or (ID. . ID.)
NIL	This *sp* matches any NIL.
Other *atoms*	*Atoms* not in the above list are patterns that match samples with which they are EQUALN. (See descriptions of METALST in Section 3.2.1 for exceptions.)
(OR. pat_1 ---- pat_n)	See text below for description.
((C. i j *pat*))	See text below for description.
((ANY. *pat*. --- pat_n))	See text below for description.

The simple pattern,

(OR. pat_1 -- pat_n)

matches any sample which matches at least one pat_j where $1 \le j \le n$. For example, the value of

(SIM (QUOTE(OR. A B N.))X)

is true if X is A, B, or any *number* and false otherwise.

The simple pattern ((ANY. pat_1 --- pat_n)) matches any sample that matches at least one (pat_j) where $1 \le j \le n$. Further, if no match is found, the pattern behaves as if the *any:clause* had been deleted.

For example, the value of

(SIM (QUOTE((ANY. A B)))X)

is true if X is (A), (B), or NIL and false otherwise. (See example 4.6 for a restriction of the use of this option.)

The simple pattern

((C. i j *pat*))

matches anything of the form

$(a_1 \text{ --- } a_i)$, $(a_1 \text{ --- } a_i\ a_{i+1})$ --- $(a_1 \text{ --- } a_j)$

where $0 \le i \le j$, and each a_x matches *pat* where $1 \le x \le j$. If i is 0 and no match is found, SIM behaves as if (C. i j *pat*) were deleted from the pattern.

For example, the value of

(SIM (QUOTE((C. 2 3 A)))X)

is true if X is (A A) or (A A A) and false otherwise.

The value of

(SIM (QUOTE((C. 0 2 R)))X)

is true if X is NIL, (R), or (R R) and false otherwise.

The value of

(SIM (QUOTE((C. 1 2 (OR. A B))))X)

is true if X is (A), (B), (A A), (B A), (A B), or (B B) and false otherwise. In this case (OR. A B) is *pat* in ((C. i j *pat*)).

(See Example 7 in Section 4 for restrictions of the use of ((C. i j *pat*)) when concatenated with other patterns.)

3.2.1 Atomic Pattern Extensions

All nonidentity matching atomic patterns such as A., V., etc., are referenced through (METALST . MANIP), a POTENTIAL FLUID SYMBOL variable. METALST is constructed in the following format:

$$\text{METALST} = ((p_1 \cdot f_1) \text{ -- } (p_n \cdot f_n))$$

where the p_i's are *atoms*, the f_i's are (FORMAL BOOLEAN SYMBOL), $1 \le i \le n$. When an atomic pattern is encountered, it is searched for as a p_i, and, if it is found, the corresponding f_i is applied to the sample to obtain the value of SIM. Therefore, by redefining METALST, new options may be easily added to the pattern matches.

For example, consider the following sequence:

```
(SECTION (USER MANIP) SYMBOL)
(SET METALST
   (CONS
     (CONS (QUOTE ARRAY.)(FUNARG BOOLEAN (A) (ARRAYP A))) METALST))
```

Then the value of (SIM (QUOTE ARRAY.)X) is true if X is an array and false otherwise.

4. EXAMPLES

If X and Y are *pats* then Z = (CONS X Y) is also a *pat*. Further, if (SIM X A) is true and (SIM Y B) is true, then (SIM (CONS X Y)(CONS A B)) is true.

Example 1:

```
Let X = N.
    Y = L.
    A = 1.7
    B = (A B)
```

```
then (SIM X A) is true
     (SIM Y B) is true
```

therefore (SIM (CONS X Y)(CONS A B)) = (SIM (QUOTE (N. L.)))(QUOTE(1.7 (A B)))) is true.

Example 2:

```
Let X = Z
    Y = (OR. A B)
```

then (SIM(CONS X Y)S) ≡ (SIM(QUOTE(Z OR. A B))S) has value true if S is (Z . A) or (Z . B) and false otherwise.

Example 3:

Let P = (X Y Z)

then P is a legal *pat* because X, Y, Z and NIL are all legal *pats* and P = (X Y Z) = (CONS X (CONS Y (CONS Z NIL))).

(SIM (QUOTE(X (OR. A B)Z))S) is true if S is (X A Z) or (X B Z) and false otherwise.

Example 4:

(SIM (QUOTE(OR. A (C. 1 3 B)))S) is true if S is A or (C. 1 3 B) and false otherwise.

Note: ((C. i j *pat*)) is a *sp* but (C. i j *pat*) is not.

(SIM (QUOTE(OR. A((C. 1 3 B))))S) is true if S is A, (B), (B B), or (B B B) and false otherwise.

Caution is necessary when using various arrangements of *or:phrases, c:phrases* and *any:phrases*. In the example above, pat_2 in (OR. pat_1 -- pat_n) is specified first as (C. 1 3 B) and second as ((C. 1 3 B). These are two quite different things.

Example 5:

(A (ANY. X Y)B) is a legal *pat* constructed by (CONS A(APPEND((ANY. X Y)) (B))) and A, ((ANY. X Y)) and B are all legal *pats*. (Recall if X and Y are *pats*, then (APPEND X Y) is a *pat*.)

(SIM (QUOTE(A(ANY. X Y)B))S) is true if S is (A X B) or (A Y B). It is also true if S is (A B) because if no matches are found for an *any:clause* the match looked for is with NIL and recall (APPEND NIL S) = S.

Example 6:

(SIM (QUOTE((ANY. A.)B))S) has the same value as (SIM (QUOTE(A. B))S). The reason that (SIM (QUOTE((ANY. A.)B))(QUOTE (B)) is false is that B is an *atom* and the *any:phrase* matches it leaving nothing in the sample to match B in the *pat*. No attempt is made by SIM to "slide" the *pat* over the sample.

Example 7:

(SIM (QUOTE((C. 0 2 A.)B))S) has value true if S is (*atom atom* B) and is false otherwise. The reason that (SIM (QUOTE((C. 0 2 A.)B))(QUOTE(B))) and (SIM (QUOTE((C. 0 2 A.)B))(QUOTE(X B))) have value false is the *c:phrase* accepts B in the sample as a match for A. and leaves nothing in the sample to match B in the *pat*. Caution is necessary when using *c:phrases* because SIM makes no attempt to "slide" the *pat* over the sample.

Example 8:

The example below demonstrates an example using an extended METALST. The new simple *pat* ARITHP. matches any prefix conglomeration of the operators + - * /.

```
(SECTION (USER MANIP) SYMBOL)
(SET METALST (CONS (CONS (QUOTE ARITHP.) ARITHP) METALST))
(FUNCTION (ARITHP BOOLEAN)(X)
  (SIM (QUOTE
    (OR. V. ((OR. PLUS TIMES)(C. Ø 1000000 ARITHP.))
            ((OR. DIFFERENCE QUOTIENT IQUOTIENT)ARITHP. ARITHP.)
            (MINUS ARITHP.)))X))
```

The value 1000000 is an arbitrary limit. Note that ARITHP is involved with the recursion of SIM; that is, ARITHP and SIM use each other for the evaluation.

DISTRIBUTION LIST

B. Barancik	2105
J. Barnett	2025
E. Book	2332
R. Bosak	2013
J. Burger	9919
D. Drukey	2105
S. Feingold	9525
Donna Firth	2310
E. Jacobs	2344
S. Kameny (50)	2009
E. Myer	2227
M. Perstein	2332
V. Schorre	2330
J. Schwartz	2123
R. Simmons	9439
E. Stefferud	9734
A. Vorhaus	2213
C. Weissman (10)	2214
A. Irvine	9627
R. Long	9913
H. Howell	9912
M. Howard	2042
D. Perry	2042
R. Berman	4317
R. Wolfson	2368