AUTOMATIC COMPUTATION OF DATA SET DEFINITIONS*

John C. Reynolds

Applied Mathematics Division
Argonne National Laboratory

November 17, 1967

This paper, with appendix deleted, has been submitted to IFIP CONGRESS '68.

*Work performed under the auspices of the U.S. Atomic Energy Commission.

AUTOMATIC COMPUTATION OF DATA SET DEFINITIONS*

John C. Reynolds

ABSTRACT

Most programming systems which attempt to provide flexible and efficient data representations require the user to specify the range of variables, parameters, and functions by extensive and detailed data structure declarations. The purpose of this paper is to suggest that much of this declarative information is redundant, and can be inferred from the non-declarative portion of the program. Specifically, in the context of a restricted but non-trivial programming language, pure LISP, a method is given for constructing a data set description of the results of a function from a program for the function and a data set description of its arguments.

A data set description is considered to be a special case of a recursive set definition, which is a set of equations of the form

$$X_1 \leftarrow \xi_1(X_1 \ldots X_n) \ldots X_n \leftarrow \xi_n(X_1 \ldots X_n)$$

where the X_i are set variables, and the ξ_i are expressions composed of set constants, set variables, and set functions. The effect of the definition is to define the sequence of sets $\theta_1 \ldots \theta_n$ via $X_i^0 = \{\}$, $X_i^{t+1} = \xi_i(X_1^t \ldots X_n^t)$, $\theta_i = \bigcup_{t=0}^{\infty} X_i^t$.

In defining LISP data sets, the following set functions are used:

$$\text{cons}_*(X,Y) = \{z \mid z = \text{cons}(x,y) \text{ for some } x \in X,\ y \in Y\}$$

$$\text{car}_*(X) = \{z \mid z = \text{car}(x) \text{ for some } x \in X - \underline{\text{atom}}\}$$

$$\text{cdr}_*(X) = \{z \mid z = \text{cdr}(x) \text{ for some } x \in X - \underline{\text{atom}}\}$$

---

*Work performed under the auspices of the U.S. Atomic Energy Commission.

along with the union operation and the constants NIL and atom, denoting the singleton set containing the atom NIL and the set of all atoms, respectively.

A construction method is described which accepts the definition of a LISP function and a recursive definition of its set of arguments and produces a recursive definition of a set containing the results of the function. A reduction process is then described which eliminates the set functions car_* and cdr_*.

As an example, the construction is applied to the LISP function

$ss(x_1) = \underline{if}\ null(x_1)\ \underline{then}\ cons(NIL,NIL)\ \underline{else}\ ssl(car(x_1), ss(cdr(x_1)))$, where:

$ssl(x_2,x_3) = ss2(x_2,x_3,x_3)$

$ss2(x_4,x_5,x_6) = \underline{if}\ null(x_5)\ \underline{then}\ x_6\ \underline{else}\ cons(cons(x_4,car(x_5)),ss2(x_4,cdr(x_5),x_6))$

and the recursive definition of the set of arguments

$$X_1 \leftarrow NIL \cup cons_*(\underline{atom}, X_1).$$

After reduction, the result of the construction is

$$X_{20} \leftarrow cons_*(X_4,X_4) \cup cons_*(X_{16},X_{20})$$

where

$$X_4 \leftarrow NIL$$

$$X_{16} \leftarrow cons_*(X_{17},X_{18})$$

$$X_{17} \leftarrow \underline{atom}$$

$$X_{18} \leftarrow NIL \cup cons_*(X_{17},X_{18}),$$

which indicates that the result of ss must always be a non-empty list whose last element is NIL and whose preceding elements are non-empty lists of atoms.

Possible extensions of the method are discussed briefly.

AUTOMATIC COMPUTATION OF DATA SET DEFINITIONS*

John C. Reynolds

Introduction

Most programming systems which attempt to provide flexible and efficient data representations require the user to specify the range of variables, parameters, and functions by extensive and detailed data structure declarations.[1,2] The purpose of this paper is to suggest that much of this declarative information is redundant, and can be inferred from the non-declarative portion of the program. Specifically, in the context of a restricted but non-trivial programming language, pure LISP,[3] we give a method for constructing a data set description of the results of a function from a program for the function and a data set description of its arguments.

To begin, the concept of a data set description or definition must be formalized. Probably the most elegant formalization is that of McCarthy,[4] who considers a data set definition to be a set of recursive set equations using the operations of cartesian product and disjoint union. Our own approach differs from McCarthy's primarily in using conventional set-theoretic union instead of disjoint union.

Recursive Set Definitions

Thus a data set definition is considered to be a special case of a recursive set definition, which is a set of equations of the form

$$X_1 \leftarrow \xi_1(X_1 \ldots X_n) \ldots X_n \leftarrow \xi_n(X_1 \ldots X_n) \qquad (1)$$

where the X_i are set variables, and the ξ_i are expressions composed of set constants, set variables, and set functions. The set functions appearing in

*Work performed under the auspices of the U.S. Atomic Energy Commission.

the ξ_i are subject to various conditions, the most important of which is isotonicity: If $X_1 \subseteq Y_1 \ldots X_k \subseteq Y_k$, then $F(X_1 \ldots X_k) \subseteq F(Y_1 \ldots Y_k)$.

The effect of a recursive set definition is to define a sequence of sets $\theta_1 \ldots \theta_n$ as the limits of $X_1 \ldots X_n$ after repeated (parallel) execution of the set definition beginning with $X_1 = \ldots = X_n = \phi$ (where ϕ denotes the empty set). Specifically, let $X_i^o = \phi$, $X_i^{t+1} = \xi_i(X_1^t \ldots X_n^t)$, and $\theta_i = \bigcup_{t=o}^{\infty} X_i^t$. The requirement of isotonicity insures that $X_i^t \subseteq X_i^{t+1}$ for all t.

In general, recursive set definitions have a variety of applications; for example, a context-free grammar can be interpreted as a recursive definition of a set of strings over a finite alphabet using the functions of union and concatenation. But our immediate concern is the definition of LISP data sets, i.e., sets of S-expressions. (We assume a familiarity with the basic concepts of LISP, as given in reference 3.)

In writing expressions denoting sets of S-expressions, we will use LISP atoms as constants denoting singleton sets containing the mentioned atom, e.g., NIL will denote the set whose only member is NIL. We will also use the symbol atom to denote the set of all atoms. The primitive LISP functions cons, car, and cdr are generalized to define the set functions:

$$\begin{aligned} \text{cons}_*(X,Y) &= \{z \mid z = \text{cons}(x,y) \text{ for some } x \varepsilon X,\ y \varepsilon Y\} \\ \text{car}_*(X) &= \{z \mid z = \text{car}(x) \text{ for some } x \varepsilon X - \text{atom}\} \\ \text{cdr}_*(X) &= \{z \mid z = \text{cdr}(x) \text{ for some } x \varepsilon X - \text{atom}\} \end{aligned} \qquad (2)$$

Two other functions will be used: union and σ_*, the latter being defined by

$$\sigma_*(X,Y) = \underline{\text{if}}\ Y = \phi\ \underline{\text{then}}\ \phi\ \underline{\text{else}}\ X. \qquad (3)$$

(σ_* is the generalization of the LISP function $\sigma = \lambda(x,y)x$.) Each of the functions cons_*, car_*, cdr_*, and σ_* distributes with union on each of its

arguments, and gives the empty set if any argument is the empty set. Moreover, $cons_*$ and σ_* give the empty set if and only if some argument denotes the empty set.

Construction of a Recursive Set Definition from a LISP Function

We now show how to construct, from the definition of a LISP function and a recursive definition of the set of all arguments for the function, a recursive definition of the set of all results of the function. Obviously "includes" will in general be "properly includes," but in a somewhat heuristic sense, the construction will produce a data set definition which is a "good fit" to the results of the function.

The construction is best shown by an example. Thus consider the LISP function: $ss(x_1) = \underset{2}{\underline{\text{if}}}\ \text{null}(x_1)\ \underset{3}{\underline{\text{then}}}\ \underset{4}{\text{cons(NIL,NIL)}}\ \underset{5}{\underline{\text{else}}}\ \underset{6}{\text{ssl}}(\underset{7}{\text{car}}(x_1),\underset{8}{\text{ss}}(\underset{9}{\text{cdr}}(x_1)))$

where:

$$ssl(x_2,x_3) = \underset{10}{ss2}(x_2,x_3,\underset{11}{x_3}) \qquad (4)$$

$$ss2(x_4,x_5,x_6) = \underset{13}{\underline{\text{if}}}\ \text{null}(x_5)\ \underset{14}{\underline{\text{then}}}\ x_6\ \underset{15}{\underline{\text{else}}}\ \underset{16}{\text{cons}}(\underset{17}{\text{cons}}(x_4,\underset{18}{\text{car}}(x_5)),\underset{19}{\text{ss2}}(x_4,\underset{20}{\text{cdr}}(x_5),x_6))$$

and the recursive definition of the set of arguments of ss:

$$X_1 \leftarrow \text{NIL} \cup cons_*(\underline{\text{atom}}, X_1)\ , \qquad (5)$$

which defines the set of all lists of atoms. Actually ss accepts a list representing a set of atoms and produces a list of lists representing all subsets of the set of atoms. However, the construction does not use this information; in a sense, its purpose is to recover a portion of this information from the function definition itself. (Note that each equation in the function definition contains distinct variables; this is required by the construction method.)

We begin by labelling each distinct expression in the function definition, excluding the premises of conditional expressions and their subexpressions, with an integer from 2 to n; let e_i denote the expressions labelled with i. In the example, this labelling is given by the numerals printed above the function definition; each e_i begins immediately below the numeral i, and n = 21. Now suppose that the evaluation of $ss(x_1)$ for any value of x_1 in θ_1 takes place in discrete time steps t = 0,1,2,..., and let S_i^t be the set of all S-expressions which can appear as the value of e_i during such an evaluation at time t or earlier. We will construct a recursive set definition whose first equation is the given input set definition, and whose remaining equations have the form $X_i \leftarrow \xi_i(X_1 \ldots X_n)$, where ξ_i satisfies the condition that $S_i^{t+1} \subseteq \xi_i(\theta_1\ S_2^t \ldots S_n^t)$. It then follows (by induction on t, assuming $S_i^o = \phi$) that $S_i^t \subseteq \theta_i$ for all t. But eventually, any result of ss must appear as the value of e_2 (the body of the definition of ss) and thus belong to S_2^t for sufficiently large t. Thus the set θ_2 contains all possible results of ss.

For $2 \leq i \leq n$, the i^{th} equation is constructed as follows:

(a) If e_i is a quoted atom, then any value of e_i (trivially) belongs to the singleton set containing the atom. Thus let ξ_i be e_i. For example,

$$X_4 \leftarrow \text{NIL}. \tag{6}$$

(b) If e_i is the variable which appears as the k^{th} formal parameter of the function f, then any value of e_i must have been computed previously as the value of the k^{th} actual parameter in some call (function designator) of f, or, if f is the main function (ss), as an input argument to the evaluation. Thus if $e_{a_1} \ldots e_{a_q}$ are the k^{th} actual parameters in all calls of f in the function definition, then let ξ_i be $X_{a_1} \cup \ldots \cup X_{a_q} \cup$ (if f is the main function then X_1

else ϕ). For example:

$$\begin{aligned} X_7 &\leftarrow X_9 \cup X_1 & X_{17} &\leftarrow X_{11} \cup X_{17} \\ X_{11} &\leftarrow X_6 & X_{19} &\leftarrow X_{12} \cup X_{21} \\ X_{12} &\leftarrow X_8 & X_{14} &\leftarrow X_{12} \cup X_{14} \end{aligned} \tag{7}$$

(c) If e_i is cons(e_j, e_k), then any value of e_i must be the cons of a previously computed value of e_j with a previously computed value of e_k. Thus let ξ_i be $cons_*(X_j, X_k)$. Similarly, if e_i is car(e_j) or cdr(e_j), let ξ_i be $car_*(X_j)$ or $cdr_*(X_j)$. For example:

$$\begin{aligned} X_3 &\leftarrow cons_*(X_4, X_4) & X_{16} &\leftarrow cons_*(X_{17}, X_{18}) \\ X_6 &\leftarrow car_*(X_7) & X_{18} &\leftarrow car_*(X_{19}) \\ X_9 &\leftarrow cdr_*(X_7) & X_{21} &\leftarrow cdr_*(X_{19}) \\ X_{15} &\leftarrow cons_*(X_{16}, X_{20}) \end{aligned} \tag{8}$$

(d) If e_i is a conditional expression of the form if <any predicate> then e_j else e_k, then any value of e_i must be a previously computed value of e_j or e_k. Thus let ξ_i be $X_j \cup X_k$. For example

$$X_2 \leftarrow X_3 \cup X_5 \qquad X_{13} \leftarrow X_{14} \cup X_{15} \ . \tag{9}$$

(e) If e_i is a call (function designator) of a defined function f, and the body of the definition of f is e_j, then any value of e_i must be a previously computed value of e_j. Thus let ξ_i be X_j. For example:

$$\begin{aligned} X_5 &\leftarrow X_{10} & X_{10} &\leftarrow X_{13} \\ X_8 &\leftarrow X_2 & X_{20} &\leftarrow X_{13} \end{aligned} \tag{10}$$

Elimination of car_* and cdr_*

Although we have succeeded in deriving a recursive definition of a set containing the results of the function ss, the meaning of this definition is obscure, since it involves the "analytic" functions car_* and cdr_* as well as the "synthetic" function $cons_*$. Thus our next task is to develop a method for transforming such definitions to remove car_* and cdr_*.

This transformation will require use of the following identities:

$$car_*(cons_*(X,Y)) = \sigma_*(X,Y) \tag{11a}$$

$$cdr_*(cons_*(X,Y)) = \sigma_*(Y,X) \tag{11b}$$

$$car_*(\underline{atom}) = car_*(\langle\text{any quoted atom}\rangle) = \phi \tag{11c}$$

$$cdr_*(\underline{atom}) = cdr_*(\langle\text{any quoted atom}\rangle) = \phi \tag{11d}$$

$$car_*(\sigma_*(X,Y)) = \sigma_*(car_*(X),Y) \tag{11e}$$

$$cdr_*(\sigma_*(X,Y)) = \sigma_*(cdr_*(X),Y) \tag{11f}$$

$$\sigma_*(X,cons_*(Y,Z)) = \sigma_*(\sigma_*(X,Y),Z) \tag{11g}$$

$$\sigma_*(X,\underline{atom}) = \sigma_*(X,\langle\text{any quoted atom}\rangle) = X \tag{11h}$$

$$\sigma_*(X,\sigma_*(Y,Z)) = \sigma_*(\sigma_*(X,Y),Z) \tag{11i}$$

The first identity is a generalization of the LISP identity car(cons(x,y)) = x; the occurrence of $\sigma_*(X,Y)$ indicates that car(cons(x,y)) has no values if y has no values (i.e., is undefined) even though it is otherwise independent of y. The second identity is a similar generalization of cdr(cons(x,y)) = y. The remaining identities are obvious consequences of the definitions of the functions involved.

It is convenient to introduce the abbreviation $\sigma_*(X, Y_1 \ldots Y_k)$ for $\sigma_*(\ldots \sigma_*(X, Y_1) \ldots Y_k)$. Note that $\sigma_*(X, Y_1 \ldots Y_k)$ is independent of the order of $Y_1 \ldots Y_k$, and that $\sigma_*(X) \equiv X$. When $\mathcal{V}$ denotes a set of variables $\{Y_1 \ldots Y_k\}$, we use $\sigma_*(X,\mathcal{V})$ to denote $\sigma_*(X, Y_1 \ldots Y_k)$.

An analytic term is defined to be any expression of the form $\sigma_*(A'(X_i), \mathcal{V})$, where A' is a composition of zero or more car_*'s and cdr_*'s, $X_i \varepsilon \{X_1 \ldots X_n\}$, and $\mathcal{V} \subseteq \{X_1 \ldots X_n\}$. A synthetic term is defined to be any expression of the form $\sigma_*(S'(X_1 \ldots X_n), \mathcal{V})$, where $\mathcal{V} \subseteq \{X_1 \ldots X_n\}$ and S' is any expression composed of $cons_*$, (some of) the variables $X_1 \ldots X_n$, atom, and quoted atoms, except that S' must not be simply a variable.

Consider an expression of the form:

$$T = A(S_1(X_1 \ldots X_n) \ldots S_n(X_1 \ldots X_n)) , \tag{12}$$

where $A(X_1 \ldots X_n)$ is an analytic term, and each $S_i(X_1 \ldots X_n)$ is a synthetic term. T will have the form:

$$\sigma_*(A'(\sigma_*(S'_{i_o}, \mathcal{V}_{i_o})), \sigma_*(S'_{i_1}, \mathcal{V}_{i_1}), \ldots , \sigma_*(S'_{i_k}, \mathcal{V}_{i_k})) . \tag{13}$$

(The arguments $X_1 \ldots X_n$ of each S'_i have been omitted for brevity.) This can be transformed, by repeated application of (11e) and (11f), into

$$\sigma_*(\sigma_*(A'(S'_{i_o}) \mathcal{V}_{i_o}), \sigma_*(S'_{i_1}, \mathcal{V}_{i_1}), \ldots , \sigma_*(S'_{i_k}, \mathcal{V}_{i_k})) \tag{14}$$

and then, by identity (11i), into

$$\sigma_*(A'(S'_{i_o}), \mathcal{V}_{i_o}, S'_{i_1}, \mathcal{V}_{i_1}, \ldots , S'_{i_k}, \mathcal{V}_{i_k}) \tag{15}$$

and, by identities (11g) and (11h), into

$$\sigma_*(A'(S'_{i_o}), \mathcal{V}_{i_o} \cup \mathcal{W}_{i_1} \cup \mathcal{V}_{i_1} \cup \ldots \cup \mathcal{W}_{i_k} \cup \mathcal{V}_{i_k}) \tag{16}$$

where $\mathcal{W}_i$ is the set of variables occurring in S'_i. Then by applying (11a) through (11d), $A'(S'_{i_o})$ can be reduced until either (i) an application of (11c) or (11d) gives $A'(S'_{i_o}) = \phi$, and therefore $T = \phi$, or (ii) all appearances of $cons_*$ and of constants are removed, so that T becomes an analytic term, or (iii) all appearances of car_* and cdr_* are removed while one or more appearances of $cons_*$ or a constant remain, so that T becomes a synthetic term.

Now each equation of a recursive set definition derived from a LISP function can be written in the form

$$X_i \leftarrow S_i(X_1 \ldots X_n) \cup A_i(X_1 \ldots X_n) \tag{17}$$

where S_i is a union of synthetic terms and A_i is a union of analytic terms. The sets θ_i defined by these equations are preserved by transforming each equation into

$$X_i \leftarrow S_i(X_1 \ldots X_n) \cup A_i(X_1 \ldots X_n) \cup A_i(S_1(X_1 \ldots X_n) \ldots S_n(X_1 \ldots X_n)), \tag{18}$$

since the additional expressions only add elements to X_i^t which already occurred in $X_i^{t'}$ for some $t' > t$. (More rigorously, it can be shown by induction on t that each transformed X_i^t is a subset of the untransformed θ_i and vice versa.) Then, by distributing union upwards, each $A_i(S_1(X_1 \ldots X_n) \ldots S_n(X_1 \ldots X_n))$ can be transformed into a union of terms of the form (12), and each of these terms can either be transformed into ϕ, so that it vanishes from the union, or into a synthetic or analytic term. Thus (18) can be converted back to the same form (17) as the original equation.

Obviously the transformation from (17) to (18) and back to (17), which we call a reduction, can be repeated an arbitrary number of times. Thus, for example, a single reduction of the equations (5-10) derived in the preceding section gives the equations:

$$\begin{aligned} X_2 &\leftarrow X_3 \cup X_5 \cup \mathrm{cons}_*(X_4, X_4) \\ X_7 &\leftarrow X_9 \cup X_1 \cup \mathrm{NIL} \cup \mathrm{cons}_*(\underline{\mathrm{atom}}, X_1) \\ X_{13} &\leftarrow X_{14} \cup X_{15} \cup \mathrm{cons}_*(X_{16}, X_{20}) \end{aligned} \tag{19}$$

while leaving the remaining equations unchanged. Eight more reductions transform the equations into:

$$
\begin{aligned}
X_1 &\leftarrow \mathrm{NIL} \cup \mathrm{cons}_*(\underline{\mathrm{atom}}, X_1) \\
X_2 &\leftarrow X_3 \cup X_5 \cup \mathrm{cons}_*(X_4, X_4) \cup \mathrm{cons}_*(X_{16}, X_{20}) \\
X_3 &\leftarrow \mathrm{cons}_*(X_4, X_4) \\
X_4 &\leftarrow \mathrm{NIL} \\
X_5 &\leftarrow X_{10} \cup \mathrm{cons}_*(X_{16}, X_{20}) \cup \mathrm{cons}_*(X_4, X_4) \\
X_6 &\leftarrow \mathrm{car}_*(X_7) \cup \sigma_*(\underline{\mathrm{atom}}, X_1) \\
X_7 &\leftarrow X_9 \cup X_1 \cup \mathrm{NIL} \cup \mathrm{cons}_*(\underline{\mathrm{atom}}, X_1) \\
X_8 &\leftarrow X_2 \cup \mathrm{cons}_*(X_4, X_4) \cup \mathrm{cons}_*(X_{16}, X_{20}) \\
X_9 &\leftarrow \mathrm{cdr}_*(X_7) \cup X_1 \cup \mathrm{NIL} \cup \mathrm{cons}_*(\underline{\mathrm{atom}}, X_1) \\
X_{10} &\leftarrow X_{13} \cup \mathrm{cons}_*(X_{16}, X_{20}) \cup \mathrm{cons}_*(X_4, X_4) \\
X_{11} &\leftarrow X_6 \cup \sigma_*(\underline{\mathrm{atom}}, X_1) \\
X_{12} &\leftarrow X_8 \cup \mathrm{cons}_*(X_4, X_4) \cup \mathrm{cons}_*(X_{16}, X_{20}) \\
X_{13} &\leftarrow X_{14} \cup X_{15} \cup \mathrm{cons}_*(X_{16}, X_{20}) \cup \mathrm{cons}_*(X_4, X_4) \\
X_{14} &\leftarrow X_{12} \cup X_{14} \cup \mathrm{cons}_*(X_4, X_4) \cup \mathrm{cons}_*(X_{16}, X_{20}) \\
X_{15} &\leftarrow \mathrm{cons}_*(X_{16}, X_{20}) \\
X_{16} &\leftarrow \mathrm{cons}_*(X_{17}, X_{18}) \\
X_{17} &\leftarrow X_{11} \cup X_{17} \cup \sigma_*(\underline{\mathrm{atom}}, X_1) \\
X_{18} &\leftarrow \mathrm{car}_*(X_{19}) \cup \sigma_*(X_4, X_4) \cup \mathrm{NIL} \cup \sigma_*(X_{16}, X_{20}) \cup \sigma_*(\mathrm{cons}_*(X_{17}, X_{18}), X_{16}, X_{20}) \\
&\qquad \cup \sigma_*(\mathrm{cons}_*(X_{17}, X_{18}), X_4) \\
X_{19} &\leftarrow X_{12} \cup X_{21} \cup \mathrm{cons}_*(X_4, X_4) \cup \mathrm{cons}_*(X_{16}, X_{20}) \cup \mathrm{NIL} \\
X_{20} &\leftarrow X_{13} \cup \mathrm{cons}_*(X_{16}, X_{20}) \cup \mathrm{cons}_*(X_4, X_4) \\
X_{21} &\leftarrow \mathrm{cdr}_*(X_{19}) \cup \sigma_*(X_4, X_4) \cup \mathrm{NIL} \cup \sigma_*(X_{20}, X_{16}) \cup \sigma_*(\mathrm{cons}_*(X_{16}, X_{20}), X_{17}, X_{18}) \\
&\qquad \cup \sigma_*(\mathrm{cons}_*(X_4, X_4), X_{17}, X_{18})
\end{aligned}
\tag{20}
$$

At this point, however, the next reduction leaves the equations unchanged. (Actually two new terms $\sigma_*(\text{cons}_*(X_{16},X_{20}),X_{17},X_{18})$ and $\sigma_*(\text{cons}_*(X_4,X_4),X_{17},X_{18})$ are added to the equation for X_{19}, but these terms are subsumed by previous terms in the equation and can be dropped.) It can be shown that repetition of the reduction process will always converge in this manner; essentially one can define the "complexity" of synthetic and analytic terms in such a way that a reduction never introduces terms more complex than those already present, and there are only a finite number of possible distinct terms of a given complexity.

Once the equations are invariant under reduction, all analytic terms may be dropped without changing the set being defined. To prove this assertion, let $X_i^o = Y_i^o = \phi$, $X_i^{t+1} = S_i(X_1^t \ldots X_n^t) \cup A_i(X_1^t \ldots X_n^t)$, and $Y_i^{t+1} = S_i(Y_1^t \ldots Y_n^t)$, so that the full equations define the sets $\bigcup_{t=o}^{\infty} X_i^t$ while the equations without analytic terms define the sets $\bigcup_{t=o}^{\infty} Y_i^t$. We show that $X_i^t = Y_i^t$ by induction on t:

(a) When $t = 0$, $X_i^o = Y_i^o = \phi$.

(b) When $t = 1$, $A_i(\phi, \ldots, \phi) = \phi$, since constants cannot appear in analytic terms. Thus $X_i^1 = S_i(\phi, \ldots, \phi) = Y_i^1$.

(c) When $t \geq 2$, assuming $X_i^{t'} = Y_i^{t'}$ for $t' < t$, then $X_i^t = S_i(X_1^{t-1} \ldots X_n^{t-1}) \cup A_i(X_1^{t-1} \ldots X_n^{t-1}) = S_i(Y_1^{t-1} \ldots Y_n^{t-1}) \cup A_i(Y_1^{t-1} \ldots Y_n^{t-1}) = Y_i^t \cup A_i(S_1(Y_1^{t-2} \ldots Y_n^{t-2}) \ldots S_n(Y_1^{t-2} \ldots Y_n^{t-2}))$. The invariance of the equations under reduction shows that the last term must be a subset of $S_i(Y_1^{t-2} \ldots Y_n^{t-2}) \cup A_i(Y_1^{t-2} \ldots Y_n^{t-2}) = S_i(X_1^{t-2} \ldots X_n^{t-2}) \cup A_i(X_1^{t-2} \ldots X_n^{t-2}) = X_i^{t-1} = Y_i^{t-1}$. But we have seen that isotonicity insures that $Y_i^{t-1} \subseteq Y_i^t$. Thus $X_i^t = Y_i^t$.

When the analytic terms have been eliminated, each remaining synthetic term, since it contains only the functions $cons_*$ and σ_*, denotes the empty set if and only if some variable occurring in the term denotes the empty set. In this situation, it is possible to derive a solvable set of Boolean equations whose solution specifies whether each θ_i is empty or not. This solution can then be used to eliminate σ_* from the recursive definition.

In our example, the Boolean equations show that all sets are non-empty, so that all σ_*-expressions can be replaced by their first arguments. Thus we obtain:

$$X_1, X_7, X_9 \leftarrow NIL \cup cons_*(\underline{atom}, X_1)$$

$$X_2, X_5, X_8, X_{10}, X_{12}, X_{13}, X_{14}, X_{20} \leftarrow cons_*(X_4, X_4) \cup cons_*(X_{16}, X_{20})$$

$$X_3 \leftarrow cons_*(X_4, X_4)$$

$$X_4 \leftarrow NIL$$

$$X_6, X_{11}, X_{17} \leftarrow \underline{atom}$$

$$X_{15} \leftarrow cons_*(X_{16}, X_{20})$$

$$X_{16} \leftarrow cons_*(X_{17}, X_{18})$$

$$X_{18} \leftarrow NIL \cup cons_*(X_{17}, X_{18})$$

$$X_{19}, X_{21} \leftarrow NIL \cup cons_*(X_4, X_4) \cup cons_*(X_{16}, X_{20})$$

where equations with the same right side have been combined. Clearly, θ_{16} is the set of all non-empty lists of atoms and θ_2, which must include all results of ss, is the set of all non-empty lists whose last element is NIL and whose preceding elements are non-empty lists of atoms.

Possible Extensions

The method described here for computing data set definitions will achieve practical utility only with considerable extension, particularly of the programming language facilities which can be handled. The following is a list of such potential extensions, in order of increasing estimated difficulty: (1) It seems straightforward to extend the method to a language with a variety of cons-functions which create different kinds of list elements or "records," as well as a variety of car-functions for accessing the fields of these records. The relevance to languages such as ref. 1 is obvious. (2) The method should be extended to allow functional arguments and free parameters. (3) Some account should be taken of the premisses in conditional expressions. Premisses which are primitive predicates are probably tractable; recursively defined predicates may not be. (4) Integers and arrays should be treated. (5) Assignment statements and statement sequences with branches of control should be treated. (6) Assignment of list element fields (e.g., replaca and replacd in LISP) should be considered.

Another direction for possible extension lies in computing data set information beyond the usual what-can-point-to-what type of specification. Thus it might be possible to determine that a particular cons never creates a list element which will remain active beyond the exit of some function; this type of information is useful in allocating storage to avoid unnecessary garbage collection or storage fragmentation.[5] Or, when list elements are being created contiguously in a block of storage, it may be possible to determine that a particular cons always creates an element which is a fixed distance from its car or cdr, so that implicit pointers can be used.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. M. Donald MacLaren of Argonne National Laboratory for several helpful criticisms and for suggesting more concise proofs for some of the theorems.

REFERENCES

1. Wirth, N., and Hoare, C. A. R., "A Contribution to the Development of ALGOL," Comm. ACM, 9, p. 413 (June, 1966).
2. "IBM System/360 Operating System, PL/I Language Specifications," Form C28-6571-3, IBM Corporation (July, 1966).
3. McCarthy, J., "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I," Comm. ACM, 3, p. 184 (April, 1960).
4. McCarthy, J., "A Basis for a Mathematical Theory of Computation," Computer Programming and Formal Systems, Ed. P. Braffort and D. Hirschberg, North-Holland, Amsterdam (1963), pp. 33-70.
5. Wirth, N. Private communication.

APPENDIX

This appendix gives a more rigorous and general treatment of the material covered in the body of the paper, and gives proofs for a number of implicit and explicit assumptions made there. It is divided into two sections dealing with recursive set definitions in general, and with their specific application to the description of LISP data sets.

I. GENERAL THEORY OF RECURSIVE SET DEFINITIONS

Throughout this appendix U is used to denote a fixed denumerable set of objects (unspecified in Section I, but taken to be the set of all S-expressions in II), P to denote the set of all subsets of U, and Q to denote the set of all finite subsets of U. Several notational conventions are used in order to discuss n-tuples of objects and of sets of objects easily. If X denotes such an n-tuple, then X_i or $(X)_i$ will denote the i^{th} component of X. If F denotes a function whose result is an n-tuple, then $F_i \equiv \lambda(X)(F(X))_i$. The union and intersection of n-tuples of sets are defined by $(X \cup Y)_i \equiv X_i \cup Y_i$ and $(X \cap Y)_i \equiv X_i \cap Y_i$. Inclusion of one n-tuple of sets in another is defined by

$$X \subseteq Y \equiv \bigwedge_{i=1}^{n} (X_i \subseteq Y_i).$$

The symbol ϕ will denote any n-tuple whose components are all the empty set, as well as the empty set itself. Various set-theoretic identities which obviously extend to n-tuples will be assumed without proof.

Definitions

$$\mathcal{F}(m,n) \equiv \{F \mid F\colon P^m \to P^n \text{ and } F(X) = \bigcup_{Y \in Q^m} F(Y \cap X)\}; \quad \mathcal{F}(n) \equiv \mathcal{F}(n,n)$$

A function in $\mathcal{F}(m,n)$ is called finitely generated. It can be shown that union and intersection, identity, and various projection functions are finitely generated, and that compositions of finitely generated functions are finitely generated. (We use a notion of functional composition which allows functional expressions to have several arguments, implying that the (n-tuple) values of these arguments are to be concatenated.)

If $F \in \mathcal{F}(n)$, we define

$$\theta(F) \equiv \bigcup_{i=0}^{\infty} F^i(\phi).$$

Both the n-tuple $\theta(F)$ and its component sets are said to be recursively defined by F.

Theorem 1

If $X, Y \in P^m$, $F \in \mathcal{F}(m,n)$, and $X \subseteq Y$, then $F(X) \subseteq F(Y)$ (Isotonicity).

Proof: If $X \subseteq Y$, then $F(X) = \bigcup_{Z \in Q^m} F(Z \cap X) = \bigcup_{Z \in Q^m} F(Z \cap X \cap Y)$.

But each $Z \cap X$ is finite and therefore is a member of Q^m. Thus

$$\bigcup_{Z \in Q^m} F(Z \cap X \cap Y) \subseteq \bigcup_{Z \in Q^m} F(Z \cap Y) = F(Y).$$

The requirement of isotonicity is sufficient to prove many of the results given below, but the stronger requirement of finite generation is needed for some results, such as Theorem 2, and simplifies the proof of others.

Lemma 1

If $F \in \mathscr{F}(n)$, then, for all i, $F^i(\phi) \subseteq F^{i+1}(\phi)$.

Proof: By induction, since $F^0(\phi) = \phi \subseteq F^1(\phi)$ and $F^i(\phi) \subseteq F^{i+1}(\phi)$ implies $F^{i+1}(\phi) = F(F^i(\phi)) \subseteq F(F^{i+1}(\phi)) = F^{i+2}(\phi)$.

Lemma 2

If $F \in \mathscr{F}(m)$ and $G \in \mathscr{F}(m,n)$, then $G(\theta(F)) = \bigcup_{i=0}^{\infty} G(F^i(\phi))$.

Proof: By isotonicity we have $G(F^i(\phi)) \subseteq G(\theta(F))$ for each i, so that $\bigcup_{i=0}^{\infty} G(F^i(\phi)) \subseteq G(\theta(F))$. On the other hand,

$$G(\theta(F)) = \bigcup_{Y \in Q^m} G(Y \cap \theta(F)) = \bigcup_{Y \in Q^m} G(Y \cap \bigcup_{i=0}^{\infty} F^i(\phi)).$$

But since the $F^i(\phi)$ are a nested sequence and Y is finite, there exists an integer k(Y) such that $Y \cap \bigcup_{i=0}^{\infty} F^i(\phi) \subseteq F^{k(Y)}(\phi)$. Thus

$$G(\theta(F)) \subseteq \bigcup_{Y \in Q^m} G(F^{k(Y)}(\phi)) \subseteq \bigcup_{i=0}^{\infty} G(F^i(\phi)).$$

Theorem 2

If $F \in \mathscr{F}(n)$ then $F(\theta(F)) = \theta(F)$.

Proof: $F(\theta(F)) = \bigcup_{i=0}^{\infty} F(F^i(\phi)) = \bigcup_{i=0}^{\infty} F^i(\phi) = \theta(F)$.

Lemma 3

If $X \in P^n$, $F \in \mathscr{F}(n)$, and $F(X) \subseteq X$, then $\theta(F) \subseteq X$.

Proof: By induction $F^i(\phi) \subseteq X$, since $F^0(\phi) = \phi \subseteq X$ and $F^i(\phi) \subseteq X$ implies $F^{i+1}(\phi) = F(F^i(\phi)) \subseteq F(X) \subseteq X$.

Lemma 4

If $F, G \in \mathscr{F}(n)$ and, for all i, $F^{i+1}(\phi) \subseteq G(F^i(\phi))$, then $\theta(F) \subseteq \theta(G)$.

Proof: By induction $F^i(\phi) \subseteq G^i(\phi)$, since $F^0(\phi) = \phi = G^0(\phi)$ and $F^i(\phi) \subseteq G^i(\phi)$ implies $F^{i+1}(\phi) \subseteq G(F^i(\phi)) \subseteq G(G^i(\phi)) = G^{i+1}(\phi)$.

Theorem 3

If $F, G \in \mathscr{F}(n)$, $F(X) \subseteq G(X)$ for all $X \in P^n$, and $G(\theta(F)) = \theta(F)$, then $\theta(F) = \theta(G)$.

Proof: If $F(X) \subseteq G(X)$ then $F^{i+1}(\phi) = F(F^i(\phi)) \subseteq G(F^i(\phi))$, and by Lemma 4, $\theta(F) \subseteq \theta(G)$. On the other hand, $G(\theta(F)) \subseteq \theta(F)$, so by Lemma 3, $\theta(G) \subseteq \theta(F)$.

Theorem 4

If $F, G \in \mathscr{F}(n)$, $H = \lambda(X)(F(X) \cup G(X))$, $G(\phi) = \phi$, and $G(F(X)) \subseteq H(X)$ for all $X \in P^n$, then $\theta(F) = \theta(H)$.

Proof: Since $F^{i+1}(\phi) \subseteq H(F^i(\phi))$, Lemma 4 gives $\theta(F) \subseteq \theta(H)$. On the other hand, by induction $H(F^i(\phi)) \subseteq \theta(F)$, since $H(\phi) = F(\phi) \subseteq \theta(F)$ and $H(F^i(\phi)) \subseteq \theta(F)$ implies $H(F^{i+1}(\phi)) = F^{i+2}(\phi) \cup G(F^{i+1}(\phi)) \subseteq F^{i+2}(\phi) \cup H(F^i(\phi)) \subseteq \theta(F)$. Then by Lemma 2, $H(\theta(F)) \subseteq \theta(F)$, and by Lemma 3, $\theta(H) \subseteq \theta(F)$.

Theorem 5

If $H \in \mathscr{F}(2n,n)$, $F = \lambda(X)H(X,X)$ and either $G = \lambda(X)H(X,F(X))$ or $G = \lambda(X)H(X,\theta(F))$, then $\theta(F) = \theta(G)$.

Proof: $G(\theta(F)) = H(\theta(F),F(\theta(F)))$ or $H(\theta(F),\theta(F)) = H(\theta(F),\theta(F)) = F(\theta(F)) = \theta(F)$, so by Lemma 3, $\theta(G) \subseteq \theta(F)$. On the other hand, $F^{i+1}(\phi) = H(F^i(\phi), F^i(\phi)) \subseteq H(F^i(\phi), F^{i+1}(\phi))$ or $H(F^i(\phi), \theta(F)) = G(F^i(\phi))$, so by Lemma 4, $\theta(F) \subseteq \theta(G)$.

Lemma 5

If $F \in \mathscr{F}(n)$, $G \in \mathscr{F}(m,n)$, $H \in \mathscr{F}(m)$, $G(\phi) = \phi$, and $G \circ H = F \circ G$, then $G(\theta(H)) = \theta(F)$.

Proof: Using Lemma 2,

$$G(\theta(H)) = \bigcup_{i=0}^{\infty} G(H^i(\phi)) = \bigcup_{i=0}^{\infty} F^i(\phi) = \theta(F).$$

Theorem 6

If $D \in \mathscr{F}(n,m)$ $H \in \mathscr{F}(n+m,n)$, I is the identity function on P^{n+m}, π: $P^{n+m} \to P^n$ is the projection function which selects the first n components of an n+m-tuple, $F = \lambda(X)H(X,D(X))$, and $G = \lambda(X)I(H(X),D(\pi(X)))$, then $\theta(F) = \pi(\theta(G))$.

Proof: Let ρ: $P^{n+m} \to P^m$ be the projection function which selects the last m components of an n+m-tuple. Then $G = \lambda(X)I(H(\pi(X),\rho(X)),D(\pi(X)))$, and by Theorem 5, $\theta(G) = \theta(G')$, where $G' = \lambda(X)I(H(\pi(X),\rho(G(X))),D(\pi(X)))$.

Since the identity function I merely concatenates its arguments, $\rho(G(X)) = D(\pi(X))$, and thus $G' = \lambda(X)I(H(\pi(X),D(\pi(X))),D(\pi(X)))$. But $\pi(G'(X)) = H(\pi(X),D(\pi(X))) = F(\pi(X))$, i.e., $\pi \circ G' = F \circ \pi$. Thus since $\pi(\phi) = \phi$, Lemma 5 gives $\pi(\theta(G')) = \theta(F)$.

Theorem 7

If $D \in \mathscr{F}(m)$, $H \in \mathscr{F}(n+m,n)$, I is the identity function on P^{n+m}, π: $P^{n+m} \to P^n$ (ρ: $P^{n+m} \to P^m$) is the projection function which selects the first n (last m) components of an n+m-tuple, $F = \lambda(X)H(X,\theta(D))$, and $G = \lambda(X)I(H(X),D(\rho(X)))$, then $\theta(F) = \pi(\theta(G))$.

Proof: We have $G = \lambda(X)I(H(\pi(X),\rho(X)),D(\rho(X)))$. By Theorem 5, $\theta(G) = \theta(G')$, where $G' = \lambda(X)I(H(\pi(X),\rho(\theta(G))),D(\rho(X)))$.

Since the identity function merely concatenates its arguments, $\rho(G(X)) = D(\rho(X))$, i.e., $\rho \circ G = D \circ \rho$. Then since $\rho(\phi) = \phi$, Lemma 5 gives $\rho(\theta(G)) = \theta(D)$, and thus $G' = \lambda(X)I(H(\pi(X),\theta(D)),D(\rho(X)))$. But then $\pi(G'(X)) = H(\pi(X),\theta(D)) = F(\pi(X))$, i.e., $\pi \circ G' = F \circ \pi$. Thus since $\pi(\phi) = \phi$, Lemma 5 gives $\pi(\theta(G')) = \theta(F)$.

Transformations of Recursive Set Definitions

In the informal presentation in the body of this paper, a recursive set definition was described as a set of equations of the form

$$X_1 \leftarrow \xi_1(X_1 \ldots X_n) \ldots X_n \leftarrow \xi_n(X_1 \ldots X_n).$$

We now view these equations as defining a function $F \in \mathscr{F}(n)$ such that $F_i(X) = \xi_i(X_1 \ldots X_n)$; then F in turn defines the sets $(\theta(F))_1 \ldots (\theta(F))_n$. From this viewpoint, Theorems 3 to 7 justify transformations of recursive set equations which preserve the sets being defined. The consequences of Theorems 3 and 4, which are applicable when the ξ_i are unions of terms, will be examined in Section II. At this point we consider the remaining theorems.

Theorem 5 justifies the substitution of the defining equations into one another. In each of the ξ_i, some (or all) occurrences of the variable components X_j may be replaced by the corresponding ξ_j. For the defining equations have the form $X_i \leftarrow \xi_i'(X_1 \ldots X_n, X_1 \ldots X_n)$ where the first n arguments denote variable occurrences to be left unchanged while the second n arguments denote variable occurrences to be replaced. If $H \in \mathscr{F}(2n,n)$ is defined by $H_i(X,Y) = \xi_i'(X_1 \ldots X_n, Y_1 \ldots Y_n)$, then the original equations define the function $F = \lambda(X)H(X,X)$ while the transformed equations define the function $G = \lambda(X)H(X,F(X))$. By Theorem 5, $\theta(F) = \theta(G)$.

Theorem 6 justifies the introduction of auxiliary equations. Suppose that the subexpressions $\eta_1 \ldots \eta_m$ appear in the defining equations, i.e., that these equations have the form

$$X_1 \leftarrow \xi_1(X_1 \ldots X_n, \eta_1(X_1 \ldots X_n) \ldots \eta_m(X_1 \ldots X_n))$$
$$\vdots$$
$$X_n \leftarrow \xi_n(X_1 \ldots X_n, \eta_1(X_1 \ldots X_n) \ldots \eta_m(X_1 \ldots X_n))$$

Then these equations may be transformed into the n+m equations:

$$X_1 \leftarrow \xi_1(X_1 \ldots X_{n+m}) \qquad X_{n+1} \leftarrow \eta_1(X_1 \ldots X_n)$$
$$\vdots \qquad\qquad\qquad\qquad \vdots$$
$$X_n \leftarrow \xi_n(X_1 \ldots X_{n+m}) \qquad X_{n+m} \leftarrow \eta_m(X_1 \ldots X_n)$$

For let $H \in \mathcal{F}(n+m,n)$ be defined by $H_i(X) = \xi_i(X_1 \ldots X_{n+m})$ and $D \in \mathcal{F}(n,m)$ be defined by $D_i(X) = \eta_i(X_1 \ldots X_n)$. Then the original equations define the function $F = \lambda(X)H(X,D(X))$ while the transformed equations define the function $G = \lambda(X)I(H(X),D(\pi(X)))$. By Theorem 6, $\pi(\theta(G)) = \theta(F)$, i.e., $\theta(G)$ and $\theta(F)$ have the same first n components.

Theorem 7 deals with the case where an n-tuple of sets is defined by a set of equations

$$X_1 \leftarrow \xi_1(X_1 \ldots X_n\ T_1 \ldots T_m) \ldots X_n \leftarrow \xi_n(X_1 \ldots X_n\ T_1 \ldots T_m)$$

containing free parameters $T_1 \ldots T_m$ denoting m sets defined by a second set of equations:

$$Y_1 \leftarrow \eta_1(Y_1 \ldots Y_m) \ldots Y_m \leftarrow \eta_m(Y_1 \ldots Y_m)$$

Then the equations may be combined into the n+m equations

$$X_1 \leftarrow \xi_1(X_1 \ldots X_{n+m}) \qquad X_{n+1} \leftarrow \eta_1(X_{n+1} \ldots X_{n+m})$$
$$\vdots \qquad\qquad\qquad\qquad \vdots$$
$$X_n \leftarrow \xi_n(X_1 \ldots X_{n+m}) \qquad X_{n+m} \leftarrow \eta_m(X_{n+1} \ldots X_{n+m}).$$

For let $H \in \mathscr{F}(n+m,n)$ be defined by $H_i(X) = \xi_i(X_1 \ldots X_{n+m})$ and $D \in \mathscr{F}(m)$ be defined by $D_i(Y) = \eta_i(Y_1 \ldots Y_m)$. Then the original equations define the function $F = \lambda(X)H(X,\theta(D))$ while the transformed equations define the function $G = \lambda(X)I(H(X),D(\rho(X)))$. By Theorem 7, $\pi(\theta(G)) = \theta(F)$, i.e., $\theta(G)$ and $\theta(F)$ have the same first n components.

Definitions

An important class of finitely generated functions is obtained by extending functions of objects to functions of sets of objects in a standard manner.

If $f: D \to U$, where $D \subseteq U^n$, then $f_*: P^n \to P$ is defined by

$$f_*(X) = \{y \mid y = f(x) \text{ for some } x \in D \cap (X_1 \times \ldots \times X_n)\}$$

A function $F: P^n \to P$ is called partially pivotal if there exists a set of variable components $\mathscr{V} = \{X_{i_1} \ldots X_{i_k}\}$ such that $F(X) = \phi$ if and only if $X_{i_1} \cap \ldots \cap X_{i_k} = \phi$. The members of $\mathscr{V}$ are called the pivots of $F(X)$. If $\mathscr{V} = \{X_1 \ldots X_n\}$, then F is called pivotal.

Theorem 8

If $f: D \to U$, where $D \subseteq U^n$, then:

(a) For $X_1 \ldots X_n, Y_i \in P$,

$$f_*(X_1 \ldots X_{i-1}, X_i \cup Y_i, X_{i+1} \ldots X_n) = f_*(X_1 \ldots X_n) \cup f_*(X_1 \ldots X_{i-1}\ Y_i\ X_{i+1} \ldots X_n)$$

(b) $f_*(X) = \phi$ when $X_i = \phi$ for any i.

(c) $f_* \in \mathscr{F}(n,1)$

(d) If $D = U^n$, i.e., f is complete, then f_* is pivotal.

Proof: (a) Since $D \cap (X_1 \times \ldots \times X_{i-1} \times (X_i \cup Y_i) \times X_{i+1} \times \ldots \times X_n) = D \cap (X_1 \times \ldots \times X_n) \cup D \cap (X_1 \times \ldots \times X_{i-1} \times Y_i \times X_{i+1} \times \ldots \times X_n)$. (b) Since $D \cap (X_1 \times \ldots \times X_n) = \phi$ if any $X_i = \phi$. (c) Since (using (a))

$$f_*(X) = f_*(X_1 \ldots X_n) = f_*\left(\bigcup_{Y_1 \in Q} (Y_1 \cap X_1) \ldots \bigcup_{Y_n \in Q} (Y_n \cap X_n)\right)$$

$$= \bigcup_{Y_1 \in Q} \ldots \bigcup_{Y_n \in Q} f_*(Y_1 \cap X_1, \ldots, Y_n \cap X_n) = \bigcup_{Y \in Q^n} f_*(Y \cap X).$$

(d) Since $U \cap (X_1 \times \ldots \times X_n)$ is empty if and only if any $X_i = \phi$.

Theorem 9

If $F \in \mathscr{F}(n)$ is defined by equations of the form

$$F_i(X) = \bigcup_{j=1}^{K_i} \xi_{ij}(X_1 \ldots X_n)$$

where each ξ_{ij} is an expression composed of constants denoting non-empty sets, (some of) the variables $X_1 \ldots X_n$, and pivotal functions, then it is possible to compute whether or not each component of $\theta(F)$ is empty.

Proof: It can be shown (by induction on length or depth of expressions) that each ξ_{ij} defines a partially pivotal function whose pivots are the component variables which actually occur in ξ_{ij}. Let M_{ijk} and N_i^m be arrays of truth values defined by

$$M_{ijk} \equiv X_k \text{ appears in } \xi_{ij}, \quad \text{and} \quad N_i^m \equiv (F^m(\phi))_i \neq \phi.$$

Then

$$N_i^0 = \underline{\text{false}}, \quad \text{and} \quad N_i^{m+1} = \bigvee_{j=1}^{K_i} \bigwedge_{k=1}^{n} (\neg M_{ijk} \vee N_k^m).$$

Obviously N_i^m may be computed for successive values of m, and if an m_o is reached such that $N_i^{m_o+1} = N_i^{m_o}$ for all i, then N_i^m will remain constant for all $m \geq m_o$, and $N_i^{m_o}$ will describe the non-emptiness of $(\theta(F))_i$.

But since $F^{m+1}(\phi) \supseteq F^m(\phi)$, either $N_i^{m+1} = N_i^m$ or $N_i^{m+1} = \underline{\text{true}}$, so that each step from m to m+1 either leaves all the N_i^m unchanged or increases the number of N_i^m which are <u>true</u>. Thus the computation will terminate at some $m_o \leq n$.

<u>Definitions</u>

For any n, we define $\sigma: U^n \to U$ to be the function such that $\sigma(x) = x_1$. This in turn defines the pivotal function $\sigma_*: P^n \to P$ such that

$$\sigma_*(X) = \underline{\text{if}}\ X_2 \cap \ldots \cap X_n = \phi\ \underline{\text{then}}\ \phi\ \underline{\text{else}}\ X_1.$$

(The value of n will usually be evident from the context in which σ_* is used.) If $\mathcal{V}$ denotes a set of component variables $\{X_{i_1} \ldots X_{i_k}\}$ then we will write $\sigma_*(Y,\mathcal{V})$ to denote $\sigma_*(Y,X_{i_1} \ \ldots \ X_{i_k})$.

The following properties of σ_* are obvious:

(a) $\sigma_*(\sigma_*(Y,\mathcal{V}_1),\mathcal{V}_2) = \sigma_*(Y,\mathcal{V}_1 \cup \mathcal{V}_2)$.

(b) If $F: P \to P$ satisfies $F(\phi) = \phi$ then $\sigma_*(F(Y),\mathcal{V}) = F(\sigma_*(Y,\mathcal{V}))$.

(c) If $F_1 \ldots F_k: P^n \to P$ are partially pivotal functions then $\sigma_*(Y, F_1(X) \ldots F_k(X)) = \sigma_*(Y,\mathcal{V}_1 \cup \ldots \cup \mathcal{V}_k)$, where $\mathcal{V}_i$ is the set of pivots of $F_i(X)$.

<u>Theorem 10</u>

If $F \in \mathcal{F}(n)$ is defined by equations which satisfy the hypotheses of Theorem 9, then it is possible to derive equations which define a function $G \in \mathcal{F}(n)$ such that $\theta(G) = \theta(F)$, which also satisfy the hypotheses of Theorem 9, and which does not contain any occurrences of σ_*.

Proof: By applying property (c) we may transform the equations defining F so that all non-initial arguments of σ_* are component variables. The equations will then have the form

$$F_i(X) = \bigcup_{j=1}^{K_i} \xi_{ij}(X_1 \ldots X_n, X_1 \ldots X_n)$$

where the second n arguments denote occurrences of component variables as non-initial arguments of σ_* and the first n arguments denote the remaining occurrences. By Theorem 5, $\theta(F) = \theta(G)$ where G is defined by

$$G_i(X) = \bigcup_{j=1}^{K_i} \xi_{ij}(X_1 \ldots X_n, (\theta(F))_1 \ldots (\theta(F))_n).$$

By Theorem 9, the truth values $N_i \equiv (\theta(F))_i \neq \phi$ can be computed, and each occurrence of $(\theta(F))_i$, which will be a non-initial argument of σ_*, can then be replaced by U if N_i is *true* or ϕ if N_i is *false*. Then, since the ξ_{ij} are composed entirely of pivotal functions, each ξ_{ij} containing any occurrences of ϕ can be eliminated from the definition of G. Each remaining occurrence of σ_* will have the form $\sigma_*(\eta_m(X), U \ldots U)$, and can be replaced by $\eta_m(X)$.

II. APPLICATIONS TO LISP

We now take U to be the set of all S-expressions, and show how to construct, from the definition of a LISP function, equations which recursively define a set including all possible results of the LISP function. We begin by defining the concept of a LISP function definition precisely.

Definitions

Given a finite set $\mathcal{V}$ of variables and a finite set $\mathcal{F}$ of function names in which each member $f_i \in \mathcal{F}$ has a fixed degree d_i, a LISP expression in $\mathcal{V}$ and $\mathcal{F}$ is defined as follows: Any quoted atom is an expression. Any member of $\mathcal{V}$ is an expression. If α_1, α_2, ... are expressions, then $\text{cons}(\alpha_1\alpha_2)$, $\text{car}(\alpha_1)$, $\text{cdr}(\alpha_1)$, if $\text{atom}(\alpha_1)$ then α_2 else α_3, if $\text{null}(\alpha_1)$ then α_2 else α_3, if $\text{eq}(\alpha_1,\alpha_2)$ then α_3 else α_4, and, for each $f_i \in \mathcal{F}$, $f_i(\alpha_1 \ldots \alpha_{d_i})$ are all expressions. In each case α_i is called the i^{th} immediate subexpression of the expression being defined.

A LISP definition of the functions $\mathcal{F} = \{f_1 \ldots f_m\}$ is a sequence of equations of the form:

$$f_1(x_{11} \ldots x_{1d_1}) = \xi_1 \ldots f_m(x_{m1} \ldots x_{md_m}) = \xi_m$$

where each ξ_i is a LISP expression in $\mathcal{V}_i = \{x_{i1} \ldots x_{id_i}\}$ and $\mathcal{F}$. (Note that different variables are used in each equation.)

The semantics of a LISP definition are described in reference 3. Actually the class of definitions described here is more limited than in 3 (the use of free parameters and functional arguments is prohibited), but the basic mechanisms of recursive functions are included.

Construction

Given a LISP definition $\mathcal{D}$ of functions $\mathcal{F} = \{f_1 \ldots f_m\}$ and a d_1-tuple T of sets of S-expressions, let $\alpha_1, \ldots, \alpha_n$ be an enumeration of all expressions in $\mathcal{D}$, i.e., the ξ_i's and all their subexpressions, in which $\alpha_i = \xi_i$ for $1 \leq i \leq m$. Assuming that the evaluation of f_1 occurs in

discrete time steps t = 0, 1, 2, ... , let S_i^t be the set of all S-expressions which can occur, at time t or earlier, as the value of α_i during the evaluation of f_1 applied to any d_1-tuple of arguments $(a_1, \ldots, a_{d_1}) \varepsilon T$. (We assume $S_i^0 = \phi$.) We give the construction from $\mathscr{D}$ of equations which define a function $F: P^{n+d_1} \to P^n$ such that $S^{t+1} \subseteq F(S^t, T)$. For $1 \leq i \leq n$:

(a) If α_i is a quoted atom, let $F_i(S,T) = \alpha_i$ (where α_i becomes a constant denoting the singleton set whose member is the quoted atom).

(b) If α_i is the variable x_{jk}, then any value of α_i must have appeared earlier during the evaluation as the k^{th} argument in a call of f_j. Therefore it must be a previously computed value of the k^{th} immediate subexpression of some expression beginning with f_j, or, if j = 1, it may be a member of T_k. Thus if $\{\alpha_{a_1} \ldots \alpha_{a_q}\}$ is the set of all expressions which occur in $\mathscr{D}$ as the k^{th} immediate subexpression of any expression beginning with f_j, let $F_i(S,T) = S_{a_1} \cup \ldots \cup S_{a_q} \cup$ (*if* j = 1 *then* T_k *else* ϕ).

(c) If α_i is $\text{cons}(\alpha_j \alpha_k)$ (or $\text{car}(\alpha_j)$ or $\text{cdr}(\alpha_j)$), then any value of α_i must be the cons (or car or cdr) of previously computed values of α_j and α_k. Thus let $F_i(S,T) = \text{cons}_*(S_j\ S_k)$ (or $\text{car}_*(S_j)$ or $\text{cdr}_*(S_j)$).

(d) If α_i is *if* <any predicate> *then* α_j *else* α_k, then any value of α_i must be a previously computed value of α_j or α_k. Thus let $F_i(S,T) = S_j \cup S_k$.

(e) If α_i is an expression beginning with f_j, then any value of α_i must be a previously computed value of $\xi_j = \alpha_j$. Thus let $F_i(S,T) = S_j$.

Theorem 11

If $F: P^{n+d_1} \to P^n$ is the function defined by equations constructed from a LISP definition of $\{f_1 \ldots f_m\}$, then $(\theta(\lambda(X)F(X,T)))_1$ includes all

results obtained by evaluating f_1 applied to any d_1-tuple of arguments $(a_1 \ldots a_{d_1}) \in T$.

Proof: Let $G = \lambda(X)F(X,T)$. Then $S^0 = \phi$, and $S^t \subseteq \theta(G)$ implies $S^{t+1} \subseteq F(S^t,T) \subseteq F(\theta(G), T) = G(\theta(G)) = \theta(G)$, so that $S^t \subseteq \theta(G)$ for all t. But any result of f_1 must eventually appear as the value of ξ_1 and thus belong to S_1^t for sufficiently large t.

Thus our construction produces a recursive definition of a set including the results of f_1 in which the sets $T_1 \ldots T_{d_k}$ of the possible arguments of f_1 appear as free parameters. But if the sets $T_1 \ldots T_{d_1}$ can also be described by recursive definitions, then, as a consequence of Theorem 7, these definitions can be combined into a single parameterless definition.

The result of the construction is a set of equations containing expressions composed of constants, component variables, and the functions $cons_*$, car_*, cdr_*, and $\cup$. The constants may be quoted atoms; we also allow the constant atom denoting the set of all atoms. We now proceed to develop an algorithm for transforming this set of equations to eliminate car_* and cdr_*. The algorithm will achieve this goal at the expense of introducing the function σ_*, which can then be eliminated through Theorem 10.

Definitions

We define the following subsets of $\mathscr{F}(n,1)$:

$$\mathscr{T}(n) = \{F \in \mathscr{F}(n,1) \mid F(X) = X_i \text{ for some } 1 \leq i \leq n\}$$

$$\mathscr{S}(n,0) = \mathscr{T}(n) \cup \{F \in \mathscr{F}(n,1) \mid F(X) = \text{some quoted atom or } F(X) = \underline{\text{atom}}\}$$

$$\mathscr{S}(n,d+1) = \mathscr{S}(n,d) \cup \{F \in \mathscr{F}(n,1) \mid F(X) = cons_*(F'(X),F''(X)) \text{ for some } F', F'' \in \mathscr{S}(n,d)\}$$

$$\mathcal{A}(n,0) = \mathcal{I}(n)$$

$$\mathcal{A}(n,d+1) = \mathcal{A}(n,d) \cup \{F \in \mathcal{F}(n,1) \mid F(X) = car_*(F'(X)) \text{ or } F(X) = cdr_*(F'(X)) \text{ for some } F' \in \mathcal{A}(n,d)\}$$

Then if $\mathcal{G}$ denotes any subset of $\mathcal{F}(n,1)$, we define:

$$\Sigma(\mathcal{G}) = \{F \in \mathcal{F}(n,1) \mid F(X) = \sigma_*(F'(X), \mathcal{Y}) \text{ for some } \mathcal{Y} \subseteq \{X_1 \ldots X_n\} \text{ and } F' \in \mathcal{G}\}$$

$$\Omega(\mathcal{G},m) = \{F \in \mathcal{F}(n,m) \mid \text{for } 1 \le i \le n,\ F_i(X) = \bigcup_{j=1}^{K_i} F_{ij}(X) \text{ where } F_{ij} \in \mathcal{G}\}$$

Lemma 6

(a) If $G \in \mathcal{A}(1,d)$ and $G \notin \mathcal{A}(1,0)$, then $G(\phi) = G(\underline{atom}) = \phi$.

(b) If $F \in \mathcal{S}(n,d)$ then F is partially pivotal.

(c) $car_*(cons_*(X,Y)) = \sigma_*(X,Y)$, $cdr_*(cons_*(X,Y)) = \sigma_*(Y,X)$.

Proof: (a) Since G is a composition of one or more car_*'s and cdr_*'s, and the domain of car and of cdr is the set of non-atomic S-expressions.

(b) Since cons is a complete function, $cons_*$ is pivotal, and any expression composed of pivotal functions is partially pivotal. (c) Since cons is complete, $cons_*(X,Y) = \bigcup_{x \in X} \bigcup_{y \in Y} \{cons(x,y)\}$, and since $car(cons(x,y)) = x$,

$car_*(cons_*(X,Y)) = \bigcup_{x \in X} \bigcup_{y \in Y} \{x\} = \sigma_*(X,Y)$. The cdr case is similar.

Lemma 7

If $F \in \Sigma(\mathcal{S}(n,d_F))$, $G \in \mathcal{A}(1,d_G)$, and $H = G \circ F$, then either $H(X) = \phi$ for all $X \in P^n$, or $H \in \Sigma(\mathcal{S}(n,d_H) \cup \mathcal{A}(n,d_H))$ where $d_H = \max(d_F, d_G)$.

Proof: Let $d_G^{min} \le d_G$ be the least value of d such that $G \in \mathcal{A}(1,d)$. We prove the theorem by induction on d_G^{min}.

(a) If $d_G^{min} = 0$, then $G \in \mathcal{T}(1)$. But the only member of $\mathcal{T}(1)$ is the identity function on P. Thus $H = F \in \Sigma(\mathcal{S}(n,d_F)) \subseteq \Sigma(\mathcal{S}(n,D_H))$.

(b) If $d_G^{min} > 0$, assuming the theorem to be true for all $G \in \mathcal{A}(1,d_G^{min}-1)$: We have $F(X) = \sigma_*(F'(X),\mathcal{V})$ where $F' \in \mathcal{S}(n,d_F)$. (i) Suppose $F' \in \mathcal{S}(n,0)$. By Lemma 6a and property (b) of σ_*, $H(X) = G(\sigma_*(F'(X),\mathcal{V})) = \sigma_*(G(F'(X)),\mathcal{V})$. If $F'(X) = X_i$ then $G \circ F' = \lambda(X)G(X_i) \in \mathcal{A}(n,d_G^{min}) \subseteq \mathcal{A}(n,d_H)$, and $H \subseteq \Sigma(\mathcal{A}(n,d_H))$. If $F'(X)$ is <u>atom</u> or a quoted atom, then by Lemma 6a, $G(F'(X)) = H(X) = \phi$. (ii) If $F' \notin \mathcal{S}(n,0)$, then $F'(X) = cons(F_1(X),F_2(X))$ where $F_1,F_2 \in \mathcal{S}(n,d_F-1)$. Since $d_G^{min} > 0$, either $G(X) = G_1(car_*(X))$ or $G(X) = G_1(cdr_*(X))$ where $G_1 \in \mathcal{A}(1,d_G^{min}-1)$. Using Lemma 6 and the properties of σ_*, we get:

$$H(X) = G_1\left(\begin{matrix}car_*\\ or\\ cdr_*\end{matrix}(\sigma_*(cons_*(F_1(X),F_2(X)),\mathcal{V}))\right) = G_1(\sigma_*(\begin{matrix}car_*\\ or\\ cdr_*\end{matrix}(cons_*(F_1(X),F_2(X))),\mathcal{V}))$$

$$= G_1(\sigma_*(\sigma_*(F_{1\text{ or }2}(X), F_{2\text{ or }1}(X)),\mathcal{V})) = G_1(\sigma_*(F_{1\text{ or }2}(X),\mathcal{V} \cup \mathcal{W})),$$

where $\mathcal{W}$ is the set of component variables occurring in $F_{2\text{ or }1}(X)$. Since $G_1 \in \mathcal{A}(1,d_G^{min}-1)$ and $\sigma_*(F_{1\text{ or }2}(X),\mathcal{V} \cup \mathcal{W}) \in \Sigma(\mathcal{S}(n,d_F-1))$, the induction hypothesis gives either $H(X) = \phi$ or $H \in \Sigma(\mathcal{S}(n,\max(d_F-1,d_G^{min}-1)) \cup \mathcal{A}(n,\max(d_F-1,d_G^{min}-1))) \subseteq \Sigma(\mathcal{S}(n,d_H) \cup \mathcal{A}(n,d_H))$.

<u>Theorem 12</u>

If $F \in \Omega(\Sigma(\mathcal{S}(k,d_F)-\mathcal{T}(k)),m)$ and $G \in \Omega(\Sigma(\mathcal{A}(m,d_G)),n)$, then there exist functions $S_{F,G} \in \Omega(\Sigma(\mathcal{S}(k,d_H)-\mathcal{T}(k)),n)$ and $A_{F,G} \in \Omega(\Sigma(\mathcal{A}(k,d_H)),n)$ where $d_H = \max(d_F,d_G)$ such that $G(F(X)) = S_{F,G}(X) \cup A_{F,G}(X)$.

Proof: Let G' be any member of $\Sigma(\mathcal{A}(m,d_G))$. Since any function in $\mathcal{A}(m,d_G)$ will depend upon a single component variable, G'(X) can be written as

$$G'(X) = \sigma_*(G''(X_{p(o)}),X_{p(1)} \cdots X_{p(q)})$$

where $G''\varepsilon\mathcal{A}(1,d_G)$. Since G" is a composition of car_*'s and cdr_*'s, Theorem 8a gives $G''(X\cup Y) = G''(X)\cup G''(Y)$, so that

$$G'(F(X)) = G'(\bigcup_{j=1}^{K_1} F_{1j}(X) \cdots \bigcup_{j=1}^{K_m} F_{mj}(X))$$

$$= \sigma_*(\bigcup_{j=1}^{K_{p(0)}} G''(F_{p(0)j}(X)), \bigcup_{j=1}^{K_{p(1)}} F_{p(1)j}(X) \cdots \bigcup_{j=1}^{K_{p(q)}} F_{p(q)j}(X))$$

where each $F_{ij}\varepsilon\Sigma(\mathcal{S}(k,d_F)-\mathcal{T}(k)) \subseteq \Sigma(\mathcal{S}(k,d_F))$. Then Theorem 8a applied to σ_* gives:

$$G'(F(X)) = \bigcup_{j_o=1}^{K_{p(0)}} \cdots \bigcup_{j_q=1}^{K_{p(q)}} \sigma_*(G''(F_{p(0)j_o}(X)),F_{p(1)j_1}(X) \cdots F_{p(q)j_q}(X)).$$

Since the F_{ij} are partially pivotal, each term in this union can be written as $\sigma_*(G''(F_{p(0)j_o}(X)),\mathcal{V})$ where $\mathcal{V}$ is the set of all pivots of $F_{p(1)j_1}(X) \cdots F_{p(q)j_q}(X)$. By Lemma 7, either $G''(F_{p(0)j_o}(X)) = \phi$, or $G''(F_{p(0)j_o}(X)) = \sigma_*(H'(X),\mathcal{W})$ where $H'\varepsilon\mathcal{S}(k,d_H)\cup\mathcal{A}(k,d_H) = (\mathcal{S}(k,d_H)-\mathcal{T}(k))\cup\mathcal{A}(k,d_H)$ (since $\mathcal{T}(k)\subseteq\mathcal{A}(k,d_H)$). In the first case the entire term vanishes from the union, since $\sigma_*(\phi,\mathcal{V}) = \phi$. In the second case, since $\sigma_*(\sigma_*(H'(X),\mathcal{W}),\mathcal{V}) = \sigma_*(H'(X),\mathcal{W}\cup\mathcal{V})$, the term can be written as H(X), where $H\varepsilon\Sigma(\mathcal{S}(k,d_H)-\mathcal{T}(k))\cup\Sigma(\mathcal{A}(k,d_H))$, and the union has the form $G'(F(X)) = \bigcup_{j=1}^{L} H_j(X)$.

Now since $G\varepsilon\Omega(\Sigma(\mathcal{A}(m,d_G)), n)$, we have $G_i(F(X)) = \bigcup_{j=1}^{K_i} G_{ij}(F(X))$, where each $G_{ij}\varepsilon\Sigma(\mathcal{A}(m,d_G))$. Thus:

$$G_i(F(X)) = \bigcup_{j=1}^{K_i} \bigcup_{j'=1}^{L_{ij}} H_{ijj'}(X),$$

where for each term, either $H_{ijj'}\varepsilon\Sigma(\mathcal{S}(k,d_H)\text{-}\mathcal{T}(k))$ or $H_{ijj'}\varepsilon\Sigma(\mathcal{A}(k,d_H))$. By defining $S_{F,G,i}(X)$ as the union of the terms of the first type, and $A_{F,G,i}(X)$ as the union of the remaining terms, we obtain functions $S_{F,G}$ and $A_{F,G}$ satisfying the theorem.

It should be noted that the proofs of this theorem and Lemma 7 not only establish the existence of $S_{F,G}$ and $A_{F,G}$ but also define a method for constructing the defining equations of $S_{F,G}$ and $A_{F,G}$ from the defining equations of F and G.

Reduction Algorithm

Given (equations defining) the functions $F\varepsilon\Omega(\Sigma(\mathcal{S}(n,d) - \mathcal{T}(n)),n)$ and $G\varepsilon\Omega(\Sigma(\mathcal{A}(n,d)),n)$, the following algorithm produces (equations defining) the function $S\varepsilon\Omega(\Sigma(\mathcal{S}(n,d)\text{-}\mathcal{T}(n)),n)$ called the reduction of F and G.

(1) Set $F^{(0)} = F$, $G^{(0)} = G$, $i = 0$.

(2) Set $F^{(i+1)} = F^{(i)} \cup S_{F^{(i)}G^{(i)}}$ and $G^{(i+1)} = G^{(i)} \cup A_{F^{(i)}G^{(i)}}$

(3) If $F^{(i+1)} = F^{(i)}$ and $G^{(i+1)} = G^{(i)}$, then terminate with $S = F^{(i)}$, otherwise increase i by one and go to step 2.

(Here the union of functions $F \cup G$ is defined by $\lambda(X)(F(X) \cup G(X))$.)

Theorem 13

The reduction algorithm terminates and produces a result such that $\theta(S) = \theta(F \cup G)$.

Proof: (a) Theorem 12 implies that at all stages of the algorithm $F^{(i)}$, $S_{F^{(i)}G^{(i)}} \varepsilon \Omega(\Sigma(\mathcal{S}(n,d)-\mathcal{T}(n)),n)$ and $G^{(i)}$, $A_{F^{(i)}G^{(i)}} \varepsilon \Omega(\Sigma(\mathcal{A}(n,d)),n)$. Thus each $F^{(i+1)}$ (or $G^{(i+1)}$) is produced from $F^{(i)}$ (or $G^{(i)}$) by adding terms from $\Sigma(\mathcal{S}(n,d)-\mathcal{T}(n))$ (or $\Sigma(\mathcal{A}(n,d))$) to each of its components. But the original functions F and G must be defined by expressions constructed from a finite vocabulary, the algorithm never introduces any new vocabulary, and for a fixed finite vocabulary the total number of terms in $\Sigma(\mathcal{S}(n,d)-\mathcal{T}(n))$ and $\Sigma(\mathcal{A}(n,d))$ is finite. Thus eventually every term being added to a component of F or G will already be present in that component, and the algorithm will terminate.

(b) At each execution of step 2, $F^{(i+1)} \cup G^{(i+1)} = F^{(i)} \cup G^{(i)} \cup G^{(i)} \circ F^{(i)} \supseteq F^{(i)} \cup G^{(i)}$. If $P = \theta(F^{(i)} \cup G^{(i)})$, then $G^{(i)} \circ F^{(i)}(P) \subseteq G^{(i)} \circ (F^{(i)} \cup G^{(i)})(P) = G^{(i)}(P)$, so that $(F^{(i+1)} \cup G^{(i+1)})(P) = (F^{(i)} \cup G^{(i)})(P) = P$. By Theorem 3, $\theta(F^{(i+1)} \cup G^{(i+1)}) = \theta(F^{(i)} \cup G^{(i)})$, and by induction, $\theta(F^{(i)} \cup G^{(i)}) = \theta(F \cup G)$ for all i.

(c) When the algorithm terminates, $F^{(i+1)} \cup G^{(i+1)} = F^{(i)} \cup G^{(i)} \cup G^{(i)} \circ F^{(i)} = F^{(i)} \cup G^{(i)}$, so that $G^{(i)}(F^{(i)}(X)) \subseteq F^{(i)}(X) \cup G^{(i)}(X)$ for all $X \varepsilon P^n$. The definition of $\Omega(\Sigma(\mathcal{A}(n,d)),n)$ implies that $G^{(i)}(\phi) = \phi$. Thus by Theorem 4, $\theta(F^{(i)} \cup G^{(i)}) = \theta(F^{(i)}) = \theta(S)$.

In summary, by introducing auxiliary equations, any recursive definition involving $cons_*$, car_*, cdr_*, σ_*, and $\cup$ can be transformed into a definition of a function $F \cup G$, where F and G are acceptable to the reduction algorithm. The result of this algorithm meets the conditions of Theorem 10, which can then be used to eliminate σ_* and obtain a definition involving only $cons_*$ and $\cup$.