ITERATIONS, LOOPS, RECURSIONS, AND GENETIC PROGRAMMING PROBLEM SOLVER (GPPS)

ITERATION

- **Four parts**
  - **initialization branch**
  - **termination branch**
  - **body of iteration (work-performing branch)**
  - **update branch**
- **Difficulties**
  - **nested iterations**
  - **unsatisfiable termination predicates**
- **Rationing**

ITERATION

DIFFERENT APPROACHES

- **DU ("Do Until") operation (GP-1)**
  - **open-ended number of steps in an iteration**
  - **open-ended number of iterations**
- **Automatically defined iterations (ADIs) (GP-2). Also called "restricted iteration"**
  - **There is only has a iteration-performing branch (IPB)**
  - **Iteration is over a known, fixed set**
    - **protein or DNA sequence (of varying length)**
    - **time-series data**
    - **two-dimensional array of pixels**
- **Automatically defined loops (ADLs) (GP-3)**
  - **loop initialization branch**
  - **loop termination branch**
  - **loop body branch**
  - **loop update branch**

AUTOMATICALLY DEFINED ITERATIONS (ADIs)

- **Overall program consisting of an automatically defined function `ADF0`, an iteration-performing branch `IPB0`, and a result-producing branch `RPB0`.**
- **Iteration is over a known, fixed set**
  - **protein or DNA sequence (of varying length**
  - **time-series data**
  - **two-dimensional array of pixels**

TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

- **Goal is to classify a given protein segment as being a transmembrane domain or non-transmembrane area of the protein**
- **Generation 20 — Run 3 — Subset-creating version**
  - **in-sample correlation of 0.976**
  - **out-of-sample correlation of 0.968**
  - **out-of-sample error rate 1.6%**

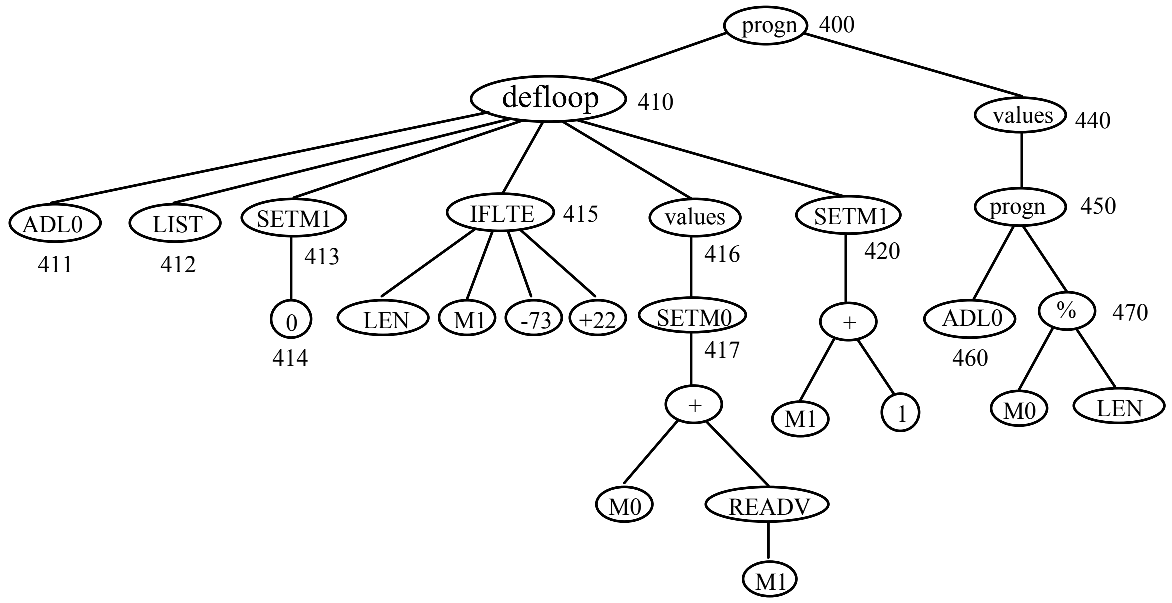
```
(progn (defun ADF0 ()

(values

(defun ADF1 ()

(ORN (ORN (ORN (I?) (H?)) (ORN (P?) (G?))) (ORN (ORN
(ORN (Y?) (N?)) (ORN (T?) (Q?))) (ORN (A?) (H?))))))

(defun ADF1 ()

(values (ORN (ORN (ORN (A?) (I?)) (ORN (L?) (W?)))
(ORN (ORN (T?) (L?)) (ORN (T?) (W?))))))

(defun ADF2 ()

(values (ORN (ORN (ORN (ORN (ORN (D?) (E?)) (ORN (ORN
(ORN (D?) (E?)) (ORN (ORN (T?) (W?)) (ORN (Q?)
(D?)))) (ORN (K?) (P?)))) (ORN (K?) (P?))) (ORN (T?)
(W?))) (ORN (ORN (E?) (A?)) (ORN (N?) (R?))))))

(progn (loop-over-residues
       (SETM0 (+ (- (ADF1) (ADF2)) (SETM3 M0))))

(values (% (% M3 M0) (% (% (% (- L -0.53) (* M0 M0))
(+ (% (% M3 M0) (% (+ M0 M3) (% M1 M2))) M2)) (% M3
M0))))))
```

COMPARISON OF EIGHT METHODS FOR SOLVING TRANSMEMBRANE SEGMENT IDENTIFICATION PROBLEM

Method	**Error**
von Heijne 1992	**2.8%**
Engelman, Steitz, and Goldman 1986	**2.7%**
Kyte and Doolittle 1982	**2.5%**
Weiss, Cohen, and Indurkhya 1993	**2.5%**
GP + Set-creating ADFs (GP-2)	1.6%
GP + Arithmetic-performing ADFs (GP-2)	1.6%
GP + ADFs + architecture-altering operations for subroutines (GP-3, chapter 16)	1.6%
GP + ADFs + architecture-altering operations for both subroutines and iterations (GP-3, chapter 17)	1.6%

AUTOMATICALLY DEFINED LOOPS

EXAMPLE OF A PROGRAM WITH A FOUR-BRANCH AUTOMATICALLY DEFINED LOOP (ADL0) AND A RESULT-PRODUCING BRANCH



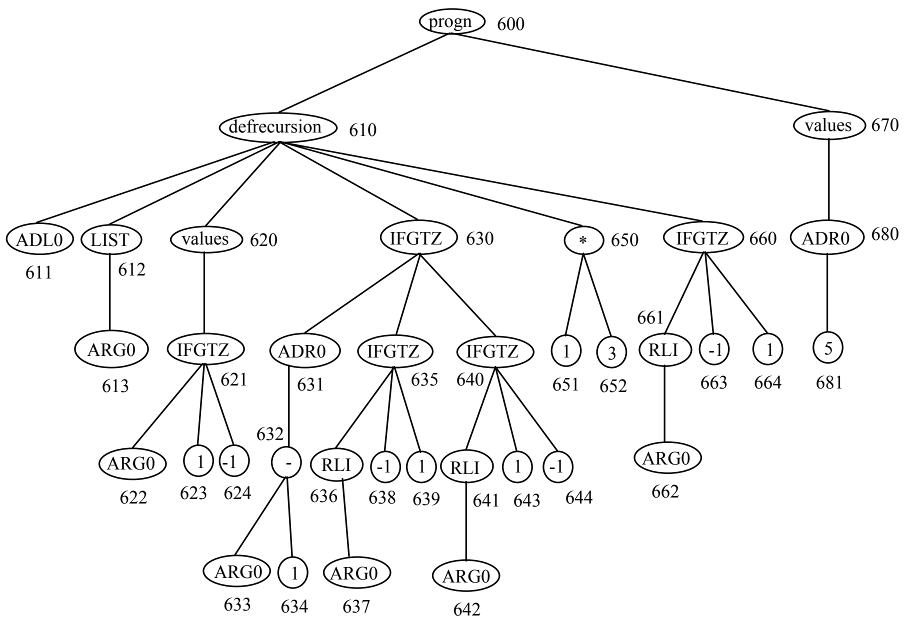
RECURSION

- **Recursion is difficult for GP because**
  - **Small changes in the STRUCTURE of a recursive program often lead to vastly different PERFORMANCE of the program and therefore very different values of fitness**
    - **Small changes in base case (initial value) as in logistic function**
    - **Small changes in transition rule**

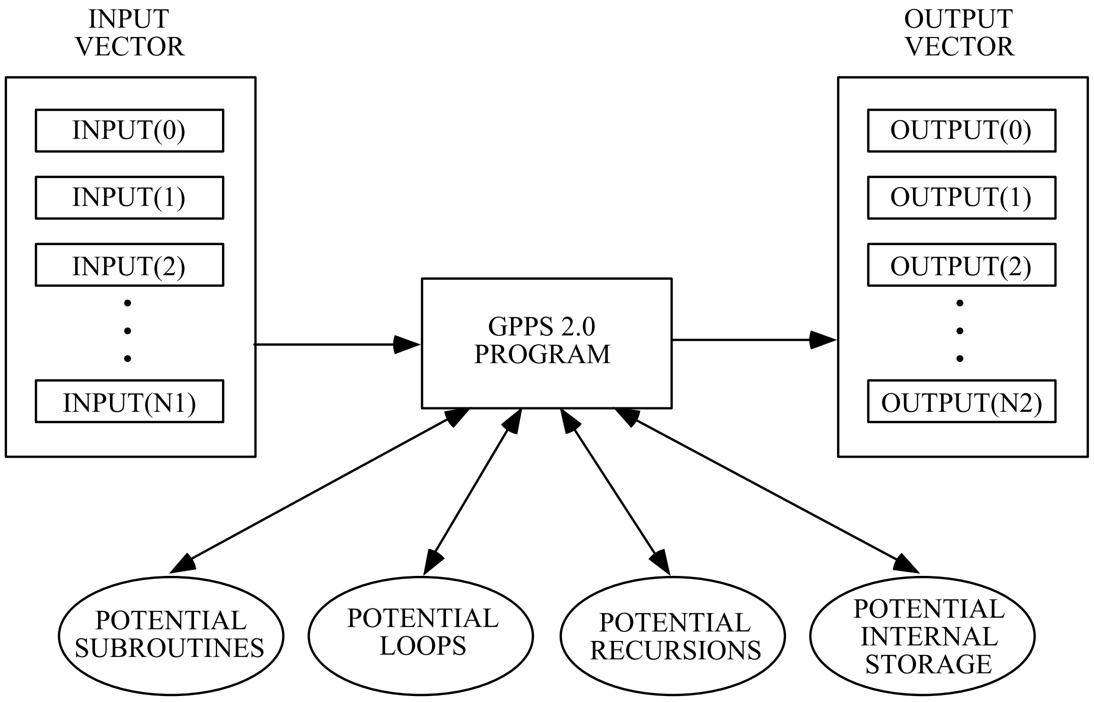
- **Practical problems of time (like iteration) because of potentially**
  - **Infinite recursion**
  - **Near-infinite recursion**

AUTOMATICALLY DEFINED RECURSION (ADR0) AND A RESULT-PRODUCING BRANCH

- **recursion condition branch, RCB**
- **recursion body branch, RBB**
- **recursion update branch, RUB**
- **recursion ground branch, RGB**



GENETIC PROGRAMMING PROBLEM SOLVER (GPPS) — VERSION 2.0



GENETIC PROGRAMMING PROBLEM SOLVER (GPPS 2.0)

Objective:	**Create, using the Genetic Programming Problem Solver 2.0, a computer program that takes the values of one independent floating-point variable *x* in the input vector and deposits a value into the output vector that closely matches the quadratic polynomial $2.718x^2 + 3.1416x$.**
Program architecture:	**One result-producing branch, RPB. Automatically defined loops, automatically defined recursions, automatically defined stores, and automatically defined function(s) and their arguments will be created during the run by the architecture-altering operations.**
Initial function set for the result-producing branches:	$F_{\text{rpb-initial}}$ = {**+, –, *, %, IFLTE, TOR, TAND, TNOT, RLI, WLO, FLOOR**}.

Initial terminal set for the result-producing branches:	**$T_{\text{rpb-initial}}$ = {$\leftarrow_{\text{bigger-reals}}$, `NINPUTS`, `NOUTPUTS`, `INDEX`}.**
Initial function set for the automatically defined functions:	**No automatically defined functions in generation 0. $F_{\text{adf-initial}} = \phi$.**
Initial terminal set for the automatically defined functions:	**No automatically defined functions in generation 0. $T_{\text{adf-initial}} = \phi$.**
Initial function set for automatically defined loops:	**No automatically defined loops in generation 0. $F_{\text{adl-initial}} = \phi$.**
Initial terminal set for automatically defined loops:	**No automatically defined loops in generation 0. $T_{\text{adl-initial}} = \phi$.**
Initial function set for automatically defined recursions:	**No automatically defined recursions in generation 0. $F_{\text{adr-initial}} = \phi$.**
Initial terminal set for automatically defined recursions:	**No automatically defined recursions in generation 0. $F_{\text{adr-initial}} = \phi$.**
Potential function set for the result-producing branches:	**$F_{\text{rpb-potential}}$ = {`ADL0`, `ADR0`, `SWB0`, `SWB1`, `ADF0`, `ADF1`, `ADF2`, `ADF3`}.**
Potential terminal set for the result-producing branches:	**$T_{\text{rpb-potential}}$ = {`LBB0`, `SRB0`, `SRB1`}.**
Potential function set for the automatically defined functions:	**$F_{\text{adf-potential}}$ = {`ADF0`, `ADF1`, `ADF2`, `ADF3`}.**

Potential terminal set for the automatically defined functions:	**$T_{adf-potential}$ = {`ARG0`, `ARG1`, `NINPUTS`, `NOUTPUTS`, `INDEX`, ←bigger-reals}.**
Potential function set for automatically defined loops:	**$F_{adl-potential}$ = {`ADL0`, `ADL1`, `ADL2`, `ADL3`}**
Potential terminal set for automatically defined loops:	**$T_{adl-potential}$ = {`NINPUTS`, `NOUTPUTS`, `INDEX`, ←bigger-reals}.**
Potential function set for automatically defined recursions:	**$F_{adr-potential}$ = {`ADR0`, `ADR0`, `ADR1`, `ADR2`, `ADR3`}.**
Potential terminal set for automatically defined recursions:	**$T_{adr-potential}$ = {`NINPUTS`, `NOUTPUTS`, `INDEX`, ←bigger-reals}.**
Fitness cases:	**20 randomly chosen values of the independent variable *x* between –1.0 and +1.0.**
Raw fitness:	**Raw fitness is the sum, over the 20 fitness cases, of the absolute value of the difference between the value deposited in the output vector and the value of the quadratic polynomial $2.718x^2 + 3.1416x$.**
Standardized fitness:	**Same as raw fitness**

Hits:	**The number of fitness cases (0 to 20) for which the value deposited in the output vector is within 0.01 of the value of the quadratic polynomial 2.718x^2 + 3.1416x.**
Wrapper:	**None.**
Parameters:	**M = 120,000. G = 1,001. `NINPUTS` = 1. `NOUTPUTS` = 1. Q = 2,000. D = 60. B = 2%. N_{rpb} = 1. S_{rpb} = 500. S_{adf} = 100. $N_{\text{max-adf}}$ = 4. $N_{\text{max-argument-adf}}$ = 2. $N_{\text{min-adf-arg}}$ = 0. $N_{\text{max-adl}}$ = 1. S_{adl} = 100. $N_{\text{max-argument-adl}}$ = 0. $N_{\text{min-argument-adl}}$ = 0. $N_{\text{max-adl-executions}}$ = 3. $N_{\text{max-adr}}$ = 1. S_{adr} = 100. $N_{\text{max-argument-adr}}$ = 0. $N_{\text{min-argument-adr}}$ = 0. $N_{\text{max-adr-executions}}$ = 9. $N_{\text{max-ads}}$ = 2.**
Result designation:	**Best-so-far pace-setting individual.**
Success predicate:	**A program scores 20 hits.**