GRISWOLD83
Griswold, R.E. and Griswold, M.T.;
The Icon Programming Language;

This book is the comprehensive reference for the Icon language. It gives a tutorial introduction to Icon, emphasizing the unusual features of the language. The book is divided into four sections, the first three addressing in increasing detail the features of the language, and the last section containing illustrative Icon programs. Several appendices contain the specification of Icon in the form of a summary of its syntax, machine dependencies and implementation limits, a reference guide to the language, and explanation of error messages. Each chapter contains exercises, with some answers available in an appendix, so the book is suitable for use as a tutorial textbook.

It is no surprise that Ralph Griswold who was very much associated with SNOBOL more that a decade ago, would be involved in the design of the Icon language. In many ways Icon seems to be a next-generation SNOBOL that takes basic control structures from structured jprogramming languages. In fact the first section of the book introduces a subset of Icon that will be familiar to people with a background in Pascal or a derivative. Icon offers the Pascal looping constructs, a case statement, if-then-else, etc. So far nothing remarkable about Icon.

Icon uses the same concept of expression success or failure as SNOBOL. The combination of generators, result sequences, and goal-directed evaluation produces a control strategy very similar to the backtracking pattern matching of SNOBOL. In Icon, control backtracking is done automatically but undoing side effects, i.e. data backtracking, is only done automatically for string operations. The language contain a couple of constructs that eases the task of implementing data backtracking explicitly. These constructs are reversible variants of the assignment and exchange statements, that restore the previous value of the assigned to variable when backtracked over.

The idea of goal directed evaluation is a key concept in Icon. It means that an expression will be evaluated every possible way until it succeeds. Since Icon is an expression based language in that every statement fails, or succeeds with a result(sequence), the goal-directed evaluation provides a powerful control strategy.

The general purpose intention of Icon is illustrated by the different data types it supports. The basic numeric types (int and real), and the reasonably well-known types list, file, procedure, and string, are part of the language. But Icon also has some unusual, perhaps unique, types; these are cset, table, and co-expression. These types are likely to be unfamiliar to many people.

Character sets (cset) are used primarily for various matching purposes, and primarily operations on strings. Set operators are defined on csets. Icon provides a kind of associative memory, in the form of tables whose indexes can be of arbitrary type. This capability exists in other languages too (Lisp, SETL) but Icon is the only one providing and supporting tables as a primitive type. Co-expressions are general expressions that have been assigned to a variable for later reference and use. Note that the value of a co-expression is a kind of expression, not the value of an expression. One can refer to one or more instances of the expression by later mentioning the co-expression variable. Because expressions may contain generators either explicitly or because of an operationa on a result sequence, co-expressions variables have associated with them some state information about the co-expression value. In many ways co-expressions behave like small co-routines, but the capabilities apply to any expression and not only to co-routine calls.

The Icon book is a very readable introduction and reference to the language. It is filled with small examples that get the ideas across to the reader, and often goes on to show how the concepts are used in solving complicated problems. Apart from the description of the Icon language in general, the real treasure of the book is that it describes and illustrates many of the exotic constructs in programming languages; among them generators, coroutines, backtracking control structure, and variations on these themes. Icon is

impressive in containing several innovative ideas and also adaptations of old ones. An interesting language and a worthwhile book.