Linear Code Sequence and Jump

Linear code sequence and jump (LCSAJ) is a software analysis method used to identify structural units in code under test. Its primary use is with dynamic software analysis to help answer the question "How much testing is enough?".[1] Dynamic software analysis is used to measure the quality and efficacy of software test data, where the quantification is performed in terms of structural units of the code under test. When used to quantify the structural units exercised by a given set of test data, dynamic analysis is also referred to as coverage analysis.

History

The LCSAJ analysis method was devised by Professor Michael Hennell in order to perform quality assessments on the mathematical libraries on which his Nuclear physics research at the University of Liverpool depended.[2] [3] Professor Hennell later founded the Liverpool Data Research Associates (LDRA) company to commercialize the software test-bed produced for this work, resulting in the LDRA Testbed product.

Introduced in 1976, the Linear Code Sequence and Jump (LCSAJ)[4] is now also referred to as the jump-to-jump path (JJ-path).[5]

Definition

An LCSAJ is a software code path fragment consisting of a sequence of code (a Linear Code Sequence, or basic block) followed by a control flow Jump, and consists of the following three items [6] :

- the start of the linear sequence (basic block) of executable statements
- the end of the linear sequence
- the target line to which control flow is transferred at the end of the linear sequence.

When used for coverage analysis, LCSAJs are considered to have a Test Effectiveness Ratio of 3.

Test Effectiveness Ratio

Coverage analysis metrics are used to gauge how much testing has been achieved. The most basic metric is the proportion of statements executed, considered to have a Test Effectiveness Ratio (TER) of one [7] :

$$\text{TER}_1 = \frac{number\ of\ statements\ executed\ by\ the\ test\ data}{total\ number\ of\ executablestatements}$$

Higher level coverage metrics can also be generated, in particular [8] :

$$\text{TER}_2 = \frac{number\ of\ control - flow\ branches\ executed\ by\ the\ test\ data}{total\ number\ of\ control - flow\ branches}$$

$$\text{TER}_3 = \frac{number\ of\ LCSAJs\ executed\ by\ the\ test\ data}{total\ number\ of\ LCSAJs}$$

These metrics satisfy a pure hierarchy, whereby when TER3 = 100% has been achieved it follows that TER2 = 100% and TER1 = 100% have also been achieved.

Both the TER1 & TER2 metrics were in use in the 1940s and the third dates from the 1970s. The requirements for achieving TER1 = 100% became an industrial norm during the late 1960s, and was the level originally selected for the DO-178 avionics standard until it was supplemented by the MCDC (Modified Condition/Decision Coverage) additional requirement in 1992.[9] Higher levels TER3 = 100% have been mandated for many other projects, including aerospace, telephony and banking.

Example

Consider the following C code[10] :

```
#include    <stdlib.h>
#include    <string.h>
#include    <math.h>

#define MAXCOLUMNS  26
#define MAXROW      20
#define MAXCOUNT    90
#define ITERATIONS  750

int main (void)
{
    int count = 0, totals[MAXCOLUMNS], val = 0;

    memset (totals, 0, MAXCOLUMNS * sizeof(int));

    count = 0;
    while ( count < ITERATIONS )
    {
        val = abs(rand()) % MAXCOLUMNS;
        totals[val] += 1;
        if ( totals[val] > MAXCOUNT )
        {
            totals[val] = MAXCOUNT;
        }
        count++;
    }

    return (0);

}
```

From this code, the following is a complete list of the LCSAJ triples for this code

LCSAJ Number	Start Line	Finish Line	Jump To Line
1	10	17	28
2	10	21	25
3	10	26	17
4	17	17	28
5	17	21	25
6	17	26	17
7	25	26	17
8	28	28	-1

A coverage level of TER3 = 100% would be achieved when test data is used that causes the execution of each of these LCSAJs at least once.

From this example it can be seen that the basic block identified by an LCSAJ triple may span a decision point, reflecting the conditions that must be in place in order for the LCSAJ to be executed. For instance, LCSAJ 2 for the above example includes the **'while'** statement where the condition **'(count < ITERATIONS)'** evaluates to TRUE.

In addition, it can also be seen that each line of code has an LCSAJ 'density' associated with it; line 17, for instance, appears within 6 unique LCSAJs - i.e. it has an LCSAJ density of 6.

This is helpful when evaluating the maintainability of the code; If a line of code is to be changed then the density is indicative of how many LCSAJs will be affected by that change.

References

[1] M.A.Hennell, D.Hedley and M.R.Woodward, "Quantifying the test effectiveness of Algol 68 programs", Proceedings of the Strathclyde ALGOL 68 conference 1977, pp. 36 - 41, ISSN 0362-1340

[2] M. A. Hennell, *An experimental testbed for numerical software. {I}. {Fortran}*, The Computer Journal 21(4):333--336, @nov, 1978

[3] M. A. Hennell and D. Hedley, *An experimental testbed for numerical software. {II}. {ALGOL 68}*, The Computer Journal 22(1):53--56, @feb, 1979

[4] M.A. Hennell, M.R.Woodward and D.Hedley, "On program analysis", Information Processing Letters, 5(5), pp. 136 - 140, 1976

[5] M. R. Woodward, M. A. Hennell, "On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC", Information and Software Technology 48 (2006) pp. 433-440

[6] M.A.Hennell, D.Hedley and I.J.Riddell, "Assessing a Class of Software Tools", Proceedings of the 7th International Conference on Software Engineering March 1984, pp. 266 - 277. ISBN 0270-5257

[7] J.R.Brown, "Practical Application of Automated Software Tools", TRW Report No. TRW-SS-72-05, presented at WESCON, 1972

[8] M.R.Woodward, D.Hedley and M.A.Hennell, “Experience with Path Analysis and Testing of Programs”, IEEE Transactions on Software Engineering, Vol. 6, No. 3, pp. 278 - 286, May 1980

[9] Software Considerations in Airborne System and Equipment Certificaton-RTCA/DO-178B, RTCA Inc., Washington D.C., December 1992

[10] British Standard BS7925-2 "Standard for Software Component Testing" Annex B "Guidelines for Testing and Test Measurement" (http://www.bsigroup.com/en/Shop/Publication-Detail/?pid=000000000001448026)

Article Sources and Contributors

Linear Code Sequence and Jump *Source*: http://en.wikipedia.org/w/index.php?oldid=410761252 *Contributors*: Chutznik, Ipatrol, Nat hillary, Ruud Koot, 2 anonymous edits

License