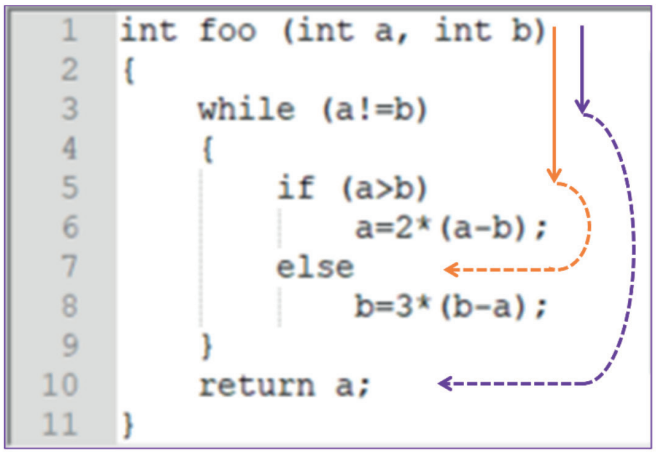
# LCSAJ coverage:
# The most effective code coverage testing mechanism

LCSAJ coverage is the most thorough of the attainable source code coverage metrics. It is attainable in that the number of LCSAJs in a code base makes it practical and proportionate to exercise a large majority of them. And it is thorough in that achieving any given level of LSCAJ coverage exercises more of the code base than achieving the same level using a comparable coverage metric (including function, statement, and branch coverage).

The implementation of LCSAJ coverage analysis in the LDRA tool suite® remains uniquely complete and thorough because of its application of static and dynamic techniques in combination.

## What is an LCSAJ?

An LCSAJ is a path fragment - a linear sequence of executable code (Figure 1). Each LCSAJ commences either from the start of the program or from a point to which control flow may jump. It is terminated either by a specific control flow jump or by the end of the program.




| LCSAJ | Sequence start | Sequence end | Jump |
|---|---|---|---|
| 1 | 1 | 3 | 10 |
| 2 | 9 | 11 | EXIT |
| 3 | 1 | 5 | 7 |
| 4 | 7 | 9 | 3 |
| 5 | 3 | 3 | 9 |
| 6 | 1 | 6 | 3 |
| 7 | 1 | 6 | 3 |
| 8 | 3 | 5 | 7 |

*Figure 1 : Each LCSAJ is a linear sequence of executable code*

## What is LCSAJ coverage?

LCSAJ coverage is a measure of how many LCSAJs are exercised by a particular test data set. It can be expressed as a ratio or a fraction in accordance with the following relationship

$$\text{LCSAJ coverage} = \frac{\text{Number of LCSAJs exercised at least once}}{\text{Total number of LCSAJs}}$$

When combined, LCSAJs form a finite set of paths through the source code application under test. The finite nature of this set of paths means that it is feasible to test all LCSAJs. In contrast, while it may be desirable to attempt to test all control flow paths, it is not technically possible for anything other than the most trivial of programs. This is because in combination, even simple constructs give rise to an exponentially large number of paths and any non-deterministic loop constructs will yield a potentially infinite number of paths.

## Why perform LCSAJ coverage?

The use of LCSAJ coverage forces the analysis of loops and the analysis of combinations of program constructs and hence ensures that more, potentially error yielding, aspects of a software application are analysed than with any other available method (Figure 2).

| Procedure Calls | Statement(100%) | Branch/Decision(100%) | LCSAJ(60%) |
|---|---|---|---|
| TunnelData::Lamp::SetLumensOutput | 100 | 100 | 100 |
| TunnelData::Lamp::SendPowerToLamp | 100 | 100 | 100 |
| TunnelData::Zone::InitialiseZone | 100 | 100 | 92 |
| TunnelData::SystemData::SystemData | 100 | 100 | 100 |

*Figure 2: LCSAJ coverage exposes more potentially troublesome code constructs than any other coverage metric*

Successfully testing all LCSAJs demonstrates that a large set of the program paths are correct, and LCSAJ coverage is therefore well suited to the analysis of high integrity software applications.

### Functional safety

LCSAJ coverage was designed for use in the development of safety-critical applications. Functional safety standards including IEC 61508 (safety-related systems), ISO 26262 (automotive) and IEC 62304 (medical devices) recommend the use of code coverage analysis, with some standards including EN 50128 (railways) specifically recommending the use of LCSAJ coverage.

### Secure coding & cybersecurity

Secure coding has an important part to play in any defence-in-depth strategy. The white-box DAST provided by the LDRA tool suite enables compiled and executed code to be tested in the development environment or, better still, on the target hardware. Code coverage facilitates confirmation that all security and other requirements are fulfilled by the code, and conversely that all code fulfils one or more requirements. Standards including IEC 62443 (industrial automation) and ISO/SAE 21434 (automotive) recommend code coverage analysis for those reasons. LCSAJ coverage is the most effective code coverage testing mechanism for security- and safety-critical applications alike.

## Measuring LCSAJ coverage with the LDRA tool suite

The LDRA tool suite measures the LCSAJ coverage attained when sets of test data are applied to the source code application under test (Figure 3).

**Linear Code Sequence and Jump (LCSAJ) Profile**

| | |
|---|---|
| **Number of LCSAJs** | 30 |
| **Number Executed** | 29 |
| **Number not Executed** | 1 |
| **LCSAJ Coverage (%)** | **97** |

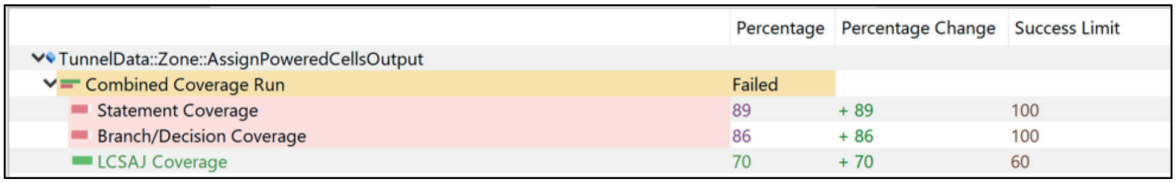
**Linear Code Sequence and Jump (LCSAJ) Coverage Table**

| Start Line | Finish Line | Jump To Line | Coverage |
|---|---|---|---|
| 268 | 290 | 293 | 90 |
| 289 | 290 | 293 | 90 |
| 289 | 290 | 296 | 90 |
| 291 | 292 | 289 | 180 |
| 293 | 295 | 291 | 180 |
| 296 | 306 | 309 | 90 |
| 296 | 306 | 329 | 0 *** |
| 305 | 306 | 309 | 95 |

*Figure 3: Reporting on LCSAJ coverage using the LDRA tool suite*

The static analysis features of the tool suite identify all the vital control flow information for each procedure and the inter-procedural links. The interfaces are accurately delineated, the loop structure is exposed, and complexity metrics are generated. This information provides the basis for the definition of the LCSAJ “start, end, jump to” triples.

Structural coverage is collated as the code is then executed using the dynamic analysis features of the LDRA tool suite during unit, integration, and/or system test. The execution count is listed for each LCSAJ, unexecuted LCSAJs are highlighted, and the collated coverage statistics from multiple executions are tabulated (Figure 4).



| | Percentage | Percentage Change | Success Limit |
|---|---|---|---|
| TunnelData::Zone::AssignPoweredCellsOutput | | | |
| Combined Coverage Run | Failed | | |
| Statement Coverage | 89 | + 89 | 100 |
| Branch/Decision Coverage | 86 | + 86 | 100 |
| LCSAJ Coverage | 70 | + 70 | 60 |

*Figure 4: Structural coverage can be collated from multiple unit, integration, and system tests by the LDRA tool suite*

Unreachable LCSAJs are also listed, which will always start from lines of code that are themselves unreachable. The removal of unreachable code yields more readable and efficient code and reduces the attack surface of the code base.

### LCSAJ density

LCSAJ density is a maintainability metric. If a line of code is to be changed then the density informs the user how many LCSAJs (and hence paths) may be affected by that change. If the density is high then confidence that the change is correct for all LCSAJs will be reduced, and therefore an increased amount of regression testing will be required.

## Conclusion

Source code coverage analysis is strongly recommended by functional and cybersecurity standards alike. LCSAJ coverage is the most effective code coverage testing mechanism yet devised, and the LDRA implementation of that mechanism is the most complete available.

LDRA’s products and services are widely used by companies whose names are synonymous with embedded electronic systems across a broad range of industrial sectors. The LDRA tool suite is TÜV SÜD certified and complies with the demands of IEC 61508, IEC 62304, IEC 60880, EN 50128 and ISO 26262, and the company’s Quality Management System is certified in accordance with ISO 9001.

More information can be found at www.ldra.com/lcsaj.


www.ldra.com

**LDRA UK & Worldwide**
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

**LDRA Technology Inc.**
2540 King Arthur Blvd, Suite 228,
Lewisville, Texas 75056
United States
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

**LDRA Technology Pvt. Ltd.**
Unit No B-3, 3rd Floor Tower B,
Golden Enclave, HAL Airport Road,
Bengaluru
560017
India
Tel: +91 80 4080 8707
e-mail: india@ldra.com


LCSAJ Technical note v 2.0 06/22