# turnHeaterOn(Programmedsettings programmedSettings)

## Number of Pridate

There are two predicates in the *turnHeaterOn* method. The first predicate has 4 clauses, and the second predicate has 1 clause.

* Predicate 1 = P1 = **if** (((curTemp < dTemp - thresholdDiff) || (override && curTemp < overTemp - thresholdDiff))&& (timeSinceLastRun > minLag))

The clause for this predicate is as follows:

* + clause “a”, which is: curTemp < dTemp – thresholdDiff.
  + clause “b”, which is: override.
  + clause “c”, which is: curTemp < overTemp – thresholdDiff.
  + clause “d”, which is: timeSinceLastRun > minLag.
* Predicate 2 = P2 = **if** (override), which is only one clause.

Moving on, I will use a, b, c, d notation for each predicate. For example, we can write predicate 1 as:

*p1 = (a || (b && c)) && d*

p2 = override

Note: We create the truth table for each predicate once here and refer to this table here after.

The truth table for p1 is as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 2 | T | T | T | F | T | T | F |
| 3 | T | T | F | T | F | T | T |
| 4 | T | T | F | F | F | T | F |
| 5 | T | F | T | T | F | T | T |
| 6 | T | F | T | F | F | T | F |
| 7 | T | F | F | T | F | T | T |
| 8 | T | F | F | F | F | T | F |
| 9 | F | T | T | T | T | T | T |
| 10 | F | T | T | F | T | T | F |
| 11 | F | T | F | T | F | F | F |
| 12 | F | T | F | F | F | F | F |
| 13 | F | F | T | T | F | F | F |
| 14 | F | F | T | F | F | F | F |
| 15 | F | F | F | T | F | F | F |
| 16 | F | F | F | F | F | F | F |

The truth table for p2 is as follows:

|  |  |
| --- | --- |
| No | Override |
| 1 | T |
| 2 | F |

## Predicate Coverage (PC):

For p1 we can use row 1 and row 2 of p1 truth table. In this way, we can test both values (T/F) of p1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 2 | T | T | T | F | T | T | F |

Values for PC:

By changing the value of timeSinceLastRun from 10 to 3 we change the value of predicate from true to false.

|  |  |
| --- | --- |
| Row 1 values | Row 2 values |
| * curTemp = 30 * dTemp = 50 * thresholdDiff = 5 * override = true * overTemp = 55 * timeSinceLastRun = 10 * minLag = 5 | * curTemp = 30 * dTemp = 50 * thresholdDiff = 5 * override = true * overTemp = 55 * timeSinceLastRun = 3 * minLag = 5 |

P2 (override).

We use both rows of p2 truth table. In other words, override = true and override = false.

Junit test cases runs for both p1 and p2. There are 4 test cases, two test cases for p1 and two test cases for p2.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Clause Coverage

We use row1 and row16 of p1 truth table for clause coverage. Since value of each clause changes from T to F from row 1, this satisfies the clause coverage test requirement.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 16 | F | F | F | F | F | F | F |

There are four clauses in p1:

* clause “a”, which is: curTemp < dTemp – thresholdDiff.
* clause “b”, which is: override.
* clause “c”, which is: curTemp < overTemp – thresholdDiff.
* clause “d”, which is: timeSinceLastRun > minLag.

For clause all clauses we will use row 1 and 16.

This makes our job easier since all clauses’ values are changed from T to F. The value on the left column makes each clause evaluate to T and change the value (on right column) make the clause to evaluate to F.

|  |  |
| --- | --- |
| **Value for all clauses (a = b = c = d = true, p1=true)** | **Value for all clauses (a = b = c = d = false, p1=false)** |
| curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 10  minLag = 5 | curTemp = 70  dTemp = 50  thresholdDiff = 5  override = false  overTemp = 55  timeSinceLastRun = 3  minLag = 5 |

* For p2 (override) since it has one clause with two values (T/F) it has been subsumed by p1. See below screenshot for details.

Below screenshot displays the two test cases (test runs) that evaluate all clauses to T and F and check the predicate final value via assertion. p2 is subsumed by p2, thus I did not write an extra test case for that.

A screenshot of a computer

Description automatically generated

## CACC:

There are four clauses in P1:

* clause “a”, which is: curTemp < dTemp – thresholdDiff.
* clause “b”, which is: override.
* clause “c”, which is: curTemp < overTemp – thresholdDiff.
* clause “d”, which is: timeSinceLastRun > minLag.

For clause “a” to be active clause we will use row 1 and 11.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 11 | F | T | F | T | F | F | F |

Values:

|  |  |
| --- | --- |
| Value for row 1 (a=true, p1 =true) | Values for row 11 (a=false, p1 = false) |
| curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 10  minLag = 5 | curTemp = 70  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 3  minLag = 5 |

For clause “b” to be active clause we will use row 1 and 13.

* clause “a”, which is: curTemp < dTemp – thresholdDiff.
* clause “b”, which is: override.
* clause “c”, which is: curTemp < overTemp – thresholdDiff.
* clause “d”, which is: timeSinceLastRun > minLag.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 13 | F | F | T | T | F | F | F |

Values:

|  |  |
| --- | --- |
| Value for row 1 (b=true, p1 =true) | Values for row 11 (b=false, p1 = false) |
| curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 10  minLag = 5 | curTemp = 70  dTemp = 50  thresholdDiff = 5  override = false  overTemp = 55  timeSinceLastRun = 10  minLag = 5 |

For clause “c” to be active clause we will use row 1 and 4.

* clause “a”, which is: curTemp < dTemp – thresholdDiff.
* clause “b”, which is: override.
* clause “c”, which is: curTemp < overTemp – thresholdDiff.
* clause “d”, which is: timeSinceLastRun > minLag.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 4 | T | T | F | F | F | T | F |

Values:

|  |  |
| --- | --- |
| Value for row 1 (c=true, p1 =true) | Values for row 4 (c=false, p1 = false) |
| curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 10  minLag = 5 | curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 3  minLag = 5 |

For clause “d” to be active clause we will use row 1 and 2.

* clause “a”, which is: curTemp < dTemp – thresholdDiff.
* clause “b”, which is: override.
* clause “c”, which is: curTemp < overTemp – thresholdDiff.
* clause “d”, which is: timeSinceLastRun > minLag.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 2 | T | T | T | F | T | T | F |

Values:

|  |  |
| --- | --- |
| Value for row 1 (c=true, p1 =true) | Values for row 4 (c=false, p1 = false) |
| curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 10  minLag = 5 | curTemp = 30  dTemp = 50  thresholdDiff = 5  override = true  overTemp = 55  timeSinceLastRun = 3  minLag = 5 |

Test run for CACC – all clauses:  
A screenshot of a computer

Description automatically generated

**Note:** I understand that selecting only rows 1 and 16 can satisfy the CACC for all clauses. However, I deliberately chose different rows for each clause for two key reasons: to deepen my understanding of the concept and to demonstrate alternative methods of achieving CACC beyond selecting just two rows.

In CACC, the active clause's value determines the predicate's value, while the values of other minor clauses are irrelevant. I argue that choosing rows 1 and 2 satisfies CACC for all clauses. For example, if we select 'a' as the major clause, its value changes from T to F, consequently changing the predicate's value from T to F (the values of 'b,' 'c,' and 'd' do not matter). Similarly, choosing 'b' as the active clause results in its value changing from T to F, leading to the predicate's value changing from T to F, with the values of 'a,' 'c,' and 'd' being inconsequential in this scenario

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | a | b | c | d | b & c | a v (b & c) | ((a v (b & c)) & d |
| 1 | T | T | T | T | T | T | T |
| 16 | F | F | F | F | F | F | F |