Acyclic complexity (draft)

Abstract. *In this paper we propose a quadratic code complexity measure [Blo19] from the perspective of testing [wik18] with the conjecture of indipendent enclosed scopes, the measure enfolds elementary condition collocation, nested and repeated formations.*

1 Introduction

A Progam consist of a sequences of instructions. The instructions can be categorized into syntax, logic and arithmetic determining the program control flow.

Software complexity is related with modularity, coupling and cohesion hence quality. 40% to 80% of software costs are emerging due to maintenance and approximatly 40% on fixing defects [SG11].

Analysing code complexity helps to identify risk, finds potential defects to test critical functionality in detail, increase quality, cohesion and decrease maintainance [RB11].

2 Existing measures

Cyclomatic complexity (CC) has been widely discussed by various authors. The most used metric has been formulated by Thomas J. McCabe in 1976 [McC76].

$$m = e - n + 2 \tag{1}$$

Where, e = the amount of edges. n = the amount of nodes.

CC is linear in it's nature and correlation with lines of code (LOC), which is why Graylin JAY et al. [al.09] are suggesting to implement LOC as complexity measure.

Mir Muhammd Suleman Sarwar et al. [MMSS13] are pointing out that it neither takes into account the difference between combined decisions, elementary conditions nor repeating structures. Their adaptation of CC includes loop iterations

$$V(G)* = V(G) + \prod_{i=1}^{n} P_i$$
$$P_i = U_i - L_i + 1$$

Where, P_i = No. of iterations of ith loop. U_i = upper bound of ith loop. L_i = lower bound of ith loop and $V(G)*$ = adjusted cyclomatic complexity for any control flow graph "G". The measure combines control flow and statements exercised.

Brian A. Nejmeh. NPATH [Nej88] TODO.

Tevfik et al. [LB15] are providing a reasonable complutable measure: asymptotic path complexity covering nested structures and loops in comparison with CC and NPATH. TODO

2.1 Coverage

Test coverage metrics are providing quantitative representation of tested structures. The approach is to maximize coverage while minimizing testing effort. Each condition branches the flow into two control sub-paths, and determines the progression of the programm. The path of an isolated decision d_j thereby is given due to decisions and conditions (MC/DC) [RB11]. Let's define a condition c_j and it's path $\gamma : C \times \Sigma \to C^*$

$$\Sigma = \{true, false\}$$

$$C = \{c_0, c_1, c_2, c_3, ..., c_n\}, c_i \mapsto \Sigma$$

$$\varepsilon : C \to \Sigma \times C$$

$$D \subseteq \{C^*, S, E\}; d_i \in D$$

The transition function $alpha$

$$\alpha : D \times \Sigma^* \to \Sigma \times D$$

and the transition $\vdash$

$$\vdash \subseteq (\Sigma^* \times D \times \Sigma) \times (\Sigma^* \times D \times \Sigma)$$

The possible combinations of conditions on γ are arising through c_j's preceeding and succeeding condition flow. An acyclic transition path through c_j is described due to passing each condition once

$$\begin{aligned} P_j &= (v_i, d_i, b_i), ... \vdash (v_j, d_j, b_j) \vdash^* (v_n, d_n, b_n) \\ &= (c_i, b_i), ... \vdash (c_j, b_j) \vdash^* (c_n, b_n) \end{aligned}$$

where

$$\alpha(d_n, v_n) \mapsto E$$

with the mantle

$$P = (b_i, v_i, d_i) \vdash^* (b_n, v_n, d_n)$$

3 Hypothesis acyclic complexity

Instead of combining arithmetic and logic we propose a metric [Blo19] reflecting the logical test effort indicated due to the amount of paths. The concept of basic paths described by [McC76] neglects loops and nesting. 100% sub-path combination coverage q is intricate due to exponential effort [wik18]. Conditions are providing the logical structure of programs (Listing 1-12). Loops and recursions are construed due to their boundary [RB11] and invariant, which is why their quantitative iteration increases inclination not path length. Let's define the inductive start of independent

$$\begin{aligned}\alpha_i =&(c_0, true) \to (c_1), (c_0, false) \to (c_1),\\ &(c_1, true) \to (E), (c_1, false) \to (E)\end{aligned}$$

and dependent branches

$$\begin{aligned}\alpha_d =&(c_0, true) \to (E), (c_0, false) \to (c_1),\\ &(c_1, true) \to (E), (c_1, false) \to (E)\end{aligned}$$

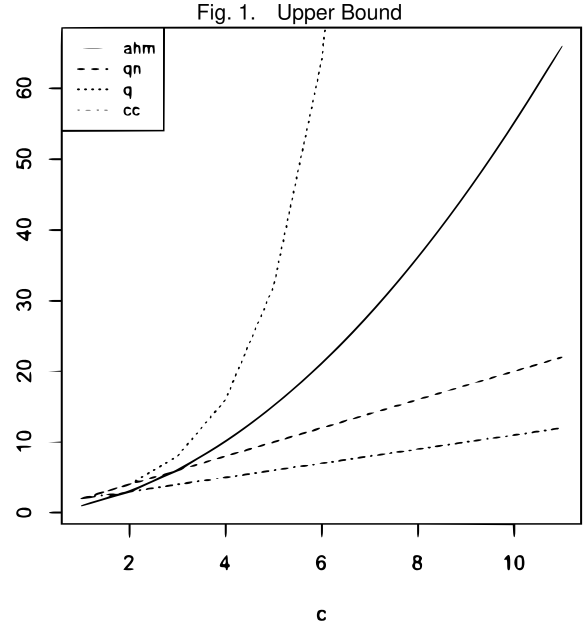
Isolated condition paths [wik18] are $log(2^n)$ subset of all q paths. If we draw the spanning trees the first branch expands four leaves, the latter three leaves.

Nesting is diminishing the amount of paths, the effort of combinatorial parameter testing is unaffected. The fallacy of subjective perceived complexity increases contrary due to remembering previous conditions (Table 2).

Table 1. Condition collocation bound

		$O(n)$	$O(2n)$	$O(n^2)$	$O(2^n)$
$\|C\|$	loc	cc	q_2	ahm	q
1	$1-l_0$	2	$2-2$	$\frac{1}{2}-1$	$2-2$
2	$1-l_1$	3	$3-4$	$1-3$	$3-4$
3	$1-l_2$	4	$4-6$	$\frac{3}{2}-6$	$4-8$
4	$1-l_3$	5	$5-8$	$2-10$	$5-16$
5	$1-l_4$	6	$6-10$	$2\frac{1}{2}-15$	$6-32$
6	$1-l_5$	7	$7-12$	$3-21$	$7-64$
7	$1-l_6$	8	$8-14$	$3\frac{1}{2}-28$	$8-128$
...					
9	$1-l_7$	10	$10-18$	$4\frac{1}{4}-45$	$10-512$
...					
19	$1-l_8$	20	$20-38$	$9\frac{1}{2}-190$	$20-524288$
...					
49	$1-l_9$	50	$50-98$	$24\frac{1}{2}-1225$	$50-5*10^{14}$

Fig. 1. Upper Bound



Listing 1. plain singular

```
if (-1<l) { }
```

Listing 2. orthogonal dual

```
if (3>l && 0<k) { }
```

Listing 3. tertiary orthogonal

```
if (2>l && 7>k && 10>n) { }
```

Listing 4. orthogonal dual

```
if (2>l || 0<k) { }
```

Listing 5. orthogonal dual nested

```
if (0<l) {
    ...
    if (2>k) { }
    ...
}
```

Listing 6. tertiary orthogonal nested

```
if (0<l && 4>k) {
  ...
  if (11>n) { }
  ...
}
```

Listing 7. tertiary

```
if (2>l && 0<k || 4>n) { }
```

Listing 8. tertiary parallel

```
if (0<l || 9>k && 1<n) { }
```

Listing 9. parallel

```
if (2>l) { }
...
if (8>k) { }
```

Listing 10. tertiary parallel identical scope

```
if (2>l || 0<k || 4>n) { }
```

Listing 11. tertiary orthogonal nested

```
if (2>l) {
  ...
  if (8>k) {
    ...
    if (0<n) { }
    ...
  }
}
```

Listing 12. singular nested parallel

```
if (0<l) {
  ...
  if (4>k) { }
  ...
  if (8>n) { }
  ...
}
```

Listing 13. orthogonal singular parallel

```
if (0<l) {
  ...
  if (8>k) { }
  ...
}
if (0<n) { }
```

Listing 14. dual parallel

```
if (0<l) { }
...
if (9>k && 1<n) { }
```

Listing 15. dual parallel

```
if (0<l) { }
...
if (9>k) {
    ...
    if (1<n) { }
}
```

Listing 16. tertiary parallel different scope

```
if (-4<l) { }
...
if (4<k) { }
...
if (8>n) { }
```

Table 2. Collocation complexity

Listing	$\|C\|$	loc	cc	q_2	ahm	q
1	1	1	2	1	1	2
2	2	1	3	3	$1\frac{1}{2}$	3
3	3	1	4	5	$1\frac{5}{6}$	4
4	2	1	3	3	2	3
5	2	5	3	3	2	3
6	3	5	4	5	$2\frac{1}{2}$	4
7	3	1	4	5	$2\frac{1}{2}$	4
8	4	1	4	5	2.5	4
9	2	3	3	2	3	4
10	3	1	4	5	3	4
11	3	8	4	5	3	4
12	3	7	4	4	4	5
13	3	6	4	4	4	6
14	3	4	4	4	4	6
15	3	6	4	4	4	6
16	3	5	4	3	6	8

Fig. 3. cc/ahm

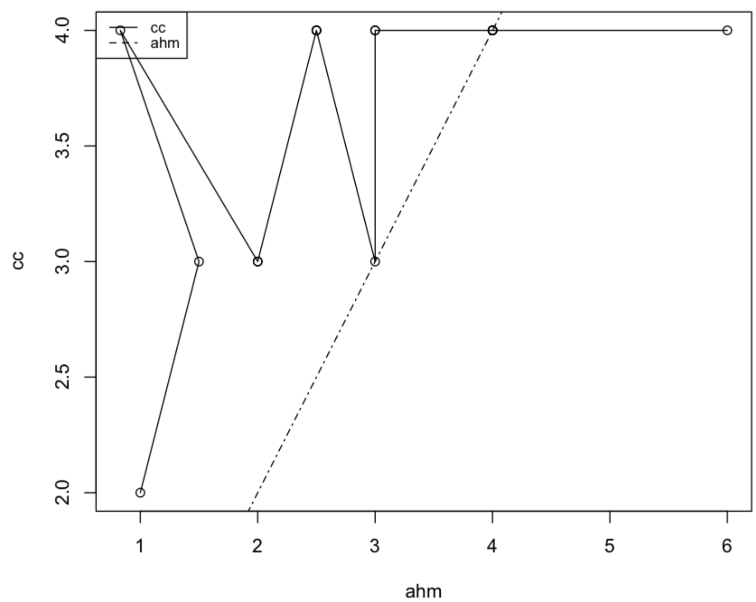
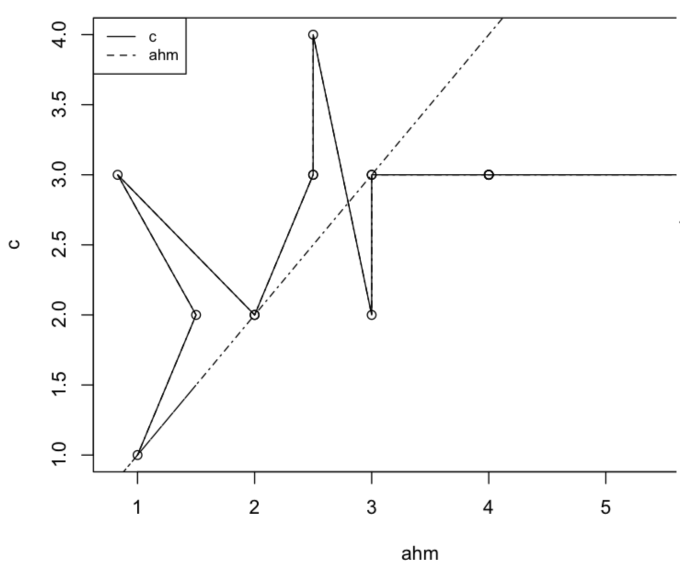


Fig. 4. loc/ahm

Fig. 2. c/ahm



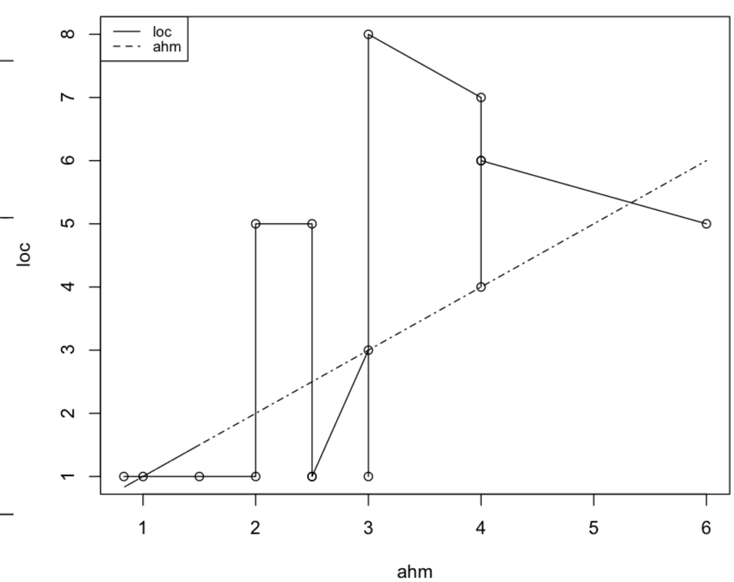


Fig. 5. q/ahm

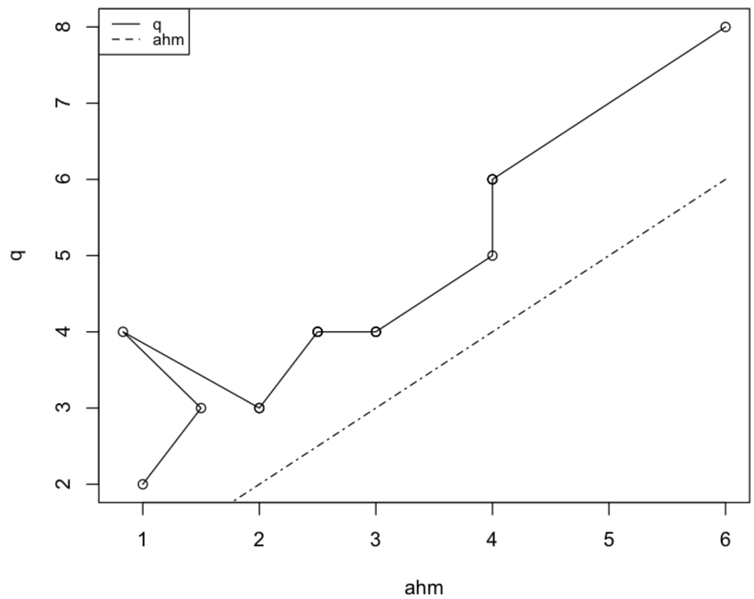


Fig. 6. q2/ahm

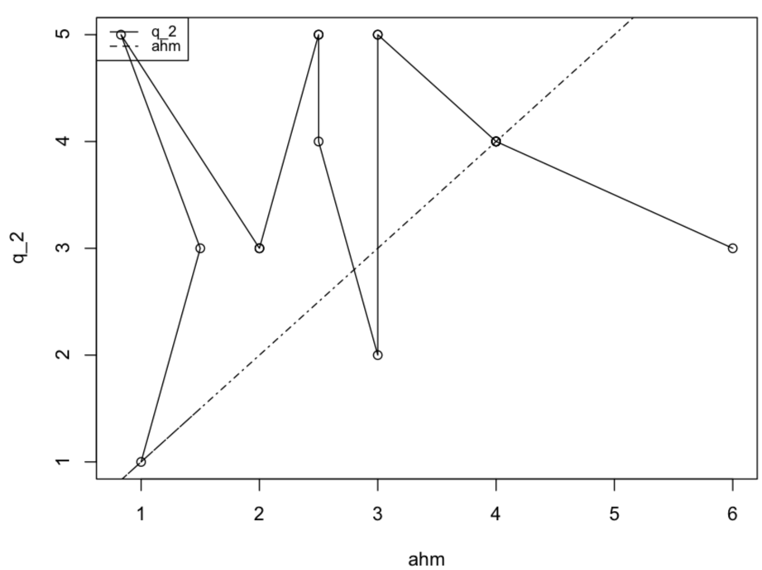


Table 3. α_i flow matrix, $q_2(\alpha_i) = 2$

	s	c_0	c_1	e
s		t		
c_0			(t,f)	
c_1				(t,f)
e				

Table 4. α_d flow matrix, $q_2(\alpha_d) = 3$

	s	c_0	c_1	e
s		t		
c_0			f	t
c_1				(t,f)
e				

The exponential amount of condition combinations 2^n is disected due to exclusiv control flow q.

$$\begin{aligned} &n = |C| \\ &2^n <= q <= n+1 \end{aligned} \tag{2}$$

. When every condition is isolated the amount of condensed edges (enclosed scope paths) q_2 in the condition flow matrix represents an intuitive measure (Table 2, 1, 3, 4).

The lower and upper bounds of possible isolated condensed condition transition paths are

$$\begin{aligned} 2*n >= q_2 >= n+1 \\ C_e \subset C; \quad m_u = |C| + |C_e| \end{aligned} \tag{3}$$

with $C_e \subset C$ nested conditions. The metric m_u is undifferentiated regarding condition nesting level. Subpath combinations caused from parallel conditions (flat-style) are neglected, as well as the subjective perception of condition memory, therefor we define the condition nesting function $\lambda(c_i)$

$$\begin{aligned} \frac{n}{2} < m_{ah} <= \frac{n(n+1)}{2} \\ m_{ah} = \quad \sum \lambda(c_i) \quad = \quad \sum_{i=0}^{n} \frac{r_i}{e_i} \end{aligned} \tag{4}$$

Table 5. Complexity Risk [Cha05]

ahm	risk
1-21	basic program
22-45	intricate, moderate risk
45-	circuitous, high risk

4 Conclusion

The suggested acyclic quadratic complexity measure m_{ah} [Blo19] expresses the logical test effort of condensed path coverage, diminishing unapparent complex patterns quantifying control flow into reasonable absolute numbers.

References

[al.09] AL., Graylin J.: Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship. In: . *Software Engineering & Applications* 2 (2009), June, S. 137–143

[Blo19] BLOEMENDAL, Jannis: ahm, acyclic complexity. (2019), April. – http://github.com/jbloemendal/ahm

[Cha05] CHARNEY, Reg.: Programming Tools: Code Complexity Metrics. In: *Linux Journal* (2005), January

[LB15] LUCAS BANG, Tevfik B. Abdulbaki Aydin A. Abdulbaki Aydin: Automatically computing path complexity of programs. In: *ESEC/SIGSOFT FSE 2015* (2015), S. 61–72

[McC76] MCCABE, Thomas J.: A complexity measure. In: *IEEE* (1976), Dec, Nr. 4, S. 308–320

[MMSS13] MIR MUHAMMD SULEMAN SARWAR, Ibrar A. Sara Shahzad S. Sara Shahzad: Cyclomatic Complexity. In: *IEEE* 1 (2013), Jan, Nr. 5, S. 274–279

[Nej88] NEJMEH, Brian A.: NPATH: A Measure of Execution Path Complexity and Its Application. In: *ACM* 31(2) (1988), S. 188–200

[RB11] REX BLACK, Jamie M.: *Advanced Software Testing Vol. 3*. Santa Barbara, CA : rookynook, 2011

[SG11] SOUMIN GHOSH, Prof. (Dr.) Ajay R. Sanjay Kumar Dubey D. Sanjay Kumar Dubey: Comparative Study of the Factors that Affect Maintainability. In: *International Journal on Computer Science and Engineering (IJCSE)* 3 (2011), Dec, Nr. 12, S. 3763–3769

[wik18] *Elementary comparision testing, control flow test case design method*. `https://en.wikipedia.org/wiki/Elementary_comparison_testing`. Version: 2018