How Testable are BDI Agents? An Analysis of Branch Coverage

(Extended Abstract)

Michael Winikoff
Department of Information Science, University of Otago
Dunedin, New Zealand
michael.winikoff@otago.ac.nz

ABSTRACT

In this paper we extend our understanding of the feasibility of testing BDI agent programs by analysing their testability with respect to the *all edges* test adequacy criterion, and comparing with previous work that considered the *all paths* criterion. Our findings extend the earlier analysis with respect to the all paths criterion to give a more nuanced understanding of the difficulty of testing BDI agents.

Keywords

Verification and validation; agent-based systems

1. INTRODUCTION

When any software system is deployed, it is important to have assurance that it will function as required. Traditionally, this assurance is obtained by testing. However, there is a general intuition that agents exhibit behaviour that is complex. Given this complexity, a key question is whether agent systems are harder, and possibly even infeasible, to assure by testing.

The only work that we are aware of that considers the question of testability is the recent work by Winikoff & Cranefield [3, 4], which investigates the testability of Belief-Desire-Intention (BDI) agent programs with respect to the *all paths* test adequacy criterion. They concluded that BDI agent programs do indeed give rise to a very large number of possible paths (see left part of Table 1). They therefore conclude that whole BDI programs are likely to be infeasible to assure via testing. They also compared BDI programs with procedural programs, and found that BDI programs are *harder* to test than equivalently sized procedural programs. However, they do acknowledge that the all paths criterion is known to be overly conservative, i.e. it requires a very large number of tests. Indeed, the all paths criterion *subsumes* a wide range of other criteria, including all edges.

In this paper we consider the testability of BDI agent programs with respect to the *all edges* [2] test adequacy criterion. Whereas the all paths criterion used by Winikoff & Cranefield is conservative, the all edges criterion is optimistic: it is regarded as "*the generally accepted minimum*" [1]. The contribution of this paper is to extend the previous work to obtain a better understanding of, and a tighter bound on, the testability of BDI agent programs.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

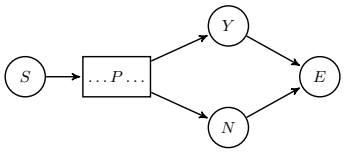
2. ALL-EDGE COVERAGE ANALYSIS

Given a program and a *test adequacy criterion* (e.g. *all paths*, *all edges*), we consider the testability of a program to be the smallest number of tests[1] that would be required to satisfy the criterion. The *all paths* criterion is satisfied iff the set of tests in the test suite T cover all *paths* in the program's control flow graph. The *all edges* criterion (also referred to as "branch coverage") is satisfied iff the set of paths in the test suite T covers all *edges* in the control-flow graph [2].

We define a BDI program P using the grammar:

$$P ::= a \mid g^{\{P^*\}} \mid P_1;P_2 \mid P_1 \triangleright P_2$$

where a is an action, $g^{\mathcal{P}}$ is a (sub-)goal with associated applicable plans $\mathcal{P} = \{P_1, \ldots, P_n\}$, $P_1; P_2$ is a sequence, and $P_1 \triangleright P_2$ represents a "backup plan": if P_1 succeeds, then nothing else is done (i.e. P_2 is ignored), but if P_1 fails, then P_2 is used.



One important feature of BDI programs is that the execution of a BDI program (or subprogram) can either succeed or fail. A failed execution triggers failure handling. We represent this by mapping a program P to a graph that is reachable from the start node S, and that has *two* outgoing edges: to Y (corresponding to a successful execution) and to N (corresponding to a failed execution).

We have derived equations (details omitted, results in Figure 1) that calculate the smallest number of paths from S to E required such that all edges appear at least once in the set of paths. In order to do this, it turns out that we need to also capture how many of these paths correspond to successful executions (go via Y) and how many go via N. We define $\mathbf{p}(P)$ to be the number of paths required to cover all edges in the graph corresponding to program P. We also define $\mathbf{y}(P)$ (respectively $\mathbf{n}(P)$) to be the number of these paths that go via Y (respectively N). By construction we have that $\mathbf{p}(P) = \mathbf{y}(P) + \mathbf{n}(P)$.

3. RESULTS

We now use the derived equations (Figure 1) to compare the all edges criterion against the all paths criterion. We know that the all paths criterion requires more tests to be satisfied, but how many more? Since comparing (complex) formulae is not easy, we follow the approach of Winikoff & Cranefield, and instantiate the formulae with a range of plausible values, to obtain actual numbers that can

[1] A single test corresponds to a path through the program's control-flow graph from its starting node to its final node.

Parameters			Number of ...			All Paths		All Edges	All Edges
j	k	d	goals	plans	actions	$n^{\checkmark}(g)$	$n^{\times}(g)$	$\mathbf{p}(g)$	$\mathbf{q}(Q)$
2	2	3	21	42	62 (13)	6.33×10^{12}	1.82×10^{13}	78	62
3	3	3	91	273	363 (25)	1.02×10^{107}	2.56×10^{107}	2,961	363
2	3	4	259	518	776 (79)	1.82×10^{157}	7.23×10^{157}	808	776
3	4	3	157	471	627 (41)	3.13×10^{184}	7.82×10^{184}	4,767	627

Table 1: Comparison of All Paths and All Edges analyses. The first number under "actions" (e.g. 62) is the number of actions in the tree, the second (e.g. 13) is the number of actions in a single execution where no failures occur.

$$
\begin{aligned}
\mathbf{p}(a) &= 2 \qquad \mathbf{y}(a) = 1 \qquad \mathbf{n}(a) = 1 \\
\mathbf{p}(P_1; P_2) &= \mathbf{n}(P_1) + \max(\mathbf{y}(P_1), \mathbf{p}(P_2)) \\
\mathbf{y}(P_1; P_2) &= \mathbf{y}(P_2) + \epsilon_1 \\
\mathbf{n}(P_1; P_2) &= \mathbf{n}(P_1) + \mathbf{n}(P_2) + \epsilon_2 \\
&\quad \text{where } \epsilon_1 + \epsilon_2 = \max(0, \mathbf{y}(P_1) - \mathbf{p}(P_2)) \\
\mathbf{p}(P_1 \triangleright P_2) &= \mathbf{y}(P_1) + \max(\mathbf{n}(P_1), \mathbf{p}(P_2)) \\
\mathbf{y}(P_1 \triangleright P_2) &= \mathbf{y}(P_1) + \mathbf{y}(P_2) + \epsilon_3 \\
\mathbf{n}(P_1 \triangleright P_2) &= \mathbf{n}(P_2) + \epsilon_4 \\
&\quad \text{where } \epsilon_3 + \epsilon_4 = \max(0, \mathbf{n}(P_1) - \mathbf{p}(P_2)) \\
\mathbf{p}(g^{\{P\}}) &= \mathbf{p}(P) \\
\mathbf{y}(g^{\{P\}}) &= \mathbf{y}(P) \\
\mathbf{n}(g^{\{P\}}) &= \mathbf{n}(P) \\
\mathbf{p}(g^{\mathcal{P}}) &= \sum_{P_i \in \mathcal{P}} \mathbf{y}(P_i) + \max(\mathbf{n}(P_i), \mathbf{p}(g^{\mathcal{P} \setminus \{P_i\}})) \\
\mathbf{y}(g^{\mathcal{P}}) &= \sum_{P_i \in \mathcal{P}} \mathbf{y}(P_i) + \mathbf{y}(g^{\mathcal{P} \setminus \{P_i\}}) + \epsilon_i \\
\mathbf{n}(g^{\mathcal{P}}) &= \sum_{P_i \in \mathcal{P}} \mathbf{n}(g^{\mathcal{P} \setminus \{P_i\}}) + \epsilon'_i \\
&\quad \text{where } \epsilon_i + \epsilon'_i = \max(0, \mathbf{n}(P_i) - \mathbf{p}(g^{\mathcal{P} \setminus \{P_i\}}))
\end{aligned}
$$

Figure 1: Equations to calculate $\mathbf{p}(P)$, $\mathbf{y}(P)$ and $\mathbf{n}(P)$

be compared. We use the same scenarios (i.e. parameters) that they used, and compare "uniform" BDI programs [3, Section 4]

Table 1 contains the results for these illustrative comparison cases (ignore the rightmost column for now). The left part of the Table (Parameters, Number of goals, plans, and actions, and All Paths) are taken from the all paths analysis of Winikoff & Cranefield [3]. The right part (All Edges, $\mathbf{p}(g)$) is the new numbers from this work.

As expected, the number of tests required to adequately test a given BDI program P with respect to the all edges test adequacy criterion is lower than the number of tests required with respect to the all paths criterion. However, what is interesting is that the numbers are very much lower. Indeed, the number of tests required with respect to the all edges criterion is sufficiently small to be feasible. For instance, in the third case ($j = 2, k = 3, d = 4$) where the (uniform) BDI program has 259 goals and 518 plans, corresponding to a non-trivial agent program, the number of required test cases is less than 1000. However, it is worth emphasising that the all edges criterion, even for traditional software, is regarded as a *minimum*. Additionally, it can be argued that agents, which are situated in an environment that is typically non-episodic, might be more likely than traditional software to be affected by the history of their interaction with the environment [3, Section 1.1], and therefore the all paths criterion would be more appropriate.

We also considered the question of whether testing BDI agent programs is *harder*. We did this by comparing the number of tests required to adequately test a BDI agent program (with respect to the all edges criterion) with the number of tests required to adequately test an equivalent-sized (abstract) procedural program.

Following Winikoff & Cranefield [3] we define an abstract procedural program as $Q ::= s \mid Q + Q \mid Q; Q$ where the base case is a statement s, and a compound program can be a combination of programs either in sequence ($Q_1; Q_2$), or as an alternative choice ($Q_1 + Q_2$). We then consider how many tests (i.e. paths) are required to cover all edges in the graph corresponding to a procedural program Q. We denote this number by $\mathbf{q}(Q)$.

The rightmost column of Table 1 shows the number of tests (paths) required to test a procedural program Q of the same size[2] as the BDI program in question for that row. Although in some cases $\mathbf{p}(g)$ is quite close to $\mathbf{q}(Q)$ (e.g. 78 vs. 62 in the first row), in other cases there is a significant difference (e.g. 4767 vs. 627). In essence, the more plans there are per goal (j), the bigger the difference. Overall, we conclude that where goals have multiple plans available, then testing a BDI agent program is indeed harder than testing an equivalently-sized procedural program. This lends strength to the earlier result of Winikoff & Cranefield, who found that BDI agent programs were harder to test than equivalently sized procedural programs with respect to the all paths criterion.

In conclusion, we analysed the testability of BDI programs with respect to the all edges criterion. Our findings extend the earlier analysis with respect to the all paths criterion to give a more nuanced understanding of the difficulty of testing BDI agents. We conclude that the number of required tests to satisfy the all edges criterion is not just lower, as expected, but very much lower. We also conclude that, as is the case for all paths, testing BDI agents is harder than testing equivalently-sized procedural programs.

REFERENCES

[1] P. Jorgensen. *Software Testing: A Craftsman's Approach*. CRC Press, second edition edition, 2002.

[2] A. P. Mathur. *Foundations of Software Testing*. Pearson, 2008. ISBN 978-81-317-1660-1.

[3] M. Winikoff and S. Cranefield. On the testability of BDI agent systems. *Journal of Artificial Intelligence Research (JAIR)*, 51:71–131, 2014.

[4] M. Winikoff and S. Cranefield. On the testability of BDI agent systems (extended abstract). In *Journal track of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4217–4221, 2015.

[5] H. Zhu, P. A. V. Hall, and J. H. R. May. Software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4):366–427, Dec. 1997.

[2] i.e. number of actions in the BDI program = number of statements in the procedural program