Regarding the issue of acceptable input ranges in WEST mentioned in the previous report, I conducted boundary testing with the WEST tool. I found that while it is possible to input a very large number, doing so caused my laptop to freeze. This is likely because the tool relies on the computing power of the local machine, and my lightweight laptop lacks sufficient computational resources.

To better understand this limitation, I revisited the WEST paper, which I had struggled with a few months ago. This time, I was able to grasp more of its content. From the definitions on page 3, I discovered that WEST imposes no explicit restrictions on the input range. It simply states that ***“Mission-time Linear Temporal Logic (MLTL) is a finite variation of LTL over bounded, closed, discrete intervals of the form [a, b], where a, b and 0≤a≤b.”*** This indicates that **WEST can theoretically accept any reasonable closed interval, with practical limits depending on the computer's performance.**

On page 5, I encountered the definition of ***Computation Length***, but parts of it remain difficult for me to fully comprehend. However, the section on ***Minimum Computation Length of Until and Release*** suggests that any natural number can theoretically serve as an input.

文本, 信件

描述已自动生成

文本

描述已自动生成

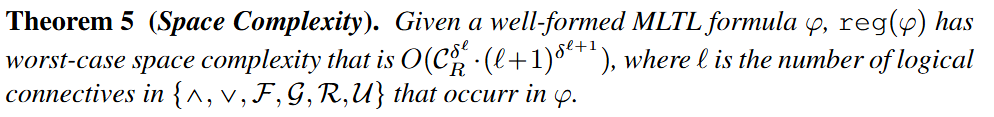
文本, 信件

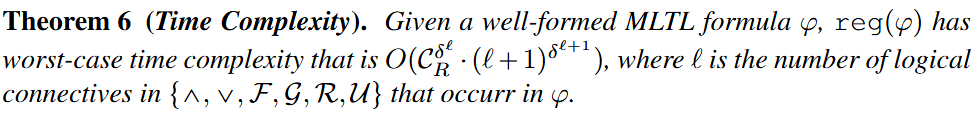
描述已自动生成

Regarding WEST's internal computation process, which involves regular expressions, the algorithm references several other algorithms. The paper does not elaborate on this process in detail, aside from some complex formulas in the appendix, which I also find challenging to interpret.

The paper does, however, provide an analysis of the algorithm's **time and space complexity** on page 11, as well as **experimental benchmarking results** on page 12. Based on this and my own testing, I can preliminarily conclude that **WEST's inability to handle long formulas stems from the limitations of local computational resources rather than issues with the algorithm itself**.

Put it in the evolution





The paper also compares WEST's algorithm to a naive brute-force approach for generating satisfying computations of MLTL formulas. The brute-force method iterates over all possible bit strings of length m⋅nm \cdot nm⋅n, where nnn is the number of propositional variables, and mmm is the computation length. Using a recursive approach to check suffix conditions, the brute-force program took nearly nine hours to process a depth-2 test suite of 1,640 formulas on a machine with an Intel(R) Core(TM) i7-4770S CPU at 3.10GHz and 32GB RAM.

In contrast, WEST executed the same test suite in under 30 minutes on the same hardware. The paper highlights WEST's efficiency, noting that while both methods produced matching results for the test suite, neither guarantees absolute correctness for all inputs.

This comparison emphasizes the significant advantage of WEST's algorithm, particularly in handling complex computations efficiently. **While the tool's performance is influenced by local hardware capabilities, the underlying algorithm showcases its strengths in both speed and practicality.**

Bounded model checking

State explosion problem

关于之前报告中提到的WEST接受范围的问题，WEST要求输入必须有一个区间范围。我首先对west工具进行了边界的测试，我发现可以输入一个较大的数字，但是较大的数字会造成我的电脑死机，我猜测这可能是因为该工具调用的是电脑内部的算力进行计算，而我电脑的算力不足，因为我的电脑是轻薄本。

于是我又把几个月前看不懂的WEST论文拿出来看了，但是这次我理解的比上次多

之后我查询了论文，发现WEST没有明确的限制输入范围，只是在论文第三页的定义中说明“Mission-time Linear Temporal Logic (MLTL) [19] is a finite variation of LTL over

bounded, closed, discrete intervals of the form [a, b] where a, b 属于 N and

0<= a<=b.”

只要是合理的闭区间west都是可以接受的，因此我认为这个具体取决于电脑的性能。

另外，我在论文的第5页找到了Computation Length definition, but to be honest, 对于以下内容我不是很理解

但是通过以下的Minimum Computation Length of Until and Release，可以初步得出，只要是自然数都可以成为输入

对于WEST内部正则表达式计算的过程，该算法引用了很多其他算法，论文里并没有详细说明，只有附件中一些复杂的公式，我对于这部分内容的理解存在一定困难。

在论文第11页中找到了该算法的时间复杂度和空间复杂度论证的过程。与此同时我也在论文的第12页找到了关于该算法的Experimental Benchmarking。基于这个和我的测试，我可以初步得出结论，为什么west为什么不能够接受长公式的输入，因为本地的算力不支持太长的公式。而算法本身应该是没有问题的

但是基于WEST也有自己的独有优势，根据我的研究，WEST的核心在于他的算法而不是工具，这个算法本身就具有一定的优势

把以下这段英文概括重述，连接前面的内容

Verifying against Naïve Brute Force A relatively straightforward approach to generating the set of all satisfying computations of an MLTL formula φ over n variables,

such that m “ cplenpφq, is to iterate over all 2m¨n possible computations, which counts

all possible length m ¨ n bit strings. An interpreter function takes computation π and

MLTL formula φ and determines if π ( φ based purely on MLTL semantics. Our test

program translates every first-order quantifier into a loop; then checking for satisfying

conditions of the suffix of a computation naturally lends itself to recursion. The full

implementation details are available in the WEST [Github]5. On an Intel(R) Core(TM)

i7-4770S CPU at 3.10GHz with 32gb RAM, the brute force program took nearly nine

hours to execute the depth two test suite of 1640 formulas. For this test suite, we fixed

the number of propositional variables at n “ 4 and the largest computation length was

m “ 5, from formulas with doubly-nested temporal operators.

In comparison, the WEST program executed the same test suite in under thirty min-

utes on the same machine. Note that the brute force program outputs only computations

of zeros and ones, and thus comparing the outputs of the WEST program requires ex-

panding out the “S” characters in the regular expressions. It is important to state that

although the full test suite matches between both implementations, absolute correctness

on all inputs is not guaranteed for either program.