A few simple tests of performance of meta-resolution

Juan Casanova

February 24, 2019

1 About this document

The aim of this document is to present and analyze a few simple tests of the performance of the implementation of the meta-resolution procedure. These tests are based on a simple working case and extended in different manners to evaluate how the running time of the program scales with these extensions.

2 Basic example

The basic example on which the measurements provided in this document is based consists in the detection of provable instantiations of the pattern:

$$\neg\exists x.A \tag{1}$$

in the following ontology:

$$\begin{array}{l} \forall x.\mathrm{margherita}(x) \implies \mathrm{pizza}(x) \\ \forall x.\mathrm{margherita}(x) \implies \neg(\exists y.\mathrm{hasTopping}(x,y)) \\ \forall x.\mathrm{pizza}(x) \implies (\exists y.\mathrm{hasTopping}(x,y) \end{array} \tag{2}$$

In CNF, the conjunction of the axioms and the negation of the conjecture consists of the following clauses:

$$\begin{array}{l} \neg\mathrm{margherita}(x) \lor \mathrm{pizza}(x) \\ \neg\mathrm{margherita}(x) \lor \neg\mathrm{hasTopping}(x,y) \\ \neg\mathrm{pizza}(x) \lor \mathrm{hasTopping}(x,\mathrm{sk}_1(x)) \\ A \end{array} \tag{3}$$

where $\mathrm{sk}_1(x)$ is a Skolem function and meta-variable A may contain Skolem constant sk_2 but no variables.

All provable atomic[1] instantiations of the meta-variable A are of the form:

$$A := \mathrm{margherita}(B) \tag{4}$$

for any term B. Due to the fact that in the signature provided to the procedure there are no free functions and only the Skolem constant sk_2, the only provable instantiation that the procedure may detect is:

$$A := \mathrm{margherita}(sk_2) \tag{5}$$

However, different proofs for this same instantiation may be found. This is discussed in more detail in §3.3.

2.1 Extending the example

The way to extend this example to verify the performance behaviour of meta-resolution, as suggested by Alan Bundy, is by introducing intermediate predicates $\mathrm{pizza}_1, \mathrm{pizza}_2, ..., \mathrm{pizza}_n$ and replacing the axiom $\forall x.\mathrm{margherita}(x) \implies \mathrm{pizza}(x)$ for the axioms:

$$\begin{array}{l} \forall x.\mathrm{pizza}_1(x) \implies \mathrm{pizza}(x) \\ \forall x.\mathrm{pizza}_2(x) \implies \mathrm{pizza}_1(x) \\ ... \\ \forall x.\mathrm{margherita}(x) \implies \mathrm{pizza}_n(x) \end{array} \tag{6}$$

or, in CNF, replacing the clause $\mathrm{margherita}(x) \vee \neg\mathrm{pizza}(x)$ for the clauses:

$$\begin{array}{l} \mathrm{pizza}_1(x) \vee \neg\mathrm{pizza}(x) \\ \mathrm{pizza}_2(x) \vee \neg\mathrm{pizza}_1(x) \\ ... \\ \mathrm{margherita}(x) \vee \neg\mathrm{pizza}_n(x) \end{array} \tag{7}$$

[1]Meta-resolution currently only detects atomic instantiations. Extension to composite formulas via inductive instantiation is not implemented yet.

3 Parameters controlled

3.1 Length of the implication chain

As explained in §2.1, we extend the example by introducing intermediate predicates and producing a longer implication chain. This parameter indicates how many intermediate predicates are introduced in each test. We will use the notation n for this parameter.

In the results presented here, we have performed measurements for values from 0 to 7 for n.

3.2 Maximum depth of proofs attempted

Meta-resolution currently has an heuristic that allows the user to specify that any proofs that exceed a certain depth (number of applications of the resolution rule) should be abandoned as improductive. We will use the notation d for this parameter. In principle, lower values of this parameter would prevent the procedure from producing a large number of improductive exploration branches. However, the value needs to be high enough so as to allow the conjecture to be provable. In our particular case, there only exists some proof if $d \geq n + 3$.

In the results presented here, we have performed measurements for values 5, 10, 15 and 20 for d.

3.3 Number of solutions

While in any of the examples tested there is only one instantiation of the meta-variable that meta-resolution will ever find ($A := \text{margherita}(sk_2)$), there are different proofs for the same instantiation. All of these will be equivalent and their variations will consists mainly in different orders in which the clauses are resolved over and/or adding spurious steps that are not relevant to find the empty clause. However, for the purposes of measuring performance of the procedure and getting an approximate idea of the size of the exploration space, it is useful to allow the procedure to output more solutions and measure the running times. We will use the notation s for this parameter.

In the results presented here, we have performed measurements for values 1, 5, 10, 20 and 50 for s.

The number of alternative proofs that there exist for a certain value of n and d is finite, and therefore some combinations of values of n, d and s are incompatible due to there being less than s different proofs.

4 Results and analysis

The results were measured on my laptop computer. The absolute numbers are not the focus of our investigation, but rather their relatives differences. Results are measured in seconds until the procedure finds up to s different solutions for the problem posed. All the values measured are presented in the following table.

Table 1: Performance times for the different values of n, d and s. In seconds. The four values in each cell indicate the measurements for $d = 5$, $d = 10$, $d = 15$ and $d = 20$ respectively. $\emptyset$ indicates that not enough solutions were present with that choice of parameters.

	$s = 1$		$s = 5$		$s = 10$		$s = 20$		$s = 50$	
$n = 0$	0.053	0.043	0.23	0.23	0.52	0.52	1.59	1.59	$\emptyset$	$\emptyset$
	0.043	0.046	0.23	0.23	0.51	0.52	1.62	1.6	$\emptyset$	$\emptyset$
$n = 1$	0.11	0.12	0.69	0.71	1.41	1.33	3.06	3.36	$\emptyset$	$\emptyset$
	0.11	0.12	0.72	0.69	1.4	1.88	3.58	3.5	$\emptyset$	$\emptyset$
$n = 2$	0.31	0.36	1.67	1.61	3.5	4	9.32	7.39	$\emptyset$	$\emptyset$
	0.38	0.28	1.43	1.44	2.77	2.76	6.8	6.73	$\emptyset$	$\emptyset$
$n = 3$	$\emptyset$	0.7	$\emptyset$	3.92	$\emptyset$	7.74	$\emptyset$	15.05	$\emptyset$	48.66
	0.52	0.51	2.62	2.6	5.32	5.31	12.9	12.94	48.1	47.77
$n = 4$	$\emptyset$	0.96	$\emptyset$	4.84	$\emptyset$	9.96	$\emptyset$	23.48	$\emptyset$	84.73
	0.97	0.96	4.85	4.83	9.96	9.93	23.35	23.32	84.62	84.78
$n = 5$	$\emptyset$	1.72	$\emptyset$	8.64	$\emptyset$	17.42	$\emptyset$	38.47	$\emptyset$	150.96
	1.71	1.72	8.63	8.64	17.46	17.46	38.43	38.5	150.84	150.68
$n = 6$	$\emptyset$	2.91	$\emptyset$	14.68	$\emptyset$	29.53	$\emptyset$	60.29	$\emptyset$	254.61
	2.92	2.92	14.8	14.62	29.49	29.45	59.95	59.98	254.2	254.39
$n = 7$	$\emptyset$	5.01	$\emptyset$	25.29	$\emptyset$	51.03	$\emptyset$	103.36	$\emptyset$	411.72
	5.02	5.0	25.31	25.32	51.11	51.11	103.28	103.20	412.06	411.74

4.1 Maximum depth

One of the first things that are easily noticeable from the results is that the maximum depth parameter d does not have the expected effect on the running times. In almost no case are the running times of the higher maximum depth tests higher than those of the lower maximum depth tests by an amount not totally explainable by experimental randomness and uncontrolled factors such as other programs that may be running on the computer.

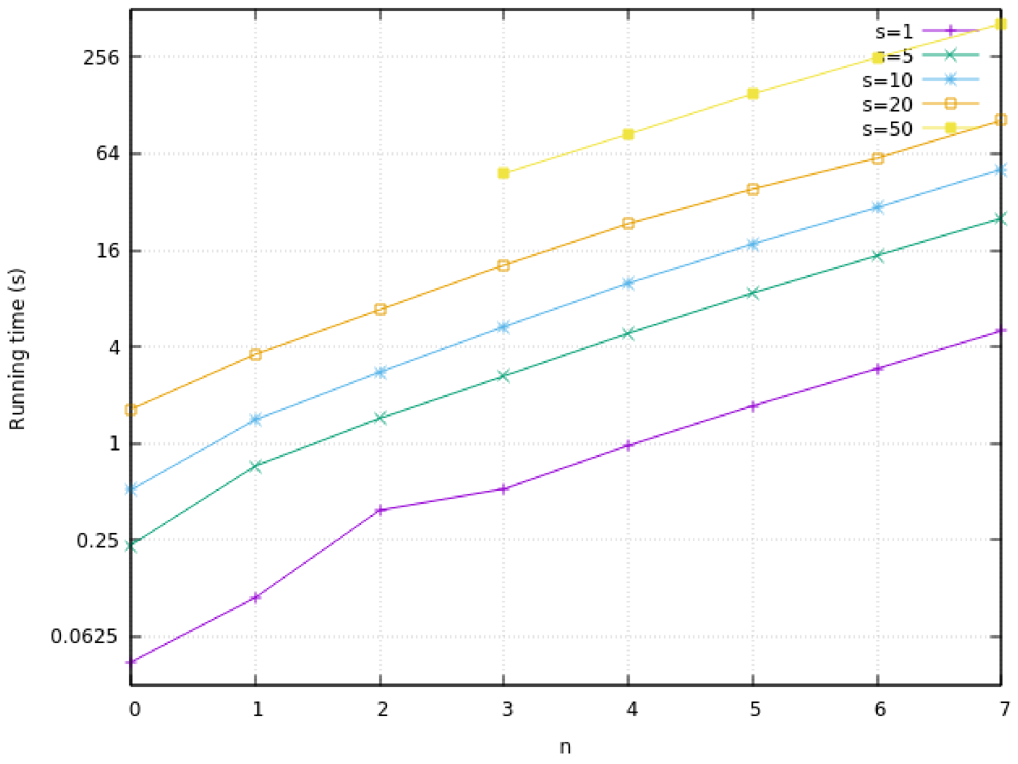
In fact, in two cases, namely for $n = 3$ when $s \neq 50$, and for $n = 2$ when $s \neq 1$, the running times of the procedure noticeably **decrease** when increasing the maximum depth parameter.

The explanation is simple: increasing the maximum depth parameter allows the procedure to find longer proofs that it was previously not able to find due to the limitation, and which come earlier in the diagonalization search than latter ones. When the depth increases past the threshold needed for that proof, the procedure finds these solutions considerably earlier because they were a lot earlier in the search. This also explains why for $n = 2$ and $s = 1$ and for $n = 3$ and $s = 50$ the differences vanish: the first proof that the procedure finds for $n = 2$ has less depth than 5, and so increasing the depth limit has no effect in the time it takes to find it, but subsequent solutions have larger depth and therefore why for $s \neq 1$ the differences are noticeably. Similarly, for $n = 3$, there are likely not many more than 50 different solutions overall, and so when the procedure is asked to find the 50 first solutions, the depth limit has little effect as it still needs to traverse most of the search space.

4.2 Length of the implication chain

The most important insight that we wanted to attain from these tests is an estimation of how the search space grows when the length of the minimum proof grows (length of the implication chain). The answer to this question seems evident from the data: the growth of the search space is exponential on the growth of the implication chain. This becomes evident by plotting the running times for different values of s as a function of n in logarithmic scale. See figure 1. Not only are all of these relations exponential, but also they seem to have the same base, independent of s, as clear by the fact that the slope in logarithmic scale is the same for all of them. More precise calculations seem to indicate that this base (slope) seems to be between 1.65 and 1.9, and slightly decrease with n and with s. That is, it seems that the running time t as a function of n has an expression of the form $t(n) = k \cdot 1.75^n$, approximately.

Figure 1: Relation between n and running time for different values of s. $d = 15$. Logarithmic scale.



4.3 Number of solutions

It is also relevant to consider how requesting more solutions to the procedure affects the running time. The relation seems to be slightly more increasing than linear, as can be observed from figure 2. That is, obtaining the next solution tends to be slightly more expensive than the last one that we just obtained.

While it is hard to extract any clear conclusions from this information at this point, considering that the diagonalization procedure makes the search grow both in breadth and in depth as it goes on, these data seem to indicate that solutions are relatively evenly spread throughout the search space, and the increased running time between each solution could be due to the increase in breadth of the search space (number of branches kept alive at the same time). However, it is also possible that this could be explained by solutions tending to appear more towards the beginning of the search space.

The most important conclusion we can extract from this, however, is that the quasi-linear growth of the running time with the number of solutions hints that the breadth growth of the search space is, in general, productive, and that it is not the case that the number of potential solutions explored grows considerably quicklier than the number of those that are actual solutions. That is, the amount of improductive search does not considerably increase during the execution of the procedure, and therefore should not, in principle, be a cause for concern.

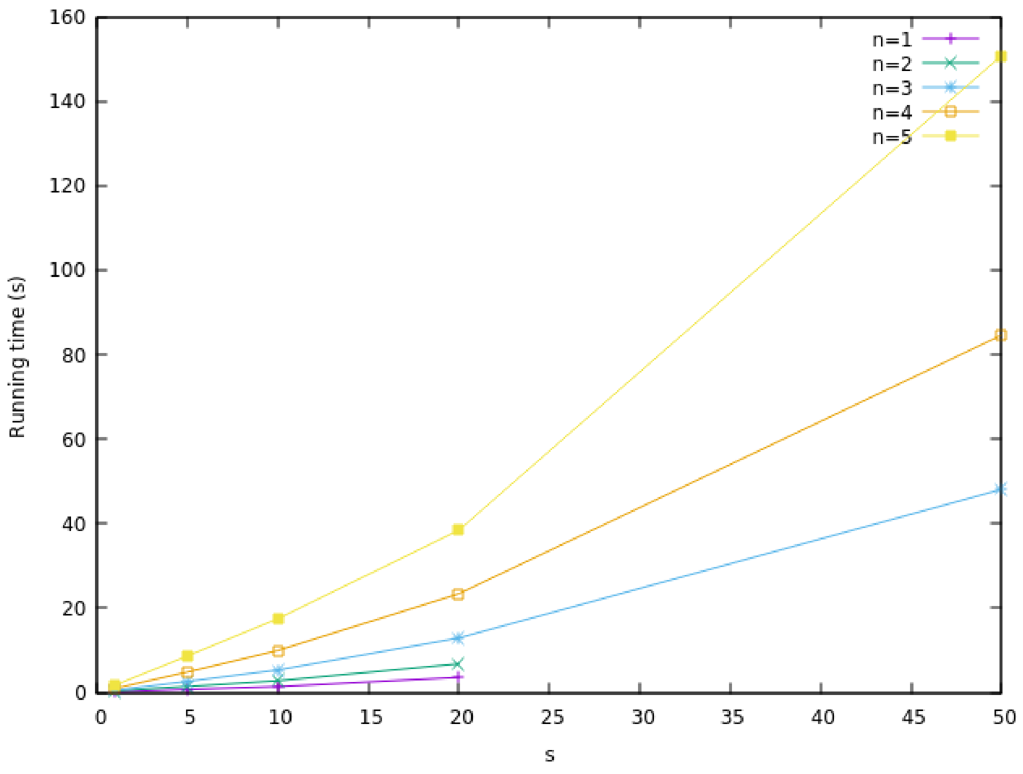
5 Limitations of the analysis

The analysis presented here is highly preliminary and clearly has important limitations.

First and perhaps most importantly, the choice of a specific and highly regular family of problems to evaluate (implication chains). It is unclear if this regularity could be masking phenomena that appear on other, more complicated, proofs and that are relevant for the performance of the procedure.

Second, the inability to measure other relevant quantities in the runs of the procedure due to the current state of the program. For example, it would be enlightening to know what the total breadth of the search space is at each point during the search (how many potential different solutions are

Figure 2: Relation between s and running time for different values of n. $d = 15$. Other values of n have similar relations and have been omitted for clarity of the plot.



being evaluated at the same time), and how this number correlates with the number of solutions found and with the running time. Similarly and in close relation, it would be interesting to know how many solutions are discarded as invalid for our problem in relation to the breadth of the search and the number of valid solutions found.

Third, the limited sample. The number of data points offered here are not enough to draw conclusive insights, even on the limited choice of problems. However, the high regularity and clear tendencies found in the results obtained seem to indicate that the limited sample might not be the biggest of issues in this case.

Finally, the inability to relevantly control factors of how the procedure runs. While there are some possibilities for heuristics to be included (and are included) in the procedure, they are primitive and do not produce a very relevant difference in the running times of the procedure. Being able to fine tune how many steps in each proof are carried out before branching out to a new potential solution or limiting or guiding the meta-variable instantiations explored could prove to be critical factors for the performance of the procedure.