— Sprint 4 —

Deadline: 5th of June

We have some bad news. We have figured out why none of our customers has complained yet. Apparently they are feeding insanely large systems to our VaaS system. Dozens of lines of code! Can you imagine that? As WolVeriNe was never intended for this kind of workload the customer computers either ran out of memory or got stuck in very long solver calls.

The analysis of this took us the better part of the last few weeks. We made some changes to WolVeriNe and the main bottleneck seems to be your solver now. We need you to significant improve your solver performance, hoping for at least three orders of magnitude.

One of our interns had a closer look at the inputs that seem to cause particularly bad performance and identified a few patterns that you should work on. We uploaded some of these examples to your Slack.

Small conflicts

Some of the examples yield individual theory calls that are unsatisfiable. Sometimes the reason for the unsatisfiability only consists of a handful of constraints. Right now the solver blindly tries many Boolean solutions that all contain the same handful constraints over and over again. Maybe if you tell the SAT solver that this small set exists and is infeasible it can do better?

We already talked to the SMT-RAT guys and apparently there already is some mechanism in place to communicate this reason which they call *infeasible subset*. You can simply add a set of formulas as such an infeasible subset by calling

```
mInfeasibleSubsets.emplace_back(...);
```

Incrementality

The SAT solver you use apparently calls your module rather frequently. It seems to deal with the Boolean structure in a somewhat clever way – we did not get the details I'm afraid – and asks your module whenever it is not completely sure what to do. Our intern noticed that these inputs to your models are usually strongly related to the input of the previous call. For example, the SAT solver oftentimes only adds a couple of new equalities and all the equalities from the previous call are still there.

Again this concept was not new to the SMT-RAT guys who called it *incrementality*. They told us that a module should in fact maintain some kind of state and only add new equalities to this state. Apparently it is possible to avoid a lot of work then.

Maybe you can work something out here? We don't know how your module works in detail of course, but adding new equalities should not be so difficult? However it seems that the SAT solver also removes equalities at times...

Reviews

As you might have noticed you are not the only team working on this issue. We actually have multiple contractors working on a backend solver that we employ in parallel. To validate your development processes we would like you to to review the code of another team.

We will mainly consider the quality of the review and provide your review to the other team. Note that a bad review has no consequences for the team under review.