0.1 Marabou coarse refinement

The marabou SMT-based tool is used to speed up queries by utilizing the reluplex algorithm. A python interface is supplied and the module is built from source to allow for customization. Whilst the source denotes an optional dependency on the 3rd party LP solver Gurobi for MILP solving, due to licensing constraints it is currently not used.

0.1.1 Building

Overall instructions can be found at https://github.com/NeuralNetworkVerification/Marabou using the latest version.

* IMPORTANT * A line must be added to MarabouCore.cpp before adding,

```
.def("__copy__",  [](const InputQuery &self) {
    return InputQuery(self);
});
```

At line 485. A file is included as example. The reasoning for this line is the current lack of support for deep copies of InputQuery objects: this is required to maintain the stack.

This is built from source using the following set of commands:

```
cd path/to/marabou/repo/folder
mkdir build
cd build
cmake ..
```

0.1.2 Overall structure

A `MarabouSolver` object acts as a wrapper around a Marabou network, translating from the Z3 format. In general this is done through `add` similarily to a Z3 solver. The few exceptions to this require the use of specialized functions `add_safepoint` and `add_counterexample` which are necessary when dealing with constraints to output variables. After applying constraints to the network, the constraints and the internal representation of the Z3 query (sexpr format) are outputted to a log file to be compared. The satisfiability of the input query is then determined using Marabou snc mode enabled, this is not entirely necessary however it marginally speeds up query speeds. The output is returned along with a model, is it exists, after which bounds are added on the input variables to the Z3 solver if the output is satisfiable.

With the model returned to the main SMLP program, bounds are established around the model using a parameter specified by the user currently - there are no formal guarantees to Marabou's precision. These bounds are implemented around the normalized values to account for the accumulation of rounding errors. Finally, Z3 checks the validity of the Marabou bounds and ultimately outputs whether the model is truly satisfiable or not, preserving the original functionality of the Z3 implementation. This is currently a design choice to formulate the Marabou model in using Z3 objects.

0.1.3 MarabouSolver work flow

The main program flow inside the MarabouSolver class is building upon a stack of InputQuery objects with the base of the stack being derived from the network and bounds. InputQuery objects are then added to top of the stack based building upon the previous layer. This is done as when a candidate is excluded it assumes an intact solver already exists and so only the threshold bounds are set.

During the `solve` method the unprocessed queries are added to the input query at the top of the stack. This input query is then run, recorded and returned, with stack operations occurring depending on the outcome of the query. In SMLP, the bounds around the model are supplied and returned to SMLP, if the models agree then the output is returned otherwise the Z3 model without the bounds is run and its output is returned.

You can detect the number of disagreements between the Z3 Solver and Marabou solver by using CRTL-F in the 'marabou.log' file which is generated and searching for "$ n". The output of this log file for each query is as follows:

Constraints added $\longrightarrow$ Constraints applied and processed $\longrightarrow$ Z3 Solver sexpr...

... $\longrightarrow$ Added equations to Marabou $\longrightarrow$ Bounds and model (if exists)

The size of the current stack is also included.

0.1.4 To Do

Marabou and Z3 seem to have the same internal constraints applied and most of, if not all bugs, have been eliminated. The current major concerns surround edge cases of which the causes are yet to be determined.

- Inconsistencies for inverse normalization of output bounds for some of Shai examples
- Segmentation fault happens randomly - could be machine dependent however work checking out

- Final model verification - should not be an issue due to the way the experiment is setup
- Removing candidate from Grid - related to Segmentation fault

Lastly, it is work moving the stack of InputQuery objects to a stack of Custom objects contained in MarabouCommon, this is due to some of the issues with working with pybind11 and the InputQuery class (C++), additionally removing applied equations is not possible and a current issue.