Breach Feature: IO Robustness

# Context

## Motivation

The robustness value of a system is hard to capture. One useful metric is the quantitative semantics (or robustness) of a given trace relative to a Signal Temporal Logic (STL) specification. However this metric can be difficult to interpret, because a single real value does not tell with respect to what quantity (input, output) or even in which physical unit the robustness is measured.

## Objective

The proposed Breach feature of IO Robustness computation, which we describe in this document, attempts to remedy this situation by focusing the analysis on a subset of the signal variables, either *input* variables or *output* variables.

# Description

## Principle

Given a set of variables, when computing the robustness in variables the other variables are treated qualitatively. We consider two possible qualitative treatments: either as having zero () robustness, or as having infinite () robustness. These interpretations are called *absolute* and *relative* robustness, respectively.

The analysis we implemented yields 4 new robustness measures:

* Input robustness, absolute;
* Input robustness, relative to fixed output;
* Output robustness, absolute;
* Output robustness, relative to fixed input.

## Interpretation

The relative robustness gives a safe approximation of the distance to satisfaction/violation based on a subset of signal variables, considering other variables fixed. The absolute robustness gives a safe approximation of this distance considering arbitrary changes in other variables.

The notion of relative robustness is most intuitive, but when input and output both contribute to the satisfaction/violation in an essential manner it cannot be measured. In this case the relative robustness does not provide a quantitative truth value, but the absolute robustness may.

For output variables, relative robustness and absolute robustness capture two extreme assumptions about the system.

1. Relative: the output is unaffected by under input perturbations;
2. Absolute: the output can change in an arbitrary fashion with input perturbations.

# Implementation

We provided Breach with a proof-of-concept implementation of the IO robustness feature. The implementation provides two (almost independent) enhancements to Breach. The first enhancement associates with an STL formula two lists of input and output signals. The second enhancement consists in computing the robustness according to inputs or outputs, using the relative or absolute interpretation.

## Changes to standard implementation

We modified the parsing of STL formulas to include information about input and output signals, which is then associated with every STL subformula. This way the names of input and output signals can be queried at any point when parsing an STL formula, typically during the robustness computation itself. This was done by directly modifying the STL formula constructor and class to include IO information, and by implementing new setter and getter function to access this IO information.

The robustness computation is changed by adding additional parameters controlling the set of signals over which the analysis is focused (*input* or *output*) and the interpretation (*relative* or *absolute*), and changing the semantics of atomic predicates that are not the focus of the analysis to either zero or infinity. This was done by modifying the Breach System class to add additional methods suffixed with IO and accepting the additional parameters. This is then propagated to the core computation (via plotting subroutines) in the evaluation functions, which are also duplicated with IO suffix and accepting the additional parameters.

## Use model

The intended use of the IO robustness indicators is as follows. The user adds information about input and output signals in the .stl file. The syntax of .stl files have been extended with the keywords

input signal   
output signal

Alternatively, the user may call the functions:

set\_in\_signal\_names(phi, X)   
set\_out\_signal\_names(phi, Y)

over the formula phi and a cell array of signal names X or Y. If needed, the corresponding getter methods can be used to show lists of input and output signals.

The robustness can be computed by using the method

GetIORobustSat(phi, params, values, t\_phi, inout, relabs)

of the class BreachSystem, where the new fields inout and relabs are string switches with possible values ‘in’, ‘out’ and ‘rel’, ‘abs’ respectively.

The robustness can also be computed and displayed using the method

PlotIORobustSat(phi, inout, relabs, depth, tau, ipts)

of the class BreachSystem, where depth, tau, ipts are optional parameters and inout and relabs are strings with the meaning as previously.

## List of files

### Modified

BreachSystem.m, STL\_Formula.m

### Created

SplotSatIO.m, STL\_Eval\_IO.m, STL\_EvalThom\_IO.m, get\_in\_signal\_names.m, get\_out\_signal\_names.m, set\_in\_signal\_names.m, set\_out\_signal\_names.m, is\_in\_signal.m, is\_out\_signal.m.

## Current limitations

* When using absolute robustness, the signals should be partitioned into input and output (no third kind of signals);
* The robustness heat-map PlotRobustMap (and auxiliary function GetSatValues) is not supported;
* BreachRequirement is not supported.