A Brief Experimental Comparison of the STC and LGG Analysis Algorithms in SpaceEx

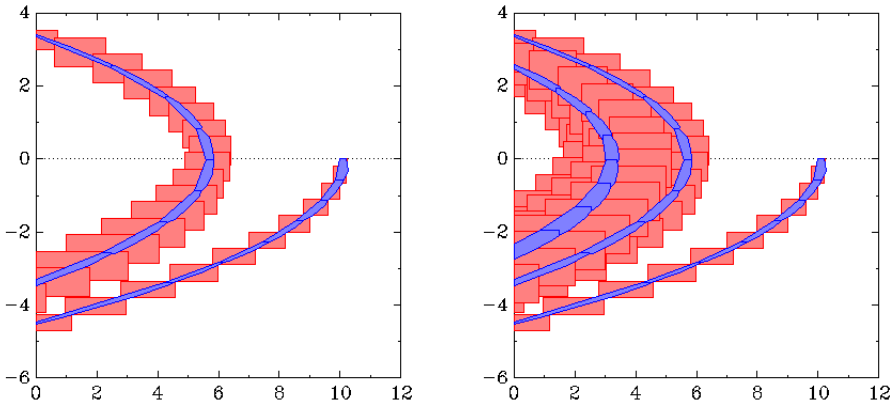
Goran Frehse, Verimag

26/11/2012

1 Introduction

SpaceEx is a verification platform for hybrid systems. The goal is to verify (ensure beyond reasonable doubt) that a given mathematical model of a hybrid system satisfies the desired safety properties. The basic functionality is to compute the sets of reacheable states of the system. The main analysis algorithm of SpaceEx is an implementation of the Le Guernic-Girard (LGG) algorithm [2]. Classic reachability algorithms use operators whose computational cost is exponential in the number of continuous variables, which limits them to systems with 3–5 continuous variables for nontrivial dynamics. The operators of the LeGuernic-Girard algorithm are polynomial, and systems with 200 variables and more have been analyzed that way. The version of the LGG algorithm that is implemented in SpaceEx uses outer polyhedral approximations to compute the image of discrete transitions, making it scalable [1]. The STC algorithm is a recent enhancement of the LGG algorithm that produces fewer convex sets for a given accuracy and computes more precise images of discrete transitions. Based on the LGG scenario, the reachable states over time (so-called flowpipes) are bounded with piecewise linear approximations of the support function over time. The approximation can be made arbitrarily precise using just two parameters: the number of template directions and an error bound that is evaluated in each of those directions. The image of discrete transitions is computed on polyhedral overapproximations that are projections from space-time onto the state space. The complexity (number of constraints) and number of polyhedra is adjusted automatically so the given error bound is met. This makes the clustering heuristics (commonly used with the LGG scenario) unnecessary in most cases and gives more predictable results than the hit-or-miss success rate of previous heuristics.

This document presents some experiments that compare the performance of STC and LGG algorithms from a purely practical perspective. The comparison is not easy to do objectively. Notably, the error and tolerance measures for both algorithms differ. Both the accuracy of the final output and



(a) 1 jump

(b) 2 jumps

Figure 1: Flowpipe cover of a bouncing ball with template directions in the axis directions (box directions), computed with tolerance value $\epsilon = 1.0$. The outer approximation of the LGG algorithm is shown red, that of the STC algorithm in blue. The plot shows position (horizontal) over velocity (vertical)

the computation time should be taken into account. This is complicated by the fact that both algorithms use an implicit internal representation (support functions) whose visualization is itself an overapproximation. Having a precise graphical output can take more time than the actual reachability computation. The reported computation times do not include the visualization step. Note that some experiments have been carried out with protoype versions, so performance results might be different from the latest release.

2 Illustrative Comparison: Bouncing Ball

In the following we consider the reachable states of a bouncing ball, for a given number of jumps of the ball. We compare two versions: LGG and the new proposed approach STC. Both have internal representations of sets that are implicit. In LGG, this is a single support function per convex set. In STC, it is a set of piecewise linear functions that describe the evolution of the support function over time for a given set of template directions. In both versions, an outer polyhedral approximation is used for the intersection. In LGG, this means the outer hull of the template directions (here box directions). In STC, this means finding a polyhedral cover of the internal flowpipe approximation, which is then projected onto the state space. This cover is constructed such that the minimum number of convex polyhedra is returned that meets the given tolerance.

Figure 1(a) shows the result for the reachable states for the first jump of the ball, more precisely, the outer approximations covering the flowpipe. In

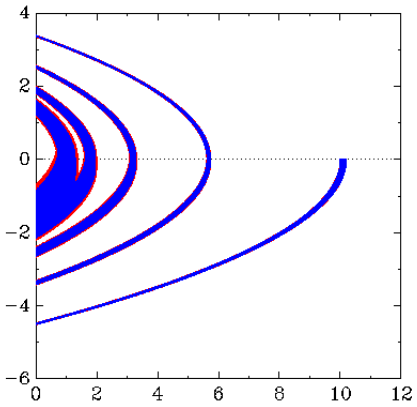


Figure 2: Flowpipe cover of a bouncing ball with template directions in the axis directions (box directions), computed with tolerance $\epsilon = 10^{-4}$. For a small tolerances, both show similar results but LGG computes 457 448 support function values while STC only computes 966. The outer approximation of the LGG algorithm is shown red, that of the STC algorithm in blue. The plot shows position (horizontal) over velocity (vertical)

both cases, the template directions are given as the axis directions, so the outer approximations are bounding boxes. For the STC algorithm, these are sets of boxes in space-time (a box at every point in time) that are projected down to the state space and therefore contain non-box constraints. Note that this projection can be carried out cheaply using support functions. Before the jump, LGG and STC more or less coincide. The jump takes place at the guard x=0 (vertical line). The image of the jump is computed using the outer approximation, so the box-shape of LGG introduces a large error that propagates. If we add one more jump, shown in Fig. 1(a), we see that each of the three red boxes that intersect with the guard has spawned its own flowpipe. For more jumps, this number increases quasi-exponentially, so clustering sets is necessary to avoid stalling. From now on, we take in LGG the convex hull of the states that can jump. That way, we avoid an explosion in the number of sets for sure. But what will the approximation error be? For LGG, the approximation error is so large the flowpipe goes to infinity after only a few jumps.

We know that as the time step in LGG goes to zero, its outer approximation approaches that of STC the same way a Euler approximation approaches a piecewise linear solution. The result for a small tolerance $\epsilon = 10^{-4}$, shown in Fig. 2, confirms this as the two approaches give similar results. But here is an excerpt from the log of LGG, showing the result of the image computations of the discrete jumps:

```
00:00:00.088 adding convex hull of 10 polyhedra to result
00:00:00.842 adding convex hull of 16 polyhedra to result
```

```
00:00:01.733 adding convex hull of 36 polyhedra to result
00:00:03.280 adding convex hull of 82 polyhedra to result
00:00:06.223 adding convex hull of 184 polyhedra to result
```

LGG needs to cluster a progressively large number of sets – this grinds progress eventually to a halt. As can be seen from the time stamp, computation time practically doubles with each jump. After each jump, the initial set of the next flowpipe is the convex hull of up to 184 sets. The convex hull is a "lazy" function – computing the support function of it actually means computing the support of up to 184 sets and taking their maximum. In consequence, the effort required per step is progressively increased. Despite taking adaptive time steps and other performance tricks, LGG computes a total of 457 448 support function values for the case shown in Fig. 2.

In STC, on the other hand, computes a total of 966 support function values. New support values are only computed where a linear interpolation is not precise enough. Note that both approaches take about the same minimal time step to achieve the desired accuracy (LGG: 0.0052, STC: 0.0048). The total computation time in LGG is 6.3s, in STC it is 0.7s. Computing more jumps widens this gap even more.

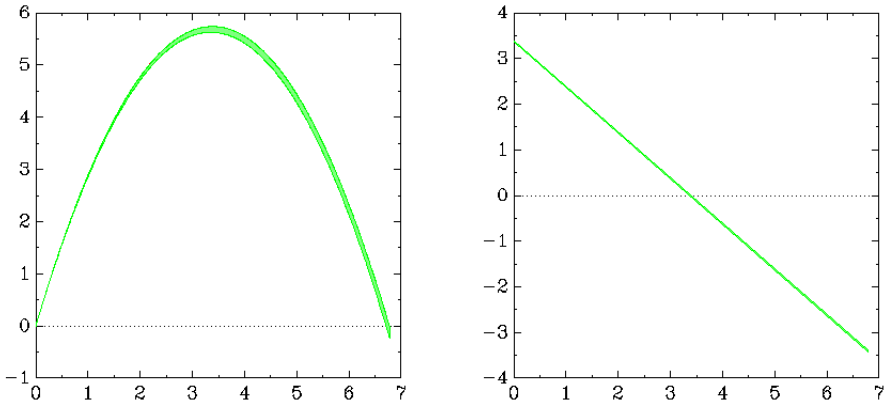
Another reason why the number of evaluations is reduced in STC is because the time steps are adapted separately for each template direction. The flowpipe is represented internally by an interval-valued piecewise linear function over time. The real solution is known to lie inside the interval. The error of the function is the width of the interval, and it decreases each time an additional breakpoint is computed for the function. To optimize performance, the computation is done by refinement: points are added until the error falls below the given tolerance. A closer look shall illustrate this point. For the tolerance $\epsilon = 10^{-4}$ given above, the computed breakpoints per direction after the first jump show in the log as follows (note that one support value can yield several breakpoints):

```
00:00:00.148 computing evolution in direction [x=1,v=0]
00:00:00.260 with 8193 points
00:00:00.260 computing evolution in direction [x=-1,v=0]
00:00:00.280 with 1025 points
00:00:00.280 computing evolution in direction [x=0,v=-1]
00:00:00.280 with 2 points
00:00:00.280 computing evolution in direction [x=0,v=1]
00:00:00.281 with 2 points
```

Recall that the trajectories of the bouncing ball in free fall are

$$x(t) = x(0) - 0.5t^2, \qquad v(t) = v(0) - t.$$

The evolution of the support function in v-direction is therefore a straight line. The STC scenario computes the first and the last point on the line, realizes that it has computed these with error zero, and stops without adding



(a) bounds on x over time

(b) bounds on v over time

Figure 3: Bounds on the evolution of x and v over time for the first flowpipe, computed with tolerance value $\epsilon = 10^{-4}$. The inner bounds (values definitely reachable) are shown in green, outer bounds (maybe reachable) are shown in red (barely visible at this high accuracy)

further points. We can plot these evolutions for the axis directions to obtain an envelope of the values of each variable over time: The bound on the support function in the x-direction provides an upper bound on the variable x over time, while the bound in the negative x-direction provides a (negated) lower bound on x.

Figure 3 shows these evolutions for the flowpipe after the first jump. The inner bound of the variable is shown in green, these are the values we know it definitely takes at some point. In red, we have the outer bound, which are the values the variable might take. If only green and no red is visible, it means that both coincide. Note that the error of the inner bound is measured per jump, however, so it does not give any globally valid information about what values are actually taken. The shown evolution for x (position) has 8193 points on the upper and 1025 points on the lower bound. For v, both upper and lower bound consist of 2 points each.

These evolution plots are natural extensions from simulation plots to sets, and could given useful feedback to modelers and verifiers. Note that there is one plot for every computed flowpipe, so one would need a good way of navigating between them. Compared to state-space plots, they are surprisingly accurate.

3 Filtered Oscillator

The filtered oscillator is a parameterized benchmark, in which the number of continuous variables can be varied [1]. A two-dimensional switched linear

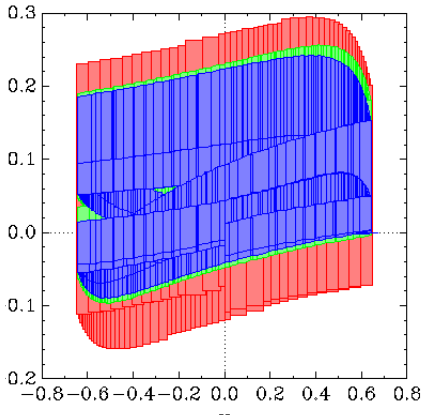
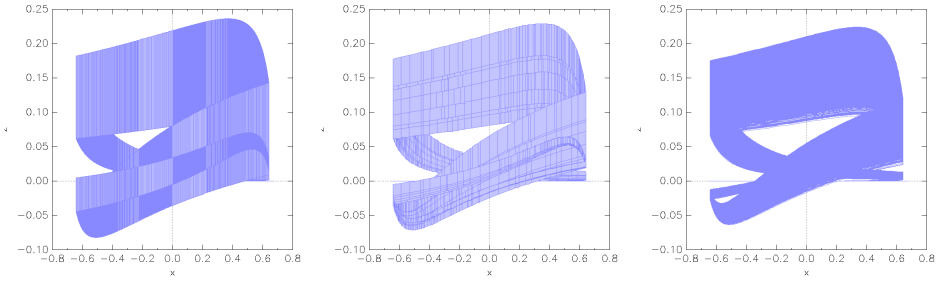


Figure 4: Flowpipe cover of a filtered oscillator with 34 continuous state variables, computed with tolerance value $\epsilon = 0.01$. The template directions are the axis directions (box directions). The outer approximation of the LGG algorithm is shown red, that of the STC algorithm in blue, and in green is shown the STC algorithm with convex hull clustering similar to LGG. The plot shows oscillator state variable x (horizontal) over output variable z (vertical)

oscillator is connected to an n-th order filter, so that the system has $n + 2$ continuous variables. In the following we will compare scalability and precision of LGG and STC on this example. Note that the behavior of the example changes with n: the filtered output converges slower to the limit cycle. As a consequence the analysis algorithm takes more iterations to find a fixed point, even if all other parameters are kept equal.

All instances of the filtered oscillator require some kind of clustering, since the first discrete state change involves dozens to hundreds of the convex sets that cover the flowpipe. This example is fairly robust with respect to the clustering algorithm used, and both taking the convex hull as well as taking the template hull of the sets gives acceptable results for the LGG algorithm (template hull being faster). The STC algorithm on the other hand choses itself the number of sets necessary to meet the given accuracy. For the comparison, we have also included a version of STC with convex hull clustering.

Figure 5 shows a projection of the reachable states of the 34-variable filtered oscillator to the oscillator state variable x and the output variable z. Three algorithms are shown: The outer approximation of the LGG algorithm, that of the STC algorithm, and for purposes of comparison the STC algorithm with convex hull clustering similar to LGG. The LGG with convex hull clustering takes 5.8s, STC with convex hull clustering takes 6.3s and standard STC takes 11.6s. For a tolerance of $\epsilon = 10^{-2}$, the STC algorithm is about twice as slow since it produces 2 convex sets at the first transition, which then propagate without further splitting as two flowpipes until termi-



(a) LGG, $\epsilon = 10^{-4}$ (b) STC, $\epsilon = 10^{-3}$ (c) 2000 simulations

Figure 5: Flowpipe cover of a filtered oscillator with 34 continuous state variables, comparing LGG and STC for same computation time, plus simulation runs for reference. The template directions are the axis directions (box directions). The plot shows oscillator state variable x (horizontal) over output variable z (vertical)

nation. In total, STC carries out 26 flowpipe computations, while the LGG, which due to clustering only propagates one set, carries out 13. However, if we measure the accuracy as the output amplitude and the simulation runs in Fig. 5(c) as reference, the STC algorithm terminates with a total error of 7%, while STC with convex hull clustering has a total error of 14%, and LGG with convex hull clustering has a total error of 31%. For a tolerance of $\epsilon = 10^{-4}$, LGG produces in 80.8s a total error of 4.4%, see Fig. 5(a). With $\epsilon = 10^{-3}$, STC produces in 76.2s a total error of 2%, see Fig. 5(b), covering the first jump with 7 convex sets.

References

[1] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In G. Gopalakrishnan and S. Qadeer, editors, *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[2] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250 – 262, 2010. IFAC World Congress 2008.