SPeeDI – a Verification Tool for Polygonal Hybrid Systems ⋆

E. Asarin[1], G. Pace[2], G. Schneider[1], and S. Yovine[1]

[1] VERIMAG, 2 av. Vignate, 38610 Gières, France
[2] INRIA Rhone-Alpes / VASY, 655 av. de l'Europe, 38330 Montbonnot, France
{asarin,gerardo,yovine}@imag.fr; gordon.pace@inria.fr

1 Introduction

Hybrid systems combining discrete and continuous dynamics arise as mathematical models of various artificial and natural systems, and as an approximation to complex continuous systems. A very important problem in the analysis of the behavior of hybrid systems is reachability. It is well-known that for most non-trivial subclasses of hybrid systems this and all interesting verification problems are undecidable. Most of the proved decidability results rely on stringent hypothesis that lead to the existence of a finite and computable partition of the state space into classes of states which are equivalent with respect to reachability. This is the case for classes of rectangular automata [1] and hybrid automata with linear vector fields [2]. Most implemented computational procedures resort to (forward or backward) propagation of constraints, typically (unions of convex) polyhedra or ellipsoids [3, 4, 5]. In general, these techniques provide semi-decision procedures, that is, if the given final set of states is reachable, they will terminate, otherwise they may fail to. Maybe the major drawback of set-propagation, reach-set approximation procedures is that they pay little attention to the geometric properties of the specific (class of) systems under analysis.

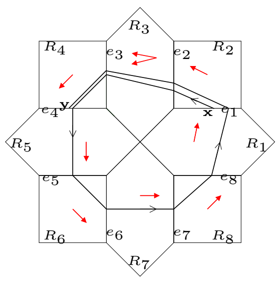


Fig. 1. SPDI.

An interesting and still decidable class of hybrid system are the (2-dimensional) polygonal differential inclusions (or SPDI for short). An *SPDI* (Fig. 1) is defined by giving a finite partition $\mathbb{P}$ of the plane into convex polygonal sets, and associating with each $P \in \mathbb{P}$ a couple of vectors $\mathbf{a}_P$ and $\mathbf{b}_P$. The SPDI is $\dot{\mathbf{x}} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ for $\mathbf{x} \in P$, where $\angle_{\mathbf{a}}^{\mathbf{b}}$ denotes the angle on the plane between the vectors $\mathbf{a}$ and $\mathbf{b}$. In [6] we have proved that (point-to-point, edge-to-edge and polygon-to-polygon) reachability is decidable and we have proposed a decision procedure that exploits the topological properties of the plane. Our procedure is not based on the computation of the reach-set but rather on the exploration of a finite number of types of qualitative behaviors obtained from the edge-signatures of trajectories (i.e., the sequences of their intersections with the edges of the polygons). Such types of signatures may

⋆ Partially supported by Projet IMAG "Modélisation et Analyse de Systèmes Hybrides", by Projet CNRS MathSTIC "Analyse Qualitative de Systèmes Hybrides" and by the European Research Consortium in Informatics and Mathematics (ERCIM).

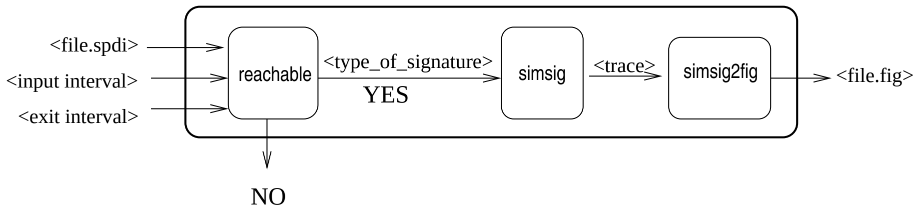


Fig. 2. Workflow of the tool.

contain loops which can be very expensive (or impossible) to explore naively. However, we have shown that loops have structural properties that are exploited by our algorithm to efficiently compute the effect of such loops. In summary, the novelty of the approach is the combination of several techniques, namely, (1) the representation of the two-dimensional continuous dynamics as a one-dimensional discrete dynamical system, (2) the characterization of the set of qualitative behaviors of the latter as a finite set of types of signatures, and (3) the "acceleration" of the iterations in the case of cyclic signatures.

2 SPeeDI

The tool SPeeDI is a collection of utilities to manipulate and reason mechanically about SPDIs, completely implemented in 5000 lines of Haskell [7], a general-purpose, lazy, functional language.

Visualization aids: To help visualize systems, the tool can generate graphical representations of the SPDI, and particular trajectories and signatures within it.

Information gathering: SPeeDI calculates edge-to-edge successor function composition and enlist signatures going from one edge to another.

Verification: The most important facet of the tool suite is that of verification. At the lowest level, the user may request whether, given a signature (with a possibly restricted initial and final edge), it is a feasible one or not. At a more general, and useful level, the user may simply give a restricted initial edge and restricted final edge, and the tool attempts to answer whether the latter is reachable from the former.

Trace generation: Whenever reachability succeeds SPeeDI generates stripes of feasible trajectories using different strategies and graphical representation of them.

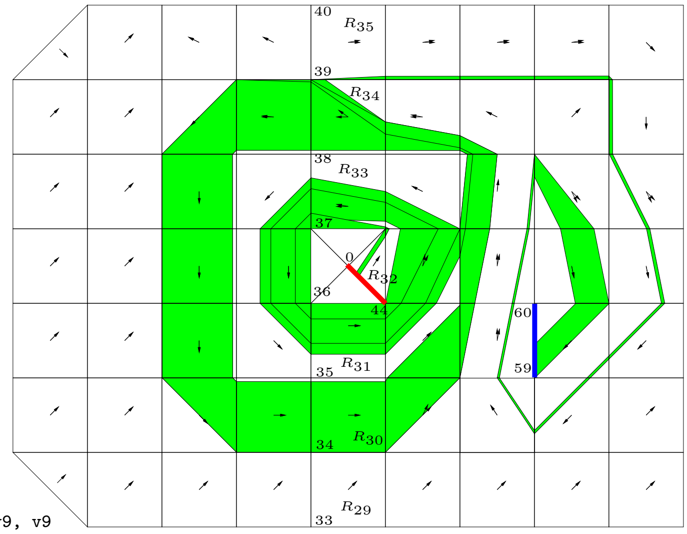
This typical usage sequence of the tool suite is captured in Figure 2.

Figure 3 illustrates a typical session of the tool on an example SPDI composed of 63 regions. The left part of the diagram shows selected portions of the input file, defining vectors, named points on the x-y plane, and regions (as sequences of point names, and pairs of differential inclusion vectors). The lower right-hand

Input file

```
Points: 0. 0.0, 0.0
* ...
33. -5.0, -35.0
34. -5.0, -25.0
35. -5.0, -15.0
36. -5.0, -5.0
37. -5.0, 5.0
38. -5.0, 15.0
39. -5.0, 25.0
* ...
Vectors:
* ...
v3. -1,0.1833333333
v8. 1,0
v9. 1,1
v12. 1, 1.5
v20. -1, 0.001
v22. 1,-0.001
v25. -1,0.7
v28. 1, 0.001
*...
Regions:
* ...
* R29
33 ? 41 ! 42 ! 34 ? 33, v9, v9
* R30
34 ! 42 ! 43 ? 35 ? 34, v22, v22
* R31
35 ? 36 ? 0 ! 44 ! 43 ! 35, v8, v8
* R32
44 ! 45 ! 0 ? 44, v12, v12
* R33
0 ? 45 ? 46 ! 38 ! 37 ! 0, v3, v20
* R34
38 ? 46 ? 47 ! 39 ! 38, v25, v20
* ...
```

Generated Figure



Session log

```
% reachable example.spdi "[1,2]" "[0,10]" 0-44 59-60
Generating and trying signatures from edge 0-44 to 59-60
Starting interval:[1.0,2.0] Finishing interval:[0.0,10.0]

(0-44,45-44)  (45-53,45-46,37-38,...,36-35,44-43,44-52)*
(53-52,53-61,54-62,54-55,46-47)(38-39,..., 46-47)* (39-47,
...,68-60,59-60)   <REACHABLE>
```

Fig. 3. Example

panel shows the signature generated by the tool `reachable` which satisfies the user's demand. The signature has two loops which are expressed with the star symbol. A trace is then generated from the signature using `simsig`. It traverses three times the first loop and two times the second one. The graphical representation of the SPDI and the trace is generated automatically using `simsig2fig`. The execution time for this example is a few seconds.

3 Comparing and Contrasting with HyTech

While SPeeDI is, as far as we know, the only verification tool for hybrid systems implementing a decision algorithm (with the exception of timed automata), it is interesting to compare it to "semi-algorithmic" hybrid system verification tools such as HyTech [8]. HyTech is a tool capable of treating hybrid linear systems of any dimension, making it much more general than SPeeDI, which is limited to two-dimensional systems without resets. On the other hand, SPeeDI implements acceleration techniques (based on the resolution of fix-point equations) which yield a complete decision procedure for SPDIs. Also, SPeeDI does not handle arbitrary polyhedra, but only polygons and line segments. For these reasons, comparing the performance of the two tools is meaningless and no fair benchmarking is really possible.

We can only compare the two tools is we restrict ourselves to SPDIs. From some experiments we have run, we have reached a number of qualitative conclusions:

- It is well known that since HyTech uses exact rational arithmetic, it can it easily run into overflow problems. This is particularly an issue when the path to the target passes through a large number of regions. This makes verification of non-trivial sized SPDIs (eg the one in figure 3) impossible.
- In the case of loops, SPeeDI calculates the limit interval without repeatedly iterating the loop. It makes use of this interval to accelerate the reachability analysis, avoiding time consuming loop traversals. In contrast, HyTech performs these iterations. Following the loops explicitly, easily leads to overflow problems, and, more seriously, in certain (even simple) configurations, this analysis never terminates.

While the first issue is limited to HyTech, the second is inherent to any tool based on non-accelerated reachability analysis. On examples which HyTech can handle, the two tools take approximately the same amount of time (a fraction of a second) to reach the result. SPeeDI, however, can handle much larger examples.

4 Discussion

We have presented a prototype tool for solving the reachability problem for the class of polygonal differential inclusions. The tool implements the algorithm published in [6] which is based on the analysis of a finite number of qualitative behaviors generated by a discrete dynamical system characterized by positive affine Poincaré maps. Since the number of such behaviors may be extremely big, the tool uses several powerful heuristics that exploit the topological properties of planar trajectories for considerably reducing the set of actually explored signatures. When reachability is successful, the tool outputs a visual representation (in the form of an Xfig file) of the stripe of trajectories that go from the initial point (edge, polygon) to the final one.

Despite the fact that functional languages, especially lazy ones, have a rather bad reputation regarding performance (see for example, [9] for a report on the experiences of writing verification tools in functional languages), we found that the performance we obtained was more than adequate for the magnitude of examples we had in mind. Furthermore, we feel that with the gain in the level of abstraction of the code, we have much more confidence in the correctness of our tool had we used a lower level language. We found laziness particularly useful in separating control and data considerations. Quite frequently, optimizations dictated that we evaluate certain complex expressions at most once, if at all. In most strict languages, this would have led to complex code which mixes data computations (which use the values of the expressions) with control computation (to decide whether this is the first time we are using the expression and, if so, evaluate it). Thanks to shared expressions and laziness, all this came for free — resulting in cleaner code, where the complex control is not done by the programmer.

Future work previews the integration of SPeeDI into a large tool suite for qualitative analysis of hybrid systems. We plan to extend its functionality beyond reachability verification. In particular, we are currently working on the implementation of the algorithm developed in [10] for constructing the phase portrait of an SPDI which is composed of viability and controllability kernels.

References

[1] Henzinger, T., Kopke, P., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: STOC'95, ACM Press (1995) 373–382

[2] Lafferriere, G., Pappas, G., Yovine, S.: Symbolic reachability computation of families of linear vector fields. Journal of Symbolic Computation **32** (2001) 231–253

[3] Asarin, E., Bournez, O., Dang, T., Maler, O.: Reachability analysis of piecewise-linear dynamical systems. In: HSCC'00, LNCS 1790, Springer (2000) 20–31

[4] Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: A model checker for hybrid systems. Software Tools for Technology Transfer **1** (1997) 110–122

[5] Kurzhanski, A., Varaiya, P.: Ellipsoidal techniques for reachability analysis. In: HSCC'00, LNCS 1790, Springer (2000)

[6] Asarin, E., Schneider, G., Yovine, S.: On the decidability of the reachability problem for planar differential inclusions. In: HSCC'01, LNCS 2034, Springer (2001)

[7] Jones, S.P., Hughes, J.: Report on Haskell 98: A non-strict, purely functional language (1999) available from `http://www.haskell.org`.

[8] Henzinger, T., P.-H.Ho, H.Wong-toi: Hytech: A model checker for hybrid systems. Software Tools for Technology Transfer **1** (1997)

[9] Leucker, M., Noll, T., Stevens, P., Weber, M.: Functional programming languages for verification tools: Experiences with ML and Haskell. In: Proceedings of the Scottish Functional Programming Workshop (SFPW'01). (2001)

[10] Asarin, E., Schneider, G., Yovine, S.: Towards computing phase portraits of polygonal differential inclusions. In: HSCC'02, LNCS 2289, Springer (2002) 49–61